Περιεχόμενα

Μέρος Ι

Κεφάλαια

Περιεχόμενα

1. Από το πρόβλημα στην ανάπτυξη αλγορίθμου

Εισαγωγή

Στο κεφάλαιο αυτό, αρχικά θα προσεγγίσουμε θέματα που αφο- ρούνα~~φορούν~~ την αναγνώρισηα~~ναγνώ ριση~~ της δομής ενός προβλήματος και της πο- λυπλοκότητάς~~πολυπλοκότητάς~~ του. Στη συνέχεια, θα γνωρίσουμε τη διαδικασία της αφαίρεσης και της απλοποίησης~~την απλοποίηση~~ ενός προβλήματος με την ανάλυσή του σε απλούστερα υποπροβλήματα. Τέλος, θα γνωρί- σουμε~~γνωρίσουμε~~ τον τρόπο που περιγράφουμε αλγοριθμικά τη λύση του, εκφρασμένη με ψευδοκώδικα ή διάγραμμα ροής.

Λέξεις κλειδιά

Πρόβλημα, επίλυση προβλήματος, ανάλυση προβλήματος, αφαί- ρεση, πολυπλοκότητα~~αφαίρεση, πολυπλοκό τητα~~, αλγόριθμος, αναπαράσταση αλγορίθμου.

Διδακτικές Ενότητες

Για να αναπτύξουμε προγράμματα που μας βοηθούν στην επίλυ- ση~~επίλυση~~ των προβλημάτων~~προβλη μάτων~~, είναι αναγκαία η κατανόηση τριών βασι- κών~~βασικών~~ εννοιών: της έννοιας του *προβλήματος*~~*προ βλήματος*~~, της έννοιας του *αλ- γορίθμου*~~*αλγορίθμου*~~ (algorithm) για τη περιγραφή της λύσης του και της *δο- μής*~~*δομής*~~ *δεδομένων* (data structure), για να μπορούν να χρησιμοποιη- θούν~~χρησιμοποιηθούν~~ τα δεδομένα από τον αλγόριθμο. Η *σχεδίαση αλγορίθμων* αποτελεί σημαντικό μέρος της διαδικασίας *επίλυσης ενός προβλή- ματος* ~~*προβλήματος*~~. Συνοδεύεται με την ανάλυση~~ανάλυσή~~ του προβλήματος~~προ- βλήματος~~ σε απλούσ- τερα~~απλούστερα~~, τη σύνθεση των λύσεων των επιμέρους προβλημάτων και κυρίως με την περιγραφή και μορφοποίηση της λύσης του. Αυτό, ώστε η λύση να μπορεί να αναπαρασταθεί σε μορφή κατανοητή από τον υπολογιστή.

1.1 Εισαγωγή στη διαχείριση της πολυπλοκότητας ενός προβλήματος

Ο άνθρωπος αντιμετωπίζει καθημερινά μικρά ή μεγάλα προβλή- ματα~~προβλήματα~~, πολλά από τα οποία προσπαθεί να τα επιλύσει κατά το δυ- νατόν~~δυνατόν~~ ευκολότερα με τη βοήθεια των εργαλείων και την τεχνολογία της εποχής του.

*Με τον όρο **πρόβλημα** προσδιορίζεται μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι ούτε γνωστή, ούτε προφανής.*

Τα προβλήματα μπορούμε να τα διακρίνουμε, με κριτήριο τη δυνατότητα επίλυσης, σε *επιλύσιμα*, όταν η λύση τους είναι γνωστή, *ανοιχτά*, όταν η λύση τους δεν έχει βρεθεί και παράλληλα δεν έχει αποδειχτεί ότι υπάρχει και *άλυτα*, όταν έχει αποδειχθεί ότι αυτή δεν υπάρχει.

Τα προβλήματα χαρακτηρίζονται επίσης, ως απλά ή σύνθετα. Εκείνα που η λύση είναι εύκολο να βρεθεί χαρακτηρίζονται ως *απλά*. Ως *σύνθετα* χαρακτηρίζονται τα προβλήματα που αποτελούν- ται από πολλά μέρη και στη λύση τους συμμετέχουν πολλοί παράγοντες, που συχνά αλληλεπιδρούν μεταξύ τους. Για παράδειγμα, προβλήματα όπως η ρύπανση του περιβάλλοντος, ή η ασφαλής πλοήγηση στο Διαδίκτυο χαρακτηρίζονται ως σύνθετα προβλήματα.

Τα προβλήματα που απαντώνται στην καθημερινότητα και στις επιστήμες ποικίλουν, με ορισμένα από αυτά να μπορούν να επιλυθούν με τη βοήθεια υπολογιστή. Αυτά τα προβλήματα σχηματίζουν μια ευρύτερη κατηγορία με την ονομασία, *υπολογιστικά προβλήματα*, προβλήματα με τα οποία θα ασχοληθούμε στη συνέχεια.

1.2 Στάδια επίλυσης προβλήματος

Η επίλυση ενός προβλήματος εξελίσσεται σε τρία στάδια: α) της *κατανόησης*, β) της *ανάλυσης* και γ) του *σχεδιασμού και της υλοποίησης* της λύσης.

Στάδιο Α. Κατανόηση προβλήματος

Στο πρώτο στάδιο, της κατανόησης του προβλήματος, γίνεται η διάκριση της δομής του προβλήματος, διαδικασία αναγκαία, ειδικά σε ένα σύνθετο πρόβλημα. Με τον όρο *δομή*, εννοούμε τα επιμέρους στοιχεία-τμήματα που αποτελούν το πρόβλημα, καθώς και τον τρόπο με τον οποίο αυτά συνδέονται και αλληλεπιδρούν. Για την κατανόηση της δομής, είναι απαραίτητη η ανάλυση του προβλήματος στα επιμέρους στοιχεία του.

Στάδιο Β. Ανάλυση προβλήματος

Είναι το στάδιο στο οποίο το αρχικό πρόβλημα, διασπάται σε άλλα επί μέρους απλούστερα προβλήματα, διαδικασία με την οποία, ελαττώνεται η δυσκολία αντιμετώπισης των προβλημάτων, όσο περισσότερο προχωράει βαθύτερα η ανάλυσή τους σε απλούστερα προβλήματα.

Στάδιο Γ. Σχεδιασμός και Υλοποίηση

Πρόκειται για το στάδιο που υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων. Στο στάδιο αυτό αρχικά σχεδιάζονται οι διαφορετικές δυνατές λύσεις και τέλος επιλέγεται και υλοποιείται η πλέον κατάλληλη.

Για τη λύση ενός προβλήματος μπορεί να διατυπωθούν διαφορετικοί αλγόριθμοι, με αποτέλεσμα να τίθεται το θέμα της αξιολόγησής τους, δηλαδή ποιος από αυτούς είναι ο βέλτιστος. Η *αξιολόγηση αλγορίθμου* είναι συχνά ένα σύνθετο θέμα και η ερώτηση «υπάρχει αποδοτικότερος αλγόριθμος που λύνει το ίδιο πρόβλημα;» μελετάται από τη Θεωρητική Πληροφορική (Θεωρία Υπολογιστικής Πολυπλοκότητας). Η *αποδοτικότητα αλγορίθμου* αναφέρεται κυρίως στην ταχύτητα εκτέλεσής του, δηλαδή πόσο χρόνο χρειάζεται για να λύσει το πρόβλημα, αλλά και στο χώρο (ποσότητα) της κύριας μνήμης που χρησιμοποιεί. Έτσι, για παράδειγμα, αν για δύο αλγόριθμους Α και Β, που λύνουν το ίδιο πρόβλημα, συγκρίνονται με τα ίδια δεδομένα και συνθήκες (γλώσσα προγραμματισμού, υπολογιστικό σύστημα, σκληρός δίσκος κ.λπ.), διαπιστωθεί ότι ο αλγόριθμος Α δίνει λύση σε μικρότερο χρόνο από το χρόνο του Β, τότε ο Α χαρακτηρίζεται ως αποδοτικότερος του Β. Παρόμοια, αν ο αλγόριθμος Α χρησιμοποιεί λιγότερη μνήμη από τον Β, χαρακτηρίζεται αποδοτικότερος του Β. Υποθέτουμε ότι και στις δύο περιπτώσεις ο άλλος παράγοντας αντίστοιχα, είναι ίδιος.

1.3 Ανάλυση προβλήματος σε απλούστερα υποπροβλήματα

Υπάρχουν διάφορες επιστημονικές μέθοδοι του τρόπου εργασίας για την ανάλυση της δομής ενός προβλήματος και την εύρεση των τμημάτων-υποπροβλημάτων που το αποτελούν. Οι μέθοδοι αυτές είναι γνωστές ως: *αναλυτική μέθοδος* (από πάνω προς τα κάτω - Top Down), *συνθετική* (από κάτω προς τα πάνω - Bottom Up) και *μικτή* μέθοδος.

προς τα κάτω – Top Down), συνθετική (από κάτω προς τα πάνω – Bottom Up) και *μικτή* μέθοδος.

Στο κεφάλαιο αυτό θα ασχοληθούμε με την αναλυτική μέθοδο επίλυσης προβλήματος (Top Down problem solving), η οποία είναι και η ευρύτερα εφαρμοζόμενη μέθοδος. Η γενική αρχή της αναλυτικής μεθόδου ορίζει ότι, για να λύσουμε κάποιο σύνθετο πρόβλημα, πρέπει να:

- Να καθορίσουμε τα υποπροβλήματα.
- Να επαναλάβουμε τη διαδικασία αυτή για κάθε ένα από τα υποπροβλήματα αυτά, σε όσο βάθος αυτό είναι αναγκαίο.
- Να προχωράμε στην άμεση επίλυσή τους, όταν φτάσουμε σε υποπροβλήματα που δεν απαιτούν επιπλέον διάσπαση.

Η λύση του προβλήματος επιτυγχάνεται με τη σύνθεση των λύσεων των επιμέρους προβλημάτων.

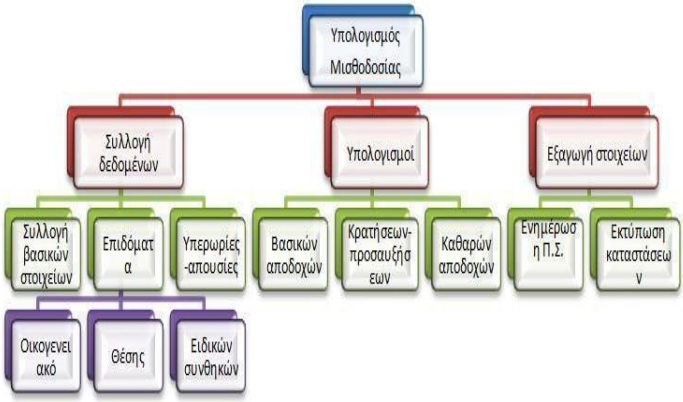
1.4 Γραφική απεικόνιση της δομής ενός προβλήματος

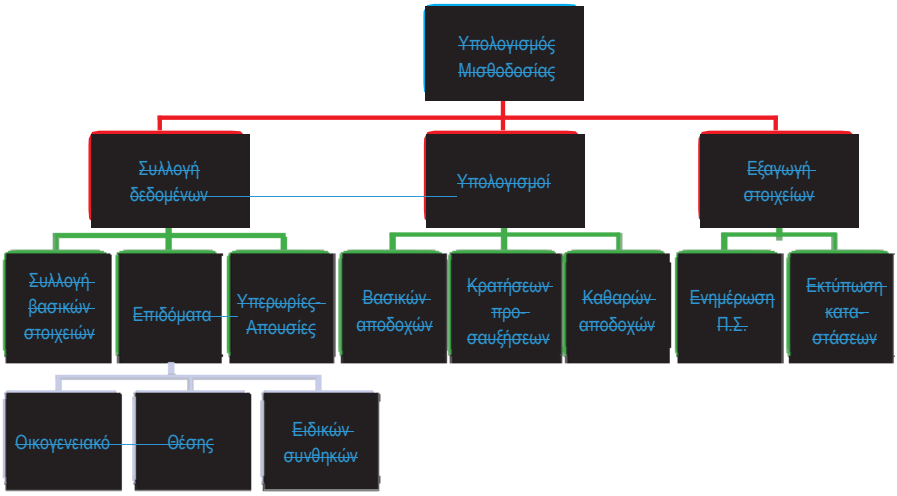
Η ανάλυση ενός προβλήματος σε άλλα απλούστερα, αναδύει παράλληλα και τη δομή του, για τη γραφική απεικόνιση της οποίας χρησιμοποιείται συχνά μια *διαγραμματική αναπαράσταση*, σύμφωνα με την οποία:

- Το αρχικό πρόβλημα αναπαρίσταται από ένα ορθογώνιο παραλληλόγραμμο.
- Κάθε ένα από τα απλούστερα προβλήματα στα οποία αναλύεται αναπαρίσταται, επίσης, από ένα ορθογώνιο παραλληλόγραμμο.
- Τα παραλληλόγραμμα που αντιστοιχούν στα απλούστερα προβλήματα σχηματίζονται ένα επίπεδο χαμηλότερα. Έτσι σε κάθε κατώτερο επίπεδο, δημιουργείται η γραφική αναπαράσταση των προβλημάτων στα οποία αναλύονται τα προβλήματα του αμέσως υψηλότερου επιπέδου.

Η διαγραμματική αναπαράσταση δίνει μια απτή απεικόνιση της δομής του προβλήματος και βοηθάει τόσο στην καλύτερη κατανόηση του ίδιου όσο και στη σχεδίαση της λύσης του.

Στην εικόνα 1.1 φαίνεται η διαγραμματική αναπαράσταση παραδε- ίγματος μισθοδοσίας~~παραδείγματος μισθοδο- σίας. Η ανάλυση του προβλήματος βρίσκεται στο Τετράδιο Εργασιών Μαθητή~~.





Εικόνα 1-1. Διαγραμματική αναπαράσταση προβλήματος *"Υ- πολογισμός Μισθοδοσίας"*
~~"Υπολογισμός Μισθοδοσίας"~~

1.5 Αναπαράσταση αλγορίθμων

Στη βιβλιογραφία συναντώνται διάφοροι τρόποι αναπαράστασης ενός αλγορίθμου, όπως με:

- *Φυσική*
  - Με *φυσική γλώσσα* (natural language), η οποία αποτελεί τον πιο απλό, ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης ενός αλγορίθμου, όπου με απλά λόγια και ελεύθερες εκφράσεις περιγράφουμε τα βήματα του αλγορίθμου. Αυτός ο τρόπος έκφρασης κρύβει αυξημένη πιθανότητα λάθους, λόγω της ασάφειας στην περιγραφή.
  - *Διαγραμματικές* *τεχνικές* (diagramming techniques), που συνιστούν ένα γραφικό τρόπο παρουσίασης του αλγορίθμου. Η πιο γνωστή τεχνική είναι το διάγραμμα ροής (flow chart).
- *Κωδικοποίηση*

- ~~Με *κωδικοποίηση*~~ (coding), δηλαδή με ένα πρόγραμμα γραμμένο είτε σε *ψευδογλώσσα* ~~*ψευδογλώσσα*~~ είτε σε κάποια γλώσσα *προγραμματισμού* που, όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο. *Ψευδογλώσσα ή Ψευδοκώδικας*~~*Ψευδοκώδικας*~~ είναι μια υποθετική δομημένη γλώσσα με στοιχεία από υπαρκτές γλώσσες~~γλώσσες~~ προγραμματισμού, με λίγες εντολές και απλοποιημένη σύνταξη, χρήσιμη ιδιαίτερα στα πρώτα στάδια εκμάθησης του προγραμματισμού.

Στο βιβλίο αυτό θα χρησιμοποιηθεί για την αναπαράσταση~~υλοποίηση~~ των αλγορίθμων η γλώσσα προγραμματισμού Python, έκδοση 2~~ενώ για την αναπαράστασή τους, το διάγραμμα ροής και η ψευδογλώσσα, όπως αυτά χρησιμοποιούνται και στο Βιβλίο *Εισαγωγή στις Αρχές της Επιστήμης των Υπολογιστών* της Β΄ τάξης ΕΠΑ.Λ~~.

2. Ανάπτυξη προγράμματος

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται, σε ένα πρώτο επίπεδο, θέματαθέματα σχετικά με τον κύκλο ανάπτυξης ενός προγράμματος και τις μεθοδολογίες που χρησιμοποιούνται. Αναλύονται θέματα ειδών προγραμματισμού και αναφέρονται αρχικές έννοιες για τον αντικειμενοστρεφήαντικειμενοστρεφή προγραμματισμό, καθώς και τεχνικές σχεδίασης και περιγραφήςπερι- γραφής συστημάτων.

Λέξεις κλειδιά

Κύκλος ανάπτυξης προγράμματος, μοντέλα ανάπτυξης και σχεδίασηςσχεδίασης λογισμικού, προγραμματιστικά υποδείγματα, αντικειμενοσ- τρεφήςαντικειμενοστρεφής / δομημένος προγραμματισμόςπρογραμματι- σμός.

Διδακτικές Ενότητες

2.1 Κύκλος ανάπτυξης προγράμματος/λογισμικού

Η διαδικασία ανάπτυξης λογισμικού εξελίσσεται σε διακριτές φάσεις φάσεις ή στάδια. Θεωρείται Θε- ωρείται υποσύνολο του κύκλου ζωής ενός συστή- ματοςσυστήματος λογισμικού που ξεκινά από την ανάλυση απαιτήσεων και τελειώνει με την παύση λειτουργίας του.

Αρκετές φορές οι θεωρητικές προσεγγίσεις για την ανάπτυξη λογισμικούλογισμικού έχουν δεχθείδε χθεί κριτική για την αποτελεσματικότητά τους, καθώς συχνά, στη πράξη, παρουσιάζονταιπαρουσι άζονται σημαντικές αποκλίσεις από τον αρχικό σχεδιασμό. Παρόλα αυτά είναι ιδιαίτεραιδιαί τερα χρήσιμο, η ανάπτυξη ενός λογισμικού να ακολουθεί συγκεκριμένα μοντέλα, τα οποία έχουν προταθεί ανάλογα με τα προτερήματα και τις αδυναμίεςαδυναμίες τους.

Μεταξύ των βασικών μεθοδολογιών (μοντέλων) που έχουν προταθείπροταθεί και ακολουθούνταιακολου θούνται, θα αναφερθούμε στο μοντέλο του *καταρ- ράκτηκαταρράκτη* (waterfall model) και της *σπειροειδούς προσέγγισης* (spiral model).

2.1.1 Μοντέλο του καταρράκτη

Πρόκειται για το μοντέλο που υποδιαιρεί τη διαδικασία ανάπτυξης ενός συστήματος λογισμικού στις ακόλουθες φάσεις:

- *Περιεχόμενα*

- Καθορισμού απαιτήσεων.
- Ανάλυσης απαιτήσεων.
- Σχεδίασης.
- Υλοποίησης.
- Ολοκλήρωσης. Λειτουργίας και συντήρησης.

Οι φάσεις εφαρμόζονται διαδοχικά με τη σειρά (γραμμικά) με κύκ- λους ανατροφοδότησης~~κύκλους ανατροφο- δότησης~~.

Εικόνα 2-1. Διαγραμματική αναπαράσταση του μοντέλου του καταρράκτη

2.1.2 Μοντέλο σπείρας

Στο μοντέλο της *σπείρας*, η ανάπτυξη ακολουθεί μια εξελικτική διαδικασία με την επαναληπτική εκτέλεση ενός κύκλου φάσεων, που σε καθεμιά δημιουργείται μια ενδιάμεση έκδοση του τελικού προϊόντος, η οποία βελτιώνεται κατά τον επόμενο κύκλο κ.ο.κ. Η διαδικασία αυτή συνεχίζεται μέχρι να παραχθεί μια έκδοση που να ικανοποιεί τις απαιτήσεις των χρηστών. Το πλεονέκτημα αυτού του μοντέλου είναι η γρήγορη ανάπτυξη ενός πρωτότυπου και η ανάλυση όλων των κινδύνων, πριν από κάθε φάση. Το μοντέλο αυτό χρησιμοποιείται περισσότερο σε έργα μεγάλης κλίμακας.

Κατά την ανάπτυξη ενός προγράμματος, εργαζόμαστε παρόμοια με τη διαδικασία ανάπτυξης ενός λογισμικού και ακολουθείται η μεθοδολογία υλοποίησης του κύ-

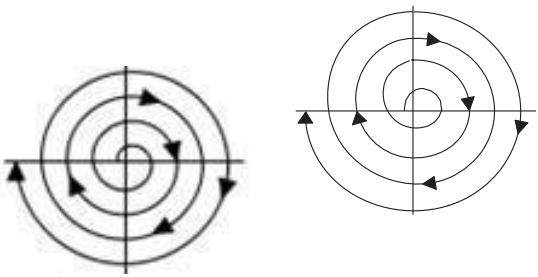
Περιεχόμενα
μεθοδολογία υλοποίησης του **κύκλου**

~~20~~

κλου **ανάπτυξης προγράμμα- τοςπρογράμματος** (program development life cycle-PDLC).

Μια τέτοια μεθοδολογία ανάπτυξης προγράμματος μπορεί να α-ποτελείταιαποτελείται από φάσειςφά- σεις, όπως:

- Ανάλυση του προβλήματος.
- Σχεδίαση.
- Συγγραφή κώδικα.
- Έλεγχος και εκσφαλμάτωση.
- Τεκμηρίωση.



Εικόνα 2-2. Διαγραμματική αναπαράσταση του μοντέλου της Σπείρας

2.1.3 Η Ή λογική συγγραφής προγράμματος ανάλογα με το είδος προγραμματισμούπρογραμ- ματισμού

Από τη δεκαετία του 1960 και μέχρι σήμερα, έχουν αναπτυχθεί διάφορα είδη προγραμματισμούπρο- γραμματισμού που τα υποστήριξαν πολλές γλώσσες προγραμματισμού. Μπορούμε να τα κατηγοριοποιήσου- μεκατηγοριοποιήσουμε σε μεγάλες κατευθύνσεις, τα λεγόμενα **προγραμματιστικά***προγραμματιστι- κά υποδείγματα* (programming paradigms).

Τα βασικά προγραμματιστικά υποδείγματα είναι τα ακόλουθα:

- Ο Προστακτικός προγραμματισμός.
- Ο Δηλωτικός προγραμματισμός.

2.1.4 Προστακτικός προγραμματισμός

Ο **Προστακτικός προγραμματισμός** (Imperative programming) βασίζεται σε εντολέςεντο- λές που υλοποιούν τα βήματα ενός αλγόριθμου, και βρίσκεται πιο κοντά στη λογική λειτουργίας του υπολογιστή. Γλώσσες που ακολούθησαν το είδος αυτό είναι οι κλασικές γλώσ- σεςκλα- σικές γλώσσες προγραμματισμού, όπως Cobol, Fortran, Pascal, C κ.ά.

Δομημένος και μη προγραμματισμός

Κατά την αρχική περίοδο του προγραμματισμού, η διακλάδωση και η αλλαγή της ροής εκτέλεσης ενός προγράμματος γίνονταν με τη χρήση της εντολής goto. Με την εντολή αυτή, η ροή εκτέλεσης μεταφερόταν σε οποιοδήποτε σημείο του προγράμματοςπρο- γράμματος, γεγονός που οδηγούσε σε μη δομημένα προγράμματα (μη δομημένος προγραμματισμός - Unstructuredunstructured programming). Στη συνέχεια δη- μιουργήθηκεδημιουργήθηκε το πρότυπο του **Δομημένου προγραμματισμού** (Structured programming), όπου η αλλαγή της ροής εκτέλεσης γίνεται με συγκεκριμένους κανόνες.

Ο **Διαδικαστικός προγραμματισμός** (Procedural programming) αποτελεί μια υποκατηγορία του δομημένουΔομημένου προγραμματισμού, με το πρόγραμμα να αποτελείται από ομάδες εντολών, τις *διαδικασί-εςδιαδικασίες* (procedures). Η γλώσσα προγραμματισμού που αξιοποίησε, ήδη από το 1970, το είδος αυτό είναι η Pascal.

Ο **Αντικειμενοστρεφής προγραμματισμός** (Object-oriented programming) βασίζεταιβα- σίζεται -σε αντίθεση με το διαδικαστικό προγ-ραμματισμόΔιαδικαστικό προγραμματισμό- σε αντικείμενα που αλληλεπιδρούναλ-ληλεπιδρούν μεταξύ τους. Ο Αντικειμενοστεφής προγραμματισμός αποτελεί ένα πρότυπο που ταιριάζει περισσότερο στη λογική οργάνωσης και λειτουργίας του πραγματικού κόσμου.

2.1.5 Δηλωτικός προγραμματισμός

Ο **Δηλωτικός προγραμματισμός** (Declarative programming) βα- σίζεταιβασίζεται στην περιγραφήπε- ριγραφή του σκοπού, ο οποίος ζητείται να επιτύχει το πρόγραμμα. Στο γενικό αυτό υπόδειγμα ανήκουν διάφορες υ-ποκατηγορίεςυποκατηγορίες προγραμματισμού, όπως είναι ο ΣυναρτησιακόςΣυ-ναρτησιακός και ο Λογικός.

Ο **Συναρτησιακός προγραμματισμός** (Functionalfunctional programming) βασίζεται σε μαθηματικέςμα- θηματικές συναρτήσεις, με γλώσσες όπως Lisp, Logo κ.ά.

Στο **Λογικό προγραμματισμό** (Logiclogic programming), ένα πρόγ-ραμμαπρόγραμμα είναι ένα σύνολοσύ- νολο από αξιώματα ή κανόνες οι οποίοι καθο- ρίζουνκαθορίζουν σχέσεις ανάμεσα σε αντικείμενα.αντικείμε- να. Υπολογισμός ενός λογι- κούλογικού προγράμματος είναι ένα συμπέρασμα που συνάγεταισυνάγε- ται από τα αποτελέσματά του, όπως για παράδειγμα γίνεται στην Prolog.

Περιεχόμενα

Στο προγραμματιστικό πρότυπο του δηλωτικού προγραμματισμού μπορούμε να θεωρήσουμε ότι ανήκουν και άλλες γλώσσες που δεν υπάγονται στις δύο προηγούμενες κατηγορίες. Χαρακτηριστικές είναι η *HTML* (HyperText Markup Language), γλώσσα σήμανσης χαρακτηρισμού υπερκειμένου και εν μέρει η *SQL* (Structured Query Language) γλώσσα για τη διαχείριση δεδομένων, σε ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (RDBMS-Relational Database Management System).

2.1.6 Λοιπά πρότυπα και τεχνικές προγραμματισμού

Εκτός από τα παραπάνω υποδείγματα προγραμματισμού υπάρχουν και άλλα τα οποία, είτε δεν μπορούν να χαρακτηριστούν πλήρως ως προγραμματιστικά υποδείγματα, είτε αποτελούν *τεχνικές* και *μεθοδολογίες προγραμματισμού*, που θα αναφέρουμε στη συνέχεια.

Παράλληλος προγραμματισμός (Parallel programming). Επιτρέπει ταυτόχρονη εκτέλεση διαδικασιών από διαφορετικούς επεξεργαστές, όπως στην γλώσσα προγραμματισμού Occam.

Προγραμματισμός οδηγούμενος από γεγονότα (Event-driven programming). Αποτελεί περισσότερο *τεχνική αρχιτεκτονικής ενός προγράμματος* σχετικά με τη ροή του, παρά προγραμματιστικό υπόδειγμα. Η ροή του προγράμματος εξαρτάται από την ύπαρξη *γεγονότων* (events), όπως είναι, για παράδειγμα, ένα μήνυμα ενός αισθητήρα ή μια ενέργεια του χρήστη με το πάτημα του ποντικιού ή ενός πλήκτρου. Παραδείγματα γλωσσών που το ενσωματώνουν είναι η Python, Java κ.ά.

Οπτικός προγραμματισμός (Visual programming). Εκφράζει τη δυνατότητα γλωσσών να παρέχουν τη δυνατότητα δημιουργίας του προγράμματος μέσω γραφικών αντικειμένων, αντί της πληκτρολόγησης του κώδικα ο οποίος αντιστοιχεί σε εντολές. Για παράδειγμα, στην κατηγορία αυτή ανήκουν περιβάλλοντα δημιουργίας σεναρίων όπως το Kodu (Kodu, 2016) και

το Alice (Alice, 2016) ή το Scratch (Scratch, 2016).

Προγραμματισμός Δέσμης ενεργειών (Script programming). Είναι τύπος προγραμματισμούπρο- γραμματισμού και όχι υπόδειγμα, δημιουργίας μικρών τμημάτων κώδικα και όχι ολοκληρωμένων προγραμμάτων. Είναι υψηλού επιπέδου προγραμματισμός που

Είναι υψηλού επιπέδου προγραμματισμός που

διερμηνεύεται κατά την εκτέλεση από ένα άλλο πρόγραμμα, όπως ένας φυλλομετρη- τής~~φυλλομε- τρητής~~.

Αρθρωτός ή Τμηματικός Προγραμματισμός (Modular programming). Σχετίζεται περισσότερο με *τεχνική σχεδίασης* λο- γισμικού~~λογισμικού~~ παρά με πρότυπο. Χαρακτηρίζεται από τη διαίρεση του προβλήματος σε απλούστερα τμήματα, αυτά με τη σειρά τους σε επί μέρους μικρότερα κ.ο.κ. Παρέχει απλούστευση της επίλυσης ενός προβλήματος~~προβλή- ματος~~, ευκολία κωδικοποίησης και συντήρησης. Γενι- κά~~Γενικά~~ ως *τμήμα* θεωρούμε ένα σύνολο ενεργειών το οποίο εκτελεί μια καθορισμένη λειτουργία ενός προγράμματος και είναι κατά το δυ- νατόν~~δυνατόν~~ ανεξάρτητο από τα άλλα τμήματα.

Ιεραρχικός σχεδιασμός

Η μέθοδος ανάλυσης ενός προβλήματος σε μικρότερα, είναι εκείνη με την οποία αντιμετωπίζουμε το πρόβλημα ως μια πολυεπίπεδη δομή. Έτσι, για τη σχεδίασή του, ξεκινάμε από το υψηλότερο επί- πεδο~~επίπεδο~~ και στη συνέχεια το αναλύουμε σε όλο και χαμηλότερα, έως ότου φθάσουμε στο κατώτερο επίπεδο ανάλυσης. Η τεχνική αυτή ονομάζεται **ιεραρχικός σχεδιασμός** (Top~~τοp~~ down design).

2.1.7 Ενδεικτικά περιβάλλοντα και γλώσσες προγραμματισμού

Pascal. Η πλέον κλασική γλώσσα δομημένου προγραμματισμού. **Visual Basic** (Visual Basic, 2016). Περιβάλλον προγραμματισμού, που ακολουθεί μικτό πρότυπο υποδειγμάτων.

C++. Επέκταση της C. Αποτελεί γλώσσα αντικειμενοστρεφούς~~Αντικειμενοστρεφούς~~ προγραμματισμού, αν και μπορεί να χρησιμοποιηθεί και για δι- δασκαλία~~διδασκαλία~~ Διαδικαστικού προγραμματισμού~~προγραμματι- σμού~~.

Java (Java, 2016). Σύγχρονη γλώσσα Αντικειμενοστρεφούς προγραμματισμού.

Python (Python, 2016). Γλώσσα που ανήκει ουσιαστικά σε μικτά υποδείγματα προγραμματισμού~~προ- γραμματισμού~~, όπως αντικειμενοστρεφές, συ- ναρτησιακό~~συναρτησιακό~~ και διαδικαστικό.

2.2 Εισαγωγή στις

2.2 Εισαγωγή στις βασικές έννοιες του αντικειμενοστρεφούς ~~αντικειμενοστρε- φούς~~ προγραμματισμού

~~2.2 Εισαγωγή στις βασικές έννοιες του αντικειμενοστρε- φούς προγραμματισμού~~

Στην παράγραφο αυτή θα αναφερθούμε επαναληπτικά σε σχέση με την προηγούμενη~~προηγού- μενη~~ τάξη, στον αντικειμενοστρεφή προγραμματισμό~~Αντικειμενοστρεφή προγραμματισμό~~, ενώ πληρέστερη ανάλυση θα γίνει στο κεφάλαιο 11.

~~Εικόνα 2-3. Είδη προγραμματισμού~~

Η μέχρι τώρα διδασκαλία του προγραμματισμού ακολουθεί το υ- πόδειγμα~~υπόδειγμα~~ του δομημένου~~Δομη μένου~~ προγραμματισμού. Ο δομημένος προγ- ραμματισμός~~Δομημένος προγραμματισμός~~ έλυσε σημαντικά προβλήματα~~προ βλήματα του~~ προγραμματισμού, αλλά οδήγησε και στην ανάπτυξη πολύπλοκων προγραμμάτων. Η κατασκευή των προγραμμάτων αυτών και η συντήρησή τους απα- ιτούσε~~απαιτούσε~~ ειδικούς με βαθιές γνώσεις σε επιμέρους θέματα, όπως στη διαχείριση μνήμης, και στα λειτουργικά συστήματα. Η πολυπλο- κότητα~~πολυπλοκότητα~~ κατασκευής των προγραμμάτων~~προ γραμμάτων~~ δημιούργησε μια άλλη λογική προγραμματισμού που κινείται σε διαφορετική~~διαφο- ρετική~~ κατεύθυνση. Πρόκειται για τον Αντικειμενοστρεφή προγραμματισμό.

Βασική αρχή πάνω στην οποία δομήθηκε ο αντικειμενοστρεφής~~Αντικειμενοστρεφής~~ προγραμματισμός είναι~~εί- ναι~~ ότι ο πραγματικός κόσμος αποτελείται από αντικείμενα (objects). Τα αντικείμενα αυτά παίρνουν μορφή και η συμπεριφορά τους εξαρτάται από τα ερεθίσματα που

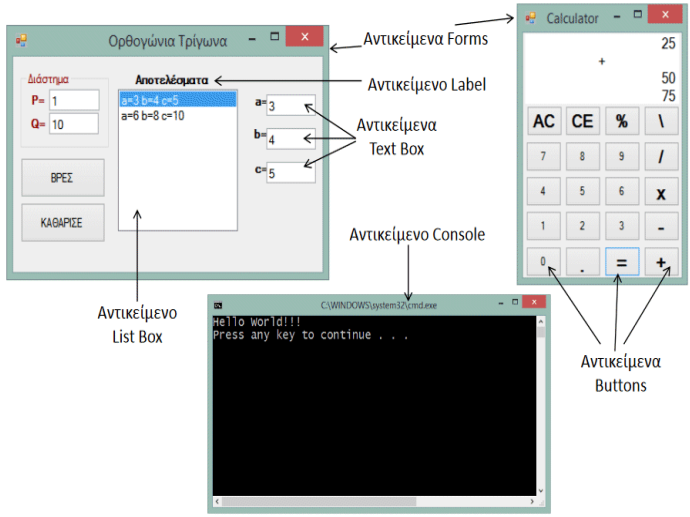
λαμβάνουν από το περιβάλλον στο οποίο βρίσκονται. Ως αντικείμενα μπορο- ύμε~~μπορούμε~~ να θεωρήσουμε σχεδόν τα πάντα, δηλαδή οτιδήποτε υπάρχει στον πραγματικό κόσμο, όπως π.χ. μαθητής, παγκάκι, πορτοκάλι, ραδιόφωνο, αυτοκίνητο κ.λπ~~λ.π~~. (εικόνα 2-34).

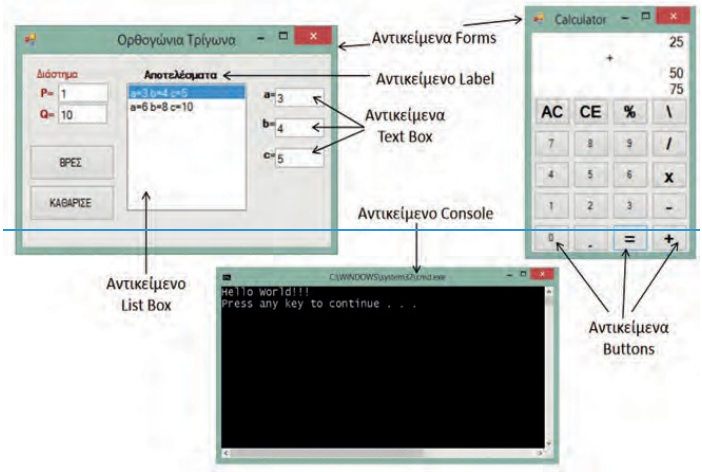
Εικόνα 2-34.
Φυσικά αντικείμενα

Περιεχόμενα

Δεν

~~Τα αντικείμενα δε~~ σημαίνει ότι τα αντικείμενα έχουν απαραίτητα ~~έχουν~~ υλική υπόστα- ση~~υπόσταση~~. Μπορούν να θεωρηθούν ως αντικείμενα έννοιες, όπως μια συγκεκριμένη μαθηματική θεωρία ή προγραμματιστικά αντικείμε- να~~αντικείμενα~~, όπως αυτά που εμφανίζει η εικόνα 2-45. Για παράδειγμα~~παρά- δειγμα~~, κάθε παράθυρο οθόνης είναι ένα αντικείμενο.





Εικόνα 2-45. Προγραμματιστικά αντικείμενα

Μέρος 1: Εμβάθυνση σε βασικές έννοιες

Περιεχόμενα

Κάθε αντικείμενο είναι ξεχωριστό και έχει ένα σύνολο από χαρακτηριστικά. Για παράδειγμα: το παγκάκι είναι ξύλινο, η μπάλα είναι στρογγυλή, ο μαθητής φοιτά σε μια τάξη κ.λπ. Το σύνολο αυτών των χαρακτηριστικών αποτελούν τις *ιδιότητες* του αντικειμένου.

Ανάλογα με την εφαρμογή που αναπτύσσουμε, καθορίζουμε τόσο τα αντικείμενα όσο και τις ιδιότητες που μας ενδιαφέρουν. Ένα αντικείμενο συνήθως μπορεί να εκτελέσει κάποια λειτουργία. Για παράδειγμα: ο μαθητής μπορεί να πάρει μια εργασία, να προβιβαστεί στην επόμενη τάξη, η μπάλα να αναπηδήσει, *ένα* παράθυρο οθόνης να κλείσει κ.λπ. Τέτοια αποτελέσματα αποτελούν τη λεγόμενη *συμπεριφορά* του αντικειμένου.

Η συμπεριφορά του αντικειμένου ενεργοποιείται από ένα *γεγονός*. Για παράδειγμα: ρίχνουμε τη μπάλα στο πάτωμα ή κάνουμε κλικ σε ένα σημείο του παραθύρου.

Το πώς ακριβώς θα συμπεριφερθεί ένα προγραμματιστικό αντικείμενο εξαρτάται από ένα τμήμα κώδικα που ανήκει στο αντικείμενο και ονομάζεται *κώδικας-χειριστής*.

Όπως ήδη αναφέραμε ένα αντικείμενο είναι ξεχωριστό, έχει, δηλαδή, τη δική του ταυτότητα. Αυτό πρακτικά σημαίνει ότι, ακόμη και αν δύο αντικείμενα έχουν ακριβώς τις ίδιες ιδιότητες, εξακολουθούν να παραμένουν δύο ανεξάρτητα αντικείμενα. Για παράδειγμα: δύο μπάλες ποδοσφαίρου έχουν τα ίδια χαρακτηριστικά ανεξάρτητα σε ποιο παιχνίδι χρησιμοποιείται η κάθε μία. Για λόγους απλότητας και "οικονομίας" θα επιθυμούσαμε σε παρόμοια αντικείμενα να καταγράφονται τα χαρακτηριστικά τους μόνο μια φορά. Έτσι οδηγούμαστε στις *κλάσεις* που είναι, ουσιαστικά, "καλούπια" για τη δημιουργία παρόμοιων αντικειμένων. Με την ίδια κλάση, μπορούν να δημιουργηθούν πολλά διαφορετικά αντικείμενα. Κάθε παράθυρο είναι ένα αντικείμενο που ανήκει στην κλάση παραθύρων, αλλά και κάθε ένα από τα επιπλέον στοιχεία που περιέχει όπως ετικέτες, κουμπιά, μπάρες κύλισης κ.λπ., είναι επίσης αντικείμενα τα οποία ανήκουν σε μια πιο γενική κατηγορία ή κλάση.

Περιεχόμενα

Το αντικείμενο έχει τιμές στις ιδιότητες, ενώ κληρονομεί τις λειτο- υργίες~~λειτουργίες~~ της κλάσης από την οποία προέρχεται.

Όταν κάνουμε μια ενέργεια (γεγονός), όπως, για παράδειγμα αρισ- τερό~~αριστερό~~ κλικ με το ποντίκι σε ένα από τα στοιχεία αυτά, ενημερώνον- ται~~ενημερώνονται~~ όλα για την ενέργεια, αλλά αντιδρά μόνο το ενεργό. Έτσι για παράδειγμα, αν έχουμε πολλά όμοια παράθυρα του φυλλομετρη- τή~~φυλλομετρητή~~ το ένα πίσω από το άλλο και πατήσουμε το κουμπί κλεισίματος (γεγονός), θα κλείσει μόνο το ενεργό παράθυρο (συμπεριφορά).

Η όλη παραπάνω λειτουργία των παραθύρων στηρίζεται σε μια εφαρμογή που λέγεται~~λέ- γεται~~ *γραφική διεπαφή χρήστη* (graphical user interface-GUI) και η οποία αποτελεί ένα χαρακτηριστικό παράδε- ιγμα αντικειμενοστρεφούς~~παράδειγμα Αντικειμενοστρεφούς~~ προγραμματισμού.

2.2.1 Σύγκριση διαδικαστικού ~~Διαδικαστικού~~ και αντικειμενοστρεφούς~~Αντικειμενοστρεφούς~~ προγραμματισμού

Οι γλώσσες αντικειμενοστρεφούς~~Αντικειμενοστρεφούς~~ προγραμματισμού επιτρέπουν ένα υψηλότερο επίπεδο *αφαίρεσης*, όρος που θίγεται στο κεφάλα- ιο~~κεφάλαιο~~ 12, από αυτές του Διαδικαστικού~~Διαδικαστι- κού~~ προγραμματισμού, για την ε- πίλυση~~επίλυση~~ των προβλημάτων της πραγματικής ζωής. Στη *δομημένη σχεδίαση* σκεφτόμαστε με τη λογική της δομής και λειτουργίας του υπολογιστή (διαδοχή, απόφαση, αποθήκευση - μνήμη κ.λπ.),~~κλπ),~~ ενώ στην *αντικειμενοστρεφή~~αντικειμενο στρεφή~~ σχεδίαση* σκεπτόμαστε με κέντρο το πρόβλημα, χρησιμοποιώντας τα αντικείμενα~~αντι κείμενα~~ του λογισμικού για την αναπαράσταση αφηρημένων οντοτήτων του προβλήματος~~προ- βλήματος~~ προς την επίλυσή του.

2.2.2 Αντικειμενοστρεφής σχεδίαση

Μεταξύ των μεθοδολογιών ανάλυσης και σχεδίασης πληροφορια- κών~~πληροφοριακών~~ συστημάτων υπάρχουν δύο που μας ενδιαφέρουν περισσό- τερο~~περισσότερο~~ στο κεφάλαιο αυτό.

Η πρώτη, σχετίζεται με το δομημένο~~Δομημένο~~ και η δεύτερη με τον αντικει- μενοστρεφή προγραμματισμό.~~Αντικειμενοστρεφή προ γραμματισμό.~~ Έτσι, όπως ο δομημένος προγ- ραμματισμός ~~Δομημένος προγραμματισμός~~ οδήγησε στην *δομημένη μεθοδολογία* σχεδίασης (structured design), ο αντικειμενοστρεφής προγραμματισμός οδή- γησε~~Αντικειμενοστρεφής προγραμματι- σμός οδήγησε~~ στην *αντικειμενοστρεφή μεθοδολογία σχεδίασης* πληροφορι- ακών~~πληροφοριακών~~ συστημάτων (object-oriented analysis and design).

Περιεχόμενα

Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language-UML)

Για τη γραφική απεικόνιση, τον προσδιορισμό, την τεκμηρίωση κ.ά. των στοιχείων και των φάσεων ενός συστήματος λογισμικού χρησιμοποιείται ευρέως η γλώσσα μοντελοποίησης UML. Αποτε- λείταιΑποτελείται από ένα σύνολο προσυμφωνημένων όρων, συμβόλων και διαγραμμάτων και κυριαρχεί στη γραφική απεικόνιση της σχεδία- σηςσχεδίασης ενός αντικειμενοστρεφούς συστήματος.

Στην έκδοση 2 διαθέτει πολλούς τύπους διαγραμμάτων, κυρίως δύο κατηγοριών: τους τύπους π ουπου σχετίζονται με την αναπαράσ- τασηαναπαράσταση της δομής (structure) του συστήματοςσυ- στήματος και τους τύπους που σχετίζονται με την αναπαράσταση της συμπεριφοράςσυμπεριφο- ράς (behavior) των δομικών στοιχείων του.

3. Βασικά στοιχεία γλώσσας προγραμματισμού

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται τα βασικά χαρακτηριστικά του ολοκληρωμένου περιβάλλοντος ανάπτυξης της γλώσσας προγραμματισμού~~προγραμματισμού~~ Python. Επίσης αναπτύσσονται~~αναπτύσσονται~~ οι βασικοί τύποι με- ταβλητών~~μεταβλητών~~, οι βασικές εντολές, οι δομές επιλογής και επανάληψης και οι συναρτήσεις.

Λέξεις κλειδιά

Προγραμματιστικό περιβάλλον ανάπτυξης, λογικές και αριθμητικές εκφράσεις, τελεστές~~τελε- στές~~, δομές και τύποι δεδομένων, μεταβλητή, εν- τολές~~εντολές~~, συναρτήσεις, διερμηνευτής.

Διδακτικές Ενότητες

Γνωριμία με το ολοκληρωμένο περιβάλλον ανάπτυξης της γλώσσας προγραμματισμού Python

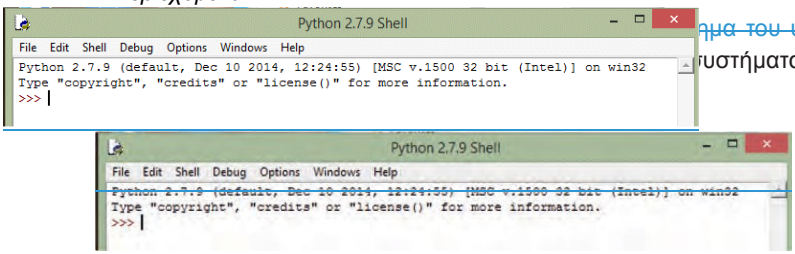
Η γλώσσα προγραμματισμού Python είναι μια σύγχρονη γλώσσα προγραμματισμού~~προγραμματι- σμού~~ που υποστηρίζει τόσο διάφορα προγραμματισ- τικά~~προγραμματιστικά~~ υποδείγματα -όπως δομημένο~~δο- μημένο~~, συναρτησιακό, αντικειμενοσ- τρεφές~~αντικειμενοστρεφές~~ όσο και τεχνικές προγραμματισμού. Επιπρόσθετα περιέχει πλούσιες βιβλιοθήκες από έτοιμο κώδικα προγραμματισμού και υποστηρίζει τον *αρθρωτό προγραμματισμό*. Τις

βιβλιοθήκες, αυτές (μονάδες λογισμικούλο- γισμικού), αλλά και άλλες επιπρόσθετες εξωτερικές βιβλιοθήκες με έτοιμο κώδικα, μπορούμε να τις χρησιμοποιούμε στα προγράμματά μας ακολουθώντας ένα σύνολοσύνο- λο απλών κανό- νωνκανόνων και εντολών. Ο τρόπος αυτός μας επιτρέπει τη σχετικά εύκο- ληεύκολη συγγραφή πιο σύνθετων προγραμμάτων. Μονάδες λογισμικού - βιβλιοθήκες μπορείμπο- ρεί να δημιουργήσει και ένας προγραμματιστής, ώστε να τις χρησιμοποιεί κατάλληλακατάλλη- λα σε διαφορετικά προγράμματά του, αποφεύγοντας να γράφει, κάθε φορά, κώδικα από την αρχή.

Το **Ολοκληρωμένο Περιβάλλον Ανάπτυξης Προγραμμάτων IDLE** (Integrated DeveLopment Environment) της Python, είναι ένα *Ελεύθερο Λογισμικό/ Λογισμικό Ανοικτού Κώδικα* (ΕΛ/ΛΑΚ), που, εύκολα, μπορούμε να το εγκαταστήσουμε στον υπολογιστή μας. Αρκεί να κατεβάσουμε από το Διαδίκτυο -και στη συνέχεια να εκτελέσουμε- το κατάλληλο αρχείο, ανάλογα με το λειτουργικό σύστημα του υπολογιστή μας, εφ' όσον είναι διαθέσιμο για διάφοεκτε-

ρα *Περιεχόμενα*

30



Εικόνα 3-1. Το περιβάλλον IDLE της Python

3.1 Μεταβλητές και τύποι δεδομένων

3.1.1 Τύποι δεδομένων

Ένα πρόγραμμα συνήθως επεξεργάζεται *δεδομένα* τα οποία μπορεί να είναι αποθηκευμένα στην κύρια μνήμη του υπολογιστή, σε αποθηκευτικό μέσο ή η εισαγωγή τους να γίνεται από κάποια μονάδα. Οι *τύποι δεδομένων* προσδιορίζουν τον τρόπο παράστασης των δεδομένων εσωτερικά στον υπολογιστή, καθώς και το είδος της επεξεργασίας τους από αυτόν. Στην Python δε δηλώνουμε ποιο τύπο δεδομένων χρησιμοποιούμε.

Οι χαρακτηριστικοί τύποι δεδομένων στην Python είναι ο αριθμητικός, ο λογικός
(boolean) και οι συμβολοσειρές ή αλφαριθμητικά (strings).

Οι **αριθμοί** στην Python είναι κυρίως τριών τύπων:

- ακέραιοι (Integer)
- αριθμοί κινητής υποδιαστολής (floating point)
- μιγαδικοί αριθμοί (complex numbers), τύπος που δε θα μας απασχολήσει στη συνέχεια.

Παραδείγματα

Ο 19, αποτελεί παράδειγμα ακέραιου.

Οι 256.14 και 28.2E-5, όπου το σύμβολο E δηλώνει δύναμη του 10, είναι παραδείγματα αριθμών κινητής υποδιαστολής (ή *floats* για συντομία). Σε αυτή την περίπτωση, το 28.2E-5 σημαίνει $28.2 * 10^{-5}$. Παρατηρούμε ότι το δεκαδικό τμήμα διαχωρίζεται με το χαρακτήρα τελεία “.” και όχι το κόμμα “,”.

Ο **λογικός τύπος** (boolean) που δέχεται μόνο δύο τιμές, την τιμή *True* (Αληθής) και την τιμή *False* (Ψευδής) και έχει σκοπό την κα- ταγραφήκαταγραφή του αποτελέσματος ενός ελέγχου.

Οι **συμβολοσειρές** είναι μια ακολουθία από χαρακτήρες. Μια συμβολοσειρά μπορείμπο- ρεί να αποτελείται από περισσότερες από μία λέξεις. Οι λέξεις μπορούν να είναι σε κάθε γλώσσα που υποστηρί- ζεταιυποστηρίζεται από το πρότυπο Unicode, άρα σε ελληνική, αγγλικήαγγλι- κή κ.ο.κ. Μπορούμε να ορίσουμε μια συμβολοσειρά με μονά εισαγωγικά ή με διπλά, αλλά όχι ανάμικτα. Με ότι ξεκινάμε, θα πρέπει πάλι να κλείνουμε.

Παράδειγμα: "“221051445"” ή ‘Καλημέρα’.

Για να ελέγξουμε τον **τύπο δεδομένων** χρησιμοποιούμε την εντολή **type ()**.

Παράδειγμα

```
>>> type( 1 )
<type ‘int’>
>>> type( 3.14 )
<type ‘float’>
>>> a = True
>>> type(False )
<type ‘bool’>
>>> type( ‘Αρετή’ )
<type ‘str’>
```

```
>>> type (1)
<type 'int'>
>>> type (3.14)
<type 'float'>
>>> type (False)
<type 'bool'>
>>> type ('Αρετή')
<type 'str'>
>>>
```

3.2 Αριθμητικές και λογικές πράξεις και εκφράσεις

Χρησιμοποιώντας τιμές κάθε τύπου δεδομένων, μπορούμε να κά- νουμεκάνουμε διάφορες πράξεις, χρησιμοποιώντας τους αντίστοιχους τε- λεστέςτελεστές. Οι *τελεστές* (operators) είναι σύμβολα ή λέξεις για τη δημι- ουργίαδημιουργία αριθμητικών και λογικών εκφράσεων. Οι βασικότεροι τε- λεστέςβασι- κότεροι

τελεστές στη γλώσσα Python είναι:

Αριθμητικοί τελεστές: Είναι τα σύμβολα που χρησιμοποιούμε για να κάνουμε μαθηματικέςμα- θηματικές πράξεις. Στη γλώσσα Python χρησιμοποιούμεχρησιμοποιούμε τους παρακάτω βασικούς αριθμητικούς τελεστές:

Πρόσθεση	+	Αφαίρεση	-	Πολλαπλασιασμός	*
Διαίρεση	/	Ύψωση σε δύναμη	**		
Το υπόλοιπο της ακέραιας διαίρεσης	%				

Σε κάθε έκφραση στην οποία υπάρχουν αριθμητικοί τελεστές ακολουθείται μια προσδιορισμένη ιεραρχία πράξεων, που είναι:

1. Ύψωση σε δύναμη.
2. Πολλαπλασιασμός, διαίρεση, υπόλοιπο ακέραιας διαίρεσης.
3. Πρόσθεση, αφαίρεση.

Αν θέλουμε να αλλάξουμε την ιεραρχία των πράξεων, μπορούμε να χρησιμοποιήσουμε παρενθέσεις. Για παράδειγμα, στην έκφραση (2+3)*5 θα εκτελεστεί πρώτα η πρόσθεση μέσα στην παρένθεση και μετά, το αποτέλεσμα θα πολλαπλασιαστεί επί 5, σε αντίθεση με την έκφραση 2+3*5 στην οποία, πρώτα θα γίνει ο πολλαπλασιασμός και μετά η πρόσθεση.

Σχεσιακοί (ή συγκριτικοί) τελεστές: χρησιμοποιούνται για τη σύγκριση δύο τιμών ή μεταβλητών, με το αποτέλεσμα μιας σύγκρισης να είναι είτε True (Αληθής) είτε False (Ψευδής). Στη γλώσσα Python χρησιμοποιούνται οι παρακάτω βασικοί σχεσιακοί τελεστές:

Μικρότερο από	<
Μικρότερο ή ίσο από	<=
Μεγαλύτερο από	>
Μεγαλύτερο ή ίσο από	>=
Ίσο με	==
Διάφορο από	!=

Τελεστές λογικών πράξεων: Στις λογικές πράξεις και εκφράσεις χρησιμοποιούνται οι λογικοί τελεστές not (ΟΧΙ), and (ΚΑΙ), or (Ή) με τις ακόλουθες λογικές λειτουργίες:

•

- not (ΟΧΙ): πράξη άρνησης
- and (ΚΑΙ): πράξη σύζευξης
- or (Ή): πράξη διάζευξης.

Το αποτέλεσμα μιας λογικής πράξης είναι True (Αληθής) ή False (Ψευδής) σύμφωνασύμφω- να με τον παρακάτω πίνακα:

P	Q	P and Q	P or Q	Not P
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Η προτεραιότητα των λογικών τελεστών είναι not, and, or με αυτή τη σειρά.

Παραδείγματα

Πίνακας 3-1. Παραδείγματα αριθμητικών/λογικών εκφράσεωνΑριθμητικές και λογικές εκφράσεις

Αριθμητικές πράξεις	Αποτελέσματα
>>> 2+8*2	18
>>> 2**2+4/2-3*4	-6
>>> 45 / 10	4 # Στην Python, η διαίρεση ακεραίωνακ επιστρέ- φειεπιστρέφει το ακέραιοακέ- ραι
>>> 45 % 10	5 # μας επιστρέφει το υπόλοιπο της ακέρα
>>> 45.0 /10	4.5 # Η διαίρεση αριθμών κινη-κινητή επιστρέφει το πηλίκο, ως αριθμό κινητής υ

	τής υποδιαστολής, μας επιστρέ- φει το πηλίκο, ως αριθμό κινη- τής υποδιαστολής
Λογικές εκφράσεις με συγ- κριτικούς συγκριτικούς τελεστές	**Αποτελέσματα**
>>> 23 ==23	True
>>> 34 != 45	True
>>> 56 <= <=12	False
Λογικές εκφράσεις με τε- λεστές τελεστές λογικών	**Αποτελέσματα**
>>> (12<11) and (23>10)	False
>>> (12<11) or (23>10)	True
>>> not(56<=12)	True

Μεταβλητές

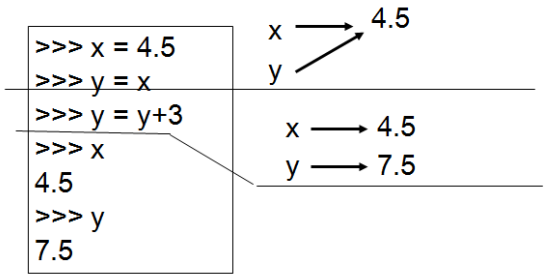
Η γλώσσα Python παρέχει εντυπωσιακές εναλλακτικές δυνατότη- τες δυνατότητες για τη διαχείριση διαχείρι- ση μεταβλητών που διευκολύνουν τον προγραμ- ματιστή προγραμματιστή. Για τη χρησιμοποίηση μιας μεταβλητής δεν απαιτείται η δήλωσή της, ενώ μπορεί να εκχωρήσουμε διαφορετικούς διαφορετι- κούς τύπους τιμών σε μια μεταβλητή κατά τη διάρκεια ενός προγράμματος, όπως ακέραιες τιμές, κινητής υποδιαστολής και συμβολοσειρές.

Για να χρησιμοποιήσουμε μια μεταβλητή απαιτείται να της δώσου- με δώσουμε ένα όνομα και στη συνέχεια να της εκχωρήσουμε κάποια τιμή. Στη γλώσσα Python υπάρχουν ορισμένοι ορι- σμένοι κανόνες που πρέπει να ακολουθούμε σχετικά με το όνομα μιας μεταβλητής μεταβλη- τής. Έτσι, για πα- ράδειγμα παράδειγμα, δεν επιτρέπεται να ξεκινά το όνομα μιας μεταβλητής με αριθμό και το όνομα που θα δώσουμε δεν πρέπει να είναι όμοιο με κάποιο όνομα ενσωματωμένης συνάρτησης ή εντολής. Συνηθί- ζουμε Συνηθίζουμε να δίνουμε ένα όνομα σχετικό

με το είδος της μεταβλητής με λατινικούς χαρακτήρες, που μπορεί να συνοδεύεται από κάποιον αριθμό, όπως για παράδειγμα: onoma1, onoma_2, timi, mesi_timi, embado κ.ά.

Για να *εκχωρήσουμε* μια τιμή σε μια μεταβλητή, χρησιμοποιούμε τον τελεστή ίσον " = ",“ = “, όπως για παράδειγμα a = 125.

Στο αριστερό μέρος δίνουμε το όνομα της μεταβλητής, στη συνέ- χεια χρησιμοποιούμεσυνέχεια χρησιμοποι- ούμε τον τελεστή εκχώρησης “=” και στο δεξιό μέρος βάζουμε την τιμή, μια άλλη μεταβλητή ή μια έκφραση που έχει ως αποτέλεσμα μια τιμή.



Προσοχή! ο τελεστής “=” δεν έχει τη σημασία που έχει το σύμβολο της ισότητας, όπως το χρησιμοποιούμε στα μαθηματικά. Στις πε- ρισσότερεςπερισσότερες γλώσσες προγραμματισμούπρογραμ- ματισμού, ο τελεστής εκχώρησης τιμής " = ""“ = “ σημαίνει ότι εκχωρούμε στη μεταβλητή μια τιμή ενός τύπου δεδομένου και ταυτόχρονα την εισάγουμε στη θέση μνήμης που αντιστοιχεί στη μεταβλητή με αυτό το όνομα. Για παράδειγμα, η εντολή a = 125 σημαίνει ότι η μεταβλητή με όνομα a αντιστοιχεί σε μια θέση μνήμης του υπολογιστή και η ακέραια τιμή 125 εισά- γεταιεισάγεται στη θέση αυτή. Αν στη συνέχεια θέσουμε a = 250, στη θέση μνήμης που αναφέρεται η μεταβλητή a εισάγεται η νέα ακέραια τιμή 250 και η παλαιά τιμή 125 παύει πλέον να υπάρχει, αφού έχει αντικατασταθεί από τη 250.

- *Περιεχόμενα*

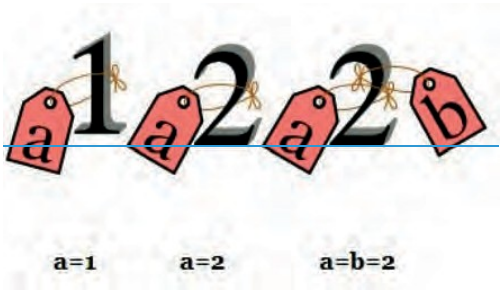
Όλα τα δεδομένα σε ένα πρόγραμμα Python αναπαρίστανται με αντικείμενα ή με σχέσεις μεταξύ των αντικειμένων, με κάθε αντικείμενοαντικείμενο να έχει μια ταυτότητα (identity), έναν τύπο και μία τιμή. ΓιαΓια παράδειγμα, το 12 είναι ένα αντικείμενο με τιμή

Περιεχόμενα

παράδειγμα, το 12 είναι ένα αντικείμενο με τιμή 12, , τύπου int (ακέραιος).

a=1 **a=2** **a=b=2**

Μέρος Α': Εμβάθυνση σε βασικές έννοιες



Οι μεταβλητές στην Python λειτουργούν σαν ετικέτες με κάποιο όνομα, που χρησιμεύουν για να αναφερόμαστε (ή αλλιώς για να δείχνουμε) σε κάποια αντικείμενα. Ο τύπος της μεταβλητής είναι ο εκάστοτε τύπος του αντικειμένου στην οποία αναφέρεται. Στο παράδειγμα a = 125, δημιουργείται η μεταβλητή με όνομα a και αναφέρεται στο αντικείμενο, με τιμή 125, με ακέραιο τύπο δεδομένων. Το σύμβολο "=" δημιουργεί ένα είδος δεσίματος μεταξύ του αντικειμένου 125 και της μεταβλητής με όνομα a. Ο τύπος της μεταβλητής είναι και αυτός ακέραιος, αφού αναφέρεται σε αντικείμενο με ακέραιο τύπο δεδομένων.

Όταν θέσουμε στη συνέχεια a = 250.0, η μεταβλητή a παύει πλέον να δείχνει το αντικείμενο με τιμή 125 και δημιουργείται ένα νέο "δέσιμο", ώστε να αναφέρεται στο αντικείμενο με τιμή 250.0 (κινητής υποδιαστολής τύπος δεδομένων). Ο τύπος δεδομένων της μεταβλητής a τώρα έχει αλλάξει και είναι κινητής υποδιαστολής, όμοιος με τον τύπο του νέου αντικειμένου στο οποίο αναφέρεται.

Η Python παρακολουθεί όλες τις τιμές και τις διαγράφει όταν πάψουν να υπάρχουν μεταβλητές που να αναφέρονται σε αυτές. Η διαδικασία αυτή ονομάζεται *συλλογή σκουπιδιών* (garbage collection).

Στο παράδειγμα που ακολουθεί η μεταβλητή a παίρνει ως τιμή διαδοχικά ένα ακέραιο, ένα πραγματικό, μια λογική τιμή και τέλος μια συμβολοσειρά. Κάθε φορά ελέγχεται ο τύπος δεδομένων με την εντολή type.

```
>>> a = 125
>>> type( a )
<type ‘int’>
>>> a = 250.7
>>> type( a )
<type ‘float’>
>>> a = True
>>> type( a )
<type ‘bool’>
>>> a =' Καλημέρα')
>>> type( a )
<type ‘str’>
```

```
>>> a = 125
>>> type (a)
<type 'int'>
>>> a = 250.7
>>> type (a)
<type 'float'>
>>> a = True
>>> type (a)
<type 'bool'>
>>> a = 'Καλημέρα'
>>> type (a)
<type 'str'>
```

Αν θέλουμε να εμφανίσουμε το περιεχόμενο της μεταβλητής, τότε μπορούμε να χρησιμοποιήσουμε την εντολή **print** μαζί με το όνο- μαόνομα της μεταβλητής. Για παράδειγμαπαρά- δειγμα: message= 'Καλημέρα', η εν- τολήεντολή print message θα εμφανίσει στην οθόνη τη λέξη Καλημέρα. Στο όνομα της μεταβλητής message, που ακολουθεί μετά την print, δεν πρέπει να χρησιμοποιηθούν εισαγωγικά (μονά ή διπλά), διότι τότε θα εμφανιστείεμφανι- στεί στην οθόνη η συμβολοσειρά message. Παρόμοια, αν θέσουμε a = 1234, η print a θα εμφανίσει την τιμή 1234 στην οθόνη.

1234 στην οθόνη.

Βασικές εντολές στη γλώσσα Python

Στις προηγούμενες παραγράφους παρουσιάστηκαν μερικές από τις βασικές εντολέςεντο- λές (statements) στην Python, όπως:

print: χρησιμοποιείται για την εμφάνιση τιμών στην οθόνη του υπολογιστή, με ποικίλεςποι- κίλες μορφές, όπως:

- print όνομα_μεταβλητής, π.χ. print name
- print αριθμός, π.χ. print 1045.34
- print 'συμβολοσειρά', π.χ. print 'θαλασσινός αέρας''θαλασσινός αέρας'
- print όνομα_μεταβλητής (τελεστής) αριθμός

```
>>> name = ‘kostas’
>>> print name
Kostas
>>> print 1045.34
1045.34
>>> message = "Τώρα "Τώρα"
>>> print message*3
    Τώρα Τώρα Τώρα
```

Εκχώρηση τιμής σε μια μεταβλητή: χρησιμοποιείται το σύμβολο "=" "=" για την εκχώρηση μιας τιμής σε μια μεταβλητή.

```
όνομα_μεταβλητής = τιμή μεταβλητής
tree = ‘Πεύκο’
x = 135
x = x **2+ 34
```

Εισαγωγή τιμής σε μια μεταβλητή από το πλητρολόγιο. Το πρόγραμμα αναμένει από το χρήστη να πληκτρολογήσει μια τιμή. Η τιμή που εισάγεται αποδίδεται σε μια μεταβλητή. Προς τούτο χρησιμοποιούνται δύο εντολές: η **input**() και η **raw_input**().

Σύνταξη: Όνομα μεταβλητής= input(‘μήνυμα’).

```
>>> number = input('Δώσε έναν αριθμό:')
Δώσε έναν αριθμό: 34
>>> print number
34
```

Αν θέλουμε να εισάγουμε ένα αλφαριθμητικό (ή αριθμό με τη μορφή αλφαριθμητικού), χρησιμοποιούμε τη raw_input:

Σύνταξη: Όνομα μεταβλητής = raw_input ()

```
>>>name = raw_input('Δώσε το όνομά σου : ')
Δώσε το όνομά σου : Kostas
>>> print name
Kostas
>>> tree = raw_input('Δώσε το όνομα ενός δέντρου με άνθη
που βλέπεις πηγαίνοντας στο σχολείο: ')
Δώσε το όνομα ενός δέντρου με άνθη που βλέπεις
πηγαίνον- τας πηγαίνοντας στο σχολείο: Αμυγδαλιά
        >>> print 'Το δέντρο ',tree,'είναι όμορφο ανθισμένο'
          Το δέντρο Αμυγδαλιά είναι όμορφο ανθισμένο
```

Γενικά, ότι εισάγεται με τη raw_input θεωρείται αυτόματα αλφαριθμητικόαλφαριθμητικό, ενώ η input προσπαθεί να το υπολογίσει. Για παράδειγμα, αν δώσουμε στην input το όνομαόνο μα μιας μεταβλητής, θα επιστρέψει το περιεχόμενο της μεταβλητής.

Εισαγωγή σχολίων

Τα σχόλια σε ένα πρόγραμμα διευκολύνουν την κατανόησή του. Στην Python, τα σχόλια εισάγονται θέτοντας μπροστά από αυτά το σύμβολο # . Τα σχόλια μπορούν να αρχίζουν και μετά από εντολές στη μέση μιας γραμμής. Ό,τι βρίσκεται δεξιά από το #, αγνοείται από το διερμηνευτή.
από το διερμηνευτή.

```
#
 #Πρόγραμμα πολλαπλασιασμός δύο αριθμών
 >>> x=input('Δώσε τον πρώτο αριθμό: ')
 Δώσε το πρώτο αριθμό: 34
 >>> y=input('Δώσε το δεύτερο αριθμό: ')
 Δώσε το δεύτερο αριθμό: 2
>>> ginomeno = x*y #πολλαπλασιάζει τις τιμές των x και y και
 # το αποτέλεσμα το εκχωρεί στη μεταβλητή ginomeno
 >>> print ginomeno
```

3.3 Βασικές (ενσωματωμένες) συναρτήσεις

Η Python παρέχει μια ποικιλία *ενσωματωμένων συναρτήσεων* για τη μετατροπή τιμών δεδομένων από έναν τύπο σε έναν άλλο, όπως οι: int(), float() και str().

Δραστηριότητα εμπέδωσης

Στο περιβάλλον της γλώσσας Python, επαληθεύστε τις παρακάτω συναρτήσεις:

- Η **float()** μετατρέπει ακεραίους και συμβολοσειρές σε δεκαδικούς αριθμούς.
- Η **int()** δέχεται οποιαδήποτε αριθμητική τιμή και τη μετατρέπει σε ακέραιο κόβοντας τα δεκαδικά ψηφία, αν υπάρχουν.
- Η **str()** δέχεται οποιαδήποτε τιμή και την μετατρέπει σε συμβολοσειρά.
- Η **abs()** επιστρέφει την απόλυτη τιμή ενός αριθμού.
- Η **pow(a,b)** επιστρέφει τη δύναμη του α υψωμένη στο β.
- Η **divmod(x,y)** επιστρέφει το ακέραιο πηλίκο και το ακέραιο υπόλοιπο της διαίρεσης x/y.

```
>>> float(10)
10.0
>>> int(5.678)
5
>>> str(5.678)
5.678
>>> abs(-45)
45
>>>
divmo
d(10,
3) (3,
1)
```

```
>>> pow(2,3)
8
```

Αν θέλουμε να διαβάσουμε από το πληκτρολόγιο έναν ακέραιο αριθμό με τη συνάρτησησυνάρτηση input(), πρέπει να χρησιμοποιούμε και τη συνάρτηση int()

a=int(input('Δώσε έναν αριθμό :'))

Περιεχόμενα

Παράδειγμα

```
>>> import math
>>> a=int(input('Δώσε~~('Δώσε~~ έναν ακέραιο αριθμό: '))~~')~~
Δώσε έναν ακέραιο αριθμό: 2345.10
# ~~#~~ο χρήστης δίνει την τιμή 2345.10
>>> print a
2345
# ~~#~~εμφανίζεται η ακέραια τιμή του αριθμού αποκόπτοντας τα δεκαδικά ψηφία
```

Δραστηριότητα

Συμβουλευτείτε τη «βοήθεια» του προγραμματιστικού περιβάλλοντος~~περιβάλλοντος~~ και πειραματιστείτε~~πειραματιστείτε~~ με τις ενσωματωμένες συναρτήσεις που παρέχει η γλώσσα Python.

Εξωτερικές βιβλιοθήκες

Εκτός από τις ενσωματωμένες συναρτήσεις οι οποίες περιλαμβάνονται~~περιλαμβάνονται~~ στη γλώσσα~~γλώσσα~~ Python, μπορεί κανείς να βρει πληθώρα εξω- τερικών~~εξωτερικών~~ βιβλιοθηκών (αρθρώματα ή modules) στους Διαδικτυακο- ύς~~Διαδικτυακούς~~ τόπους υποστήριξης της γλώσσας. Μια βιβλιοθήκη είναι ένα αρχείο το οποίο περιέχει μια συλλογή από σχετικές συναρτήσεις.

Ενδεικτικά, μπορούν να αναφερθούν: η Μαθηματική~~μαθηματική~~ βιβλιοθήκη και οι βιβλιοθήκες γραφικών. Οι βιβλιοθήκες αυτές για να χρησιμοποιηθούν,~~χρησιμοποιηθούν~~ θα πρέπει πρώτα να εισαχθούν στο πρόγραμμά μας. Η εισαγωγή αυτή γίνεται με την εντολή **import**. Για παράδειγμα~~παράδειγ- μα~~, προκειμένου να χρησιμοποιήσουμε μαθηματικές συναρτήσεις σε ένα πρόγραμμα~~πρόγραμ- μα~~, θα πρέπει μία από τις αρχικές εντολές του προγράμματός~~προγράμματός~~ μας να είναι: import math.

Για να έχουμε πρόσβαση σε μια από τις συναρτήσεις, θα πρέπει να δηλώσουμε το όνομα της μονάδας και το όνομα της συνάρτη- σης~~συνάρτησης~~, χωρισμένα με μια τελεία, μορφή που ονομάζεται *συμβολισ- μός~~συμβολισμός~~ με τελεία* (dot notation). Ας δούμε ένα παράδειγμα συνάρτη- σης~~συνάρτησης~~ για την τετραγωνική ρίζα, sqrt().

```
>>> import math
>>> riza = math.sqrt(2)
>>> print riza
1.41421356237
>>> math.sqrt(3)
1.7320508075688772
>>> x = math.pi
>>> print x
3.14159265359
```

Σε επόμενο κεφάλαιο θα δούμε διάφορα παραδείγματα από έτοι- μεςέτοιμες βιβλιοθήκες λογισμικούλο γισμικού για την υλοποίηση ελκυστικών προγ- ραμμάτωνπρογραμμάτων με χρήση γραφικών, καθώς και τη δημιουργία δικών μας.

3.4 Δομή προγράμματος και καλές πρακτικές

Μερικές καλές πρακτικές και συντακτικές συμβάσεις που πρέπει να τηρούμε κατά τη συγγραφή ενός προγράμματος, είναι οι πα- ρακάτω ναπαρακάτω:

- Δίνουμε
  - Να δίνουμε ένα χαρακτηριστικό τίτλο στο πρόγραμμα με τη μορφή σχολίων, τα οποία ξεκινάνε με το σύμβολο #. Αυτό, παρότι δεν είναι αναγκαίο, είναι ιδιαίτερα χρήσιμο.
  - Προσέχουμε Να προσέχουμε τα κενά διαστήματα πριν την κάθε εντολή, καθώς, όπως θα δούμε στο επόμενο κεφάλαιο, η Python βασίζεται σε αυτά, για να ορίσει ομάδεςομά- δες εντολών.
  - Επιλέγουμε Να επιλέγουμε τους κατάλληλους τελεστές.
  - Να χρησιμοποιούμε τα ίδια εισαγωγικά (μονά εισαγωγικά με μονά, διπλά εισαγωγικάεισα γωγικά με διπλά) για μια συμβολοσειρά.
  - Προσθέτουμε, Να προσθέτουμε, όπου κρίνουμε χρήσιμο, επεξηγηματικά σχόλια μέσα στον κώδικα. Θυμίζουμε ότι τα σχόλια περιγράφουν τη λειτουργία ενός προγράμματος προγράμ- ματος ή γενικότερα, ενός τμήματος κώδικα και γράφονται, για να βοηθήσουν τους ανθρώπους -και όχι τον υπολογιστή- στην κατανόηση και συντήρηση

ενός προγράμματος. Όταν ένα πρόγραμμα μεταφράζεται, τα σχόλια αγνοούνται~~αγνοού νται~~. Τα σχόλια που περιγράφουν τη λειτουργία ενός προγράμματος, το ρόλο των μεταβλητών και τη λειτουργία πολύπλοκων τμημάτων κώδικα, αποτελούν παράδειγμα καλού προγραμματιστικού στυλ.

- Δίνουμε~~● Να δίνουμε~~ ονόματα μεταβλητών που έχουν σχέση με τη χρήση τους.

3.5 Τύποι και δομές δεδομένων στις γλώσσες προγραμματισμού

Στην παράγραφο αυτή θα παρουσιάσουμε τους *Τύπους* και τις *Δομές Δεδομένων* στη γενική τους μορφή, όπως ορίζονται στις περισσότερες Γλώσσες Προγραμματισμού.~~Προγραμμα- τισμού.~~ Γίνεται σε συνέχεια της εξειδίκευσής τους στην γλώσσα προγραμματισμού Python που προηγήθηκε.

Δραστηριότητα

Μελετήστε, συμπληρώστε ή επεκτείνετε όπου πρέπει, τα παρακάτω~~παρακάτω~~ σχετικά με όρους των προηγούμενων κεφαλαίων.

Τύποι και Δομές Δεδομένων

Εννοιολογικοί προσδιορισμοί

Ένας *Τύπος Δεδομένων* (Data Type) είναι ένα σύνολο τιμών δεδομέ- νων~~δεδομένων~~ και λειτουργιών επί αυτών των τιμών. Οι τύποι δεδομένων είτε είναι προκαθορισμένοι~~προκαθορι- σμένοι~~ από τις γλώσσες προγραμματισμού και καλο- ύνται~~καλούνται~~

Περιεχόμενα

Πρωτογενείς τύποι δεδομένων~~Τύποι Δεδομένων~~ (Primitive data types~~Data Types~~) είτε δημιο- υργούνται~~δημιουργούνται~~ από τον προγραμματιστή~~πρόγραμμα- τιστή~~, οπότε καλούνται *Μη πρωτογενείς~~Πρωτογενείς~~ τύποι δεδομένων* (Non-primitive data types).~~Primitive Data Types).~~ Διακρίνονται δε, ανάλογα με τη σύσταση των μερών που τους αποτελούν~~αποτε- λούν~~, σε: *Απλούς* τύπους και *Σύνθετους* τύπους δεδομένων.

Απλοί τύποι δεδομένων

Περιεχόμενα

~~Στους απλούς τύπους δεδομένων, οι τιμές των δεδομένων είναι στοι-~~
~~χείαστοιχεία~~ μη περαιτέρω-χωριζόμενα (άτομα). Δηλαδή κάθε δεδομένο έχει μία και μοναδική τιμή.

Περιεχόμενα

Σύνθετοι

Σύνθετος τύπος δεδομένων ()(Composite Data Type) είναι εκείνος, που αποτελείταιαποτε λείται από Πρωτογενείς ή / και άλλους σύνθετους τύπους, όπου μια μεταβλητή μπορεί να πάρει ως τιμή μια ενότητα τιμών. Οι σύνθετοισύνθετοι τύποι καλούνται και Δομές Δεδομένων. Παραδείγματα σύνθετων τύπων είναι η Εγγραφή, το ΣύνολοΣύνο λο, ο Πίνακας.

Συνήθεις τύποι σε γλώσσες προγραμματισμού

Απλοί

- *Ακέραιος* (),(Integer), με τιμές τους ακέραιους αριθμούς μέσα σε ένα κάτω και ένα άνω όριο και πράξεις: +, -,*, /, mod, div, := καθώς και οι συγκρίσεις >, =, <
  - >, =, <
- *Πραγματικός* (real).
- *Χαρακτήρας* (C),Character), με τιμές από το σύνολο των χαρακτήρωνχαρακτήρων που διαθέτει ο υπολογιστής.
- *Λογικός* (Boolean), για την αναπαράσταση Λογικών δεδομένωνδεδομένων, με τιμές True (σωστό), False (λάθος) και επιτρεπτές πράξειςπράξεις για τις τιμές αυτές τις: and, or, not.
- *Αλφαριθμητικός*-ng-*string*.

Σύνθετοι Τύποι Δεδομένων

- *Πίνακας* ().(Array). *Συνήθως* τα στοιχεία του καθορίζονται με τη βοήθεια δεικτώνδει κτών. Μπορεί να είναι πολλών διαστάσεων.
- *Εγγραφή* ().(Record/tuple/struct).
- *Λίστα* (t(List).
- *Σύνολο* (Set).
- *Σωρός* (Heap).
- *Στοίβα* (Stack).
- *Ουρά* (Queue).
- *Δένδρο* (Tree).
- *Γράφος* (Graph).

Κατηγοριοποίηση

Επίπεδα αφαίρεσης σε Τύπους Δεδομένων

Ένας τύπος δεδομένων αρχικά και στο στάδιο σχεδίασης, υπάρχει μόνο διανοητικάμόνον δια- νοητικά, ορίζοντας έτσι έναν *αφηρημένο τύπο δεδομένων*-ΑΤΔ (Abstract data typeData Type). Στο στάδιο αυτό έχει οριστεί με βάση τις λειτουργίες που επιτελεί, κρατώνταςκρα- τώντας κρυφή την εκτέλεση. Οι περισ- σότερεςπερισσότερες αντικειμενοστρεφείς γλώσσες προγραμματισμού παρέχουν τη δυνατότητα να καθορίζονται από το χρήστη τύποι αφηρημένων στοιχείων. Δεν μας απασχολεί, στο επίπεδο αυτό, ούτε ο τρόπος με τον οποίο θα αναπαρασταθούν τα δεδομένα ούτε ο τρόπος που θα υλοποιηθούν οι λειτουργίες σε κώδικα (code implementation). Μας ενδιαφέρειενδια- φέρει δηλαδή, το τι θα υπολογιστεί και όχι το πώς. Με την πα- ραπάνωπαραπάνω λογική οι Σύνθετοι Τύποι Δεδομένων μπορούν να χαρακτη- ριστούνχαρακτηριστούν ως ΑΤΔ. Έτσι, για παράδειγμα, μια γέφυρα είναι δυνατόν είτε να την βλέπουμε ως μια οντότητα (υψηλό επίπεδο αφαίρεσης) είτε ως πολλά τμήματα -δοκοί, στερεωτικά κ.ά., όπως θα την έβλεπε ένας Μηχανικός.

4. Αλγοριθμικές δομές

Εισαγωγή

Στο κεφάλαιο αυτό γίνεται η εμβάθυνση σε βασικές έννοιες, αλγοριθμικές δομές και τεχνικές προγραμματισμού, καθώς και στην αξιοποίησή τους για την επίλυση προβλημάτων με προγραμματισμό, σε γλώσσα Python.

Λέξεις κλειδιά

Αλγοριθμικές δομές, δομή ακολουθίας, δομή επιλογής, δομή επανάληψης, συναρτήσεις.

Διδακτικές Ενότητες

4.1 Αλγοριθμικές δομές - Ροές εκτέλεσης προγράμματος

4.1.1 Δομή ακολουθίας

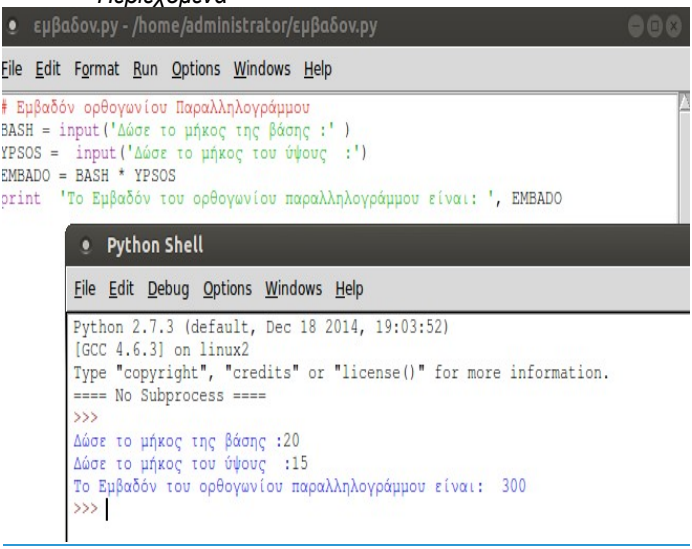
Πρόκειται για μια σειρά από εντολές που εκτελούνται η μία μετά την άλλη με τη σειρά. Η δομή ακολουθίας χρησιμοποιείται πρακτικά για την επίλυση απλών προβλημάτων, όπου είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών.

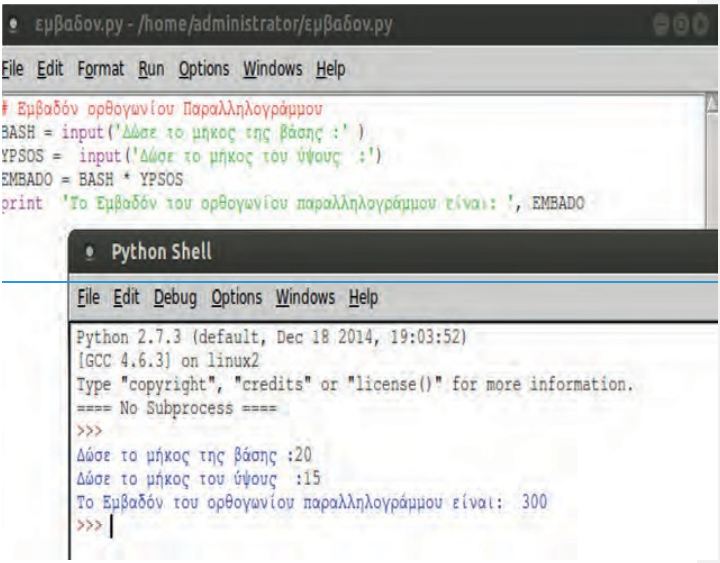
Χρησιμοποιώντας, αποκλειστικά, τη δομή ακολουθίας, μπορούμε να λύσουμε περιορισμένα προβλήματα στα οποία:

- η σειρά των βημάτων είναι καθορισμένη
- όλα τα βήματα εκτελούνται πάντοτε
- δεν υπάρχουν εξαιρέσεις.

Παράδειγμα

Να γραφεί πρόγραμμα που να υλοποιεί τον αλγόριθμο: εμβαδόν ορθογωνίου παραλληλογράμμουπα ραλληλογράμμου σε γλώσσα Python.


εμβαδον.py - /home/administrator/εμβαδον.py
File Edit Format Run Options Windows Help
# Εμβαδόν ορθογωνίου Παραλληλογράμμου
BASH = input('Δώσε το μήκος της βάσης :')
YPSOS = input('Δώσε το μήκος του ύψους :')
EMBADO = BASH * YPSOS
print 'Το Εμβαδόν του ορθογωνίου παραλληλογράμμου είναι: ', EMBADO
Python Shell
File Edit Debug Options Windows Help
Python 2.7.3 (default, Dec 18 2014, 19:03:52)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>>
Δώσε το μήκος της βάσης :20
Δώσε το μήκος του ύψους :15
Το Εμβαδόν του ορθογωνίου παραλληλογράμμου είναι: 300
>>>



4.1.2 Δομή επιλογής if ~~(ΑΝ)~~

Η δομή επιλογής if (ΑΝ) χρησιμοποιείται, όταν θέλουμε να εκτε-λεστείεκτελεστεί μια ακολουθία εντολών, μόνον, εφόσον πληρείται μία συγ- κεκριμένησυγκεκριμένη συνθήκη. Για να επιλύσουμε πιο σύνθετα προβλήματα, πρέπει να είμαστε σε θέση να δημιουργούμε στον αλγόριθμο λογι- κά ~~αλγό- ριθμο λογικά~~ μονοπάτια, εξετάζοντας απλά λογικά ερωτήματα για την εκτέ- λεσηεκτέλεση ή όχι μιας ομάδας εντολών.

Απλή δομή επιλογής

Αν η συνθήκη είναι αληθής, τότε το σύνολο των εντολών που πε-ριέχονταιπεριέχονται στην δομή if, θα εκτελεστούν. Αλλιώς, η ροή του προγ- ράμματοςπρογράμματος θα προσπεράσει τη δομή if και θα συνεχίσει από την εντολή που βρίσκεται αμέσως μετά το τέλος της if.

```
#   if <συνθήκη ελέγχου>:
```

Περιεχόμενα

εντολές του θα εκτελεσθούν αν ισχύει (αληθής-True) η συνθήκη ελέγχου

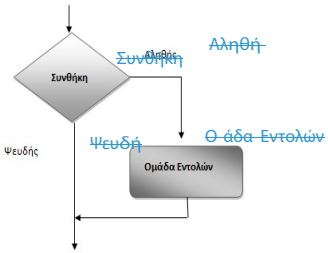
Παράδειγμα

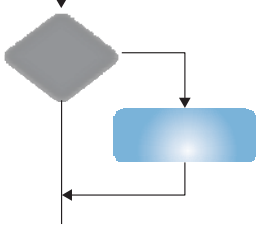
Περιεχόμενα

Μέρος 1. Εμβάθυνση σε βασικές έννοιες

Περιεχόμενα

```
#πρόγραμμα εμφάνισης προειδοποίησης αρνητικού ποσού.
a = input('Δώσε ένα ποσό')
if a <0:
        print "ΠΡΟΣΟΧΗ το ποσό είναι αρνητικό"
```





Εικόνα 4 -1. Διάγραμμα ροής για την απλήτη δομή επιλογής if...else

(Α
Ν..
.Α
ΑΛ
ΙΩ
Σ)

Σύνθετη δομή επιλογής

Αν, ανάλογα με την αποτίμηση μιας συνθήκης, θέλουμε να εκτε-λεστούν διαφορετικέςεκτελεστούν διαφορε- τικές ακολουθίες εντολών, τότε μπορούμε να χρησιμοποιήσουμε τη δομή επιλογής **if…else** (ΑΝ…ΑΛΛΙΩΣ).

Αν ισχύει η συνθήκη (έχει τιμή TRUE), θα εκτελεστεί η Α ομάδα εντολών της if, αλλιώςαλ- λιώς (αν δεν ισχύει-έχει τιμή FALSE), θα εκτε- λεστείεκτελεστεί η Β ομάδα εντολών της else.

Η εντολή ελέγχου if …. else (ΑΝ_ΑΛΛΙΩΣ) συντάσσεται:

```
if <συνθήκη ελέγχου>:
        #εντολές που θα εκτελεσθούν, αν η συνθήκη ελέγχου είναι
```

αληθής

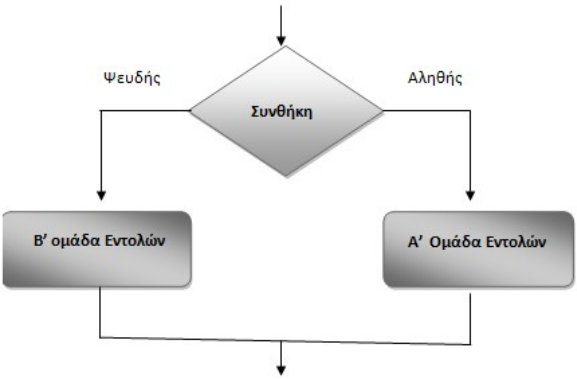
Περιεχόμενα

else:

ψευδής

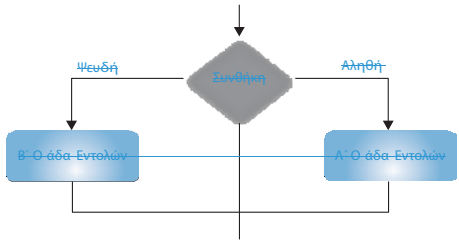
#εντολές που θα εκτελεσθούν αν η συνθήκη ελέγχου είναι ψευδής

Σημείωση: Οι ομάδες εντολών που θα εκτελεστούν, αν ισχύει μια συνθήκη, ορίζονται ~~ορίζο νται~~ ως ένα μπλοκ με τη χρήση εσοχής (κενά). Οι διαδοχικές εντολές που έχουν την ίδια εσοχή ανήκουν στο ίδιο μπλοκ. Οι εσοχές μπαίνουν αυτόματα, αν πατήσουμε Enter μετά



Εικόνα 4-2. Διάγραμμα ροής για τη δομή if-else

Enter μετά το σύμβολο της άνω και κάτω τελείας ":". Δεν πρέπει να διαγράψουμε διαγράψουμε αυτά τα κενά διαστήματα.



Δραστηριότητα: Δομή επιλογής if…else ΑΝ_ΑΛΛΙΩΣ

Να γραφεί αλγόριθμος και πρόγραμμα σε Python που να διαβάζει τις ηλικίες δύο ατόμων και να τις καταχωρεί στις μεταβλητές Α και Β αντίστοιχα. Στη συνέχεια, να συγκρίνει τις ηλικίες των δύο από- μωνατόμων και, αν η ηλικία του Α είναι μεγαλύτερη του Β, να εμφανίζει στην οθόνη το μήνυμα: ""Η ηλικία Α είναι μεγαλύτερη από τη Β". Διαφορετικά". Δια- φορετικά, να εμφανίζει στην οθόνη το μήνυμα ""Η ηλικία Α είναι μικρότερη ή ίση (<=) της ηλικίας Β". μικρότερη ή ίση (<=) της ηλικίας Β".

Πρόγραμμα σε Python

```
## Σύγκριση ενός αριθμού Α ως προς έναν αριθμό
B a = input( "Δώσε ("Δώσε την ηλικία Α του πρώτου
ατόμου" : ")
b = input( "Δώσε ("Δώσε την ηλικία ΒΒ του δεύτερου ατόμου" :")
if a > b:
    print "Η"Η ηλικία Α είναι μεγαλύτερη από τη Β"
Β" else:

  print "Η"Η ηλικία Α είναι μικρότερη ή ίση (<=) της ηλικίας Β"Β"
```

Πολλαπλή Επιλογή: Η Python προσφέρει τη δυνατότητα για σύνταξη σύνθετων δομών επιλογής με τη χρήση της εντολής **elif**.

Η σύνταξη είναι η εξής:

```
if <συνθήκη>:
    <εντολές>
elif <συνθήκη2>:
    <εντολές_2>
else:
    <εντολές_3>
```

Δραστηριότητα: Πολλαπλή επιλογή

Να γραφεί πρόγραμμα που να διαβάζει τον κωδικό συναγερμού (υποθέτουμε ότι είναι αριθμοί από 1 έως 50 που αντιστοιχούν σε δυνατά σημεία ελέγχου ενός σπιτιού) και να τυπώνει ανάλογο μήνυμα σχετικά με τη ζώνη που τον προκάλεσε. Υποθέτουμε ότι στη ζώνη 1 ανήκουν οι κωδικοί 1 έως και 10, στη Ζώνη 2 οι κωδικοί 11 έως και 20 και στη ζώνη 3 οι μεγαλύτεροι του 20.

```
#Πρόγραμμα εντοπισμού ζώνης συναγερμού
alarm = input("Δώσε έναν κωδικό συναγερμού:
") if alarm > 20:
  z=3
elif alarm>10:
  z=2
else:
  z=1
print "Πρόβλημα από τη ζώνη:
", z print "τέλος"
```

Εμφωλευμένες δομές επιλογής: Οι πολλαπλές επιλογές μπορο- ύν~~μπορούν~~ να υλοποιηθούν~~υλοποιη- θούν~~ και με εμφωλευμένες δομές Αν-αλλιώς, όπως στη δραστηριότητα που ακολουθεί~~ακο- λουθεί~~.

Δραστηριότητα ~~:~~ Δομή επιλογής - εμφωλευμένες δομές επιλογής

Α) Μελετήστε το παρακάτω πρόγραμμα.

- Τι θα τυπώσει το παρακάτω πρόγραμμα για τιμές 1, 7, 9, 16;
- Τι πιστεύετε ότι κάνει το πρόγραμμα;

Β) Να τροποποιηθεί κατάλληλα το παρακάτω πρόγραμμα, ώστε να περιέχει και την περίπτωση που ο χρήστης δώσει, κατά λάθος, ως βαθμό έναν αρνητικό αριθμό.

```
# Πρόγραμμα ελέγχου υπεριώδους ακτινοβολίας
# Εισαγωγή τιμής δείκτη ακτινοβολίας από το χρήστη
# Δεδομένα: Ο Δείκτης UV αποτελεί διεθνώς ένα μέγεθος έκ- φρασης- ~~έκφρασης~~ της επικινδυνότητας της ηλιακής υπεριώδους ακτινοβο- λίας~~ακτινοβολίας~~ που φθάνει στο έδαφος
# Όρια κινδύνου: 0-5.9 ελάχιστος ή μικρός, 6-10.9 Μεγάλος-Πολύ μεγάλος, >=11 <=15 Ακραία κατάσταση
uv = input(' Παρακαλώ δώσε έναν αριθμό έκθεσης δείκτη UV: ')
if uv<=15:
# Εμφωλευμένο if-
  ~~if uv>= 11:~~
 if uv>= 11:
     print ' Ακραία κατάσταση κινδύνου '
   else:
     if uv>= 6:
       print ' Μεγάλος ή Πολύ μεγάλος κίνδυνος '
        # για 6<= uv < 11
     else:
       if uv>= 0:
        print 'Ελάχιστος ή μικρός κίνδυνος'
        # για uv 0 <= uv < 6
else:
 # για uv >15
    print 'Ο αριθμός που έδωσες υπερβαίνει το 15. Μη αποδεκτή τιμή!'
```

4.1.3 Δομή επανάληψης (for και while)

Συχνά σε ένα πρόγραμμα, μια ομάδα εντολών είναι αναγκαίο να εκτελείται περισσότερες περισσό τερες από μία φορές. Υπάρχουν δύο τύποι επαναλήψεων επαναλήψεων:

- Οι προκαθορισμένοι, όπου το πλήθος των επαναλήψεων είναι δεδομένο, πριν αρχίσουν οι επαναλήψεις. Για παράδειγμα: ο υπολογισμός του μέσου όρου βαθμολογίας των μαθητών ενός τμήματος 22 μαθητών.
- Οι μη προκαθορισμένοι, όπου το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών του σώματος της επανάληψης. Για παράδειγμα: ο υπολογισμός των μορίων όσων υποβάλλουν αίτηση σε ένα διαγωνισμό του Δημοσίου.

Στη Στην γλώσσα προγραμματισμού Python, χρησιμοποιούμε την εν- τολή εντολή **for** για να εκτελεστεί ένα τμήμα του κώδικα για έναν καθορισ- μένο καθορισμένο αριθμό επαναλήψεων, ενώ την εντολή while για να εκτελείται υπό συνθήκη και μάλιστα, όσο αυτή είναι αληθής.

Στην εντολή for χρησιμοποιείται η συνάρτηση **range**() για τον καθορισμό καθορισμό των επαναλήψεων επα- ναλήψεων.

```
for onoma_metavlitis in range (αρχή, μέχρι, βήμα):
  Εντολή_1
  Εντολή_2
  …..
  Εντολή_ν
```

Δραστηριότητα: Χρήση της εντολής for

Να γραφεί αλγόριθμος και το αντίστοιχο πρόγραμμα σε γλώσσα Python που να υπολογίζει το άθροισμα των περιττών αριθμών από το 1 έως και το 100.

Πρόγραμμα σε Python

```
# Πρόγραμμα, αθροίζω περιττούς
athroisma = 0
for i in range(1,100,2):
  athroisma = athroisma + i
print ' Το αποτέλεσμα είναι ', athroisma
```

Προσεγγίζοντας τη συνάρτηση range

Η range() είναι μια ενσωματωμένη συνάρτηση της γλώσσας Python, η οποία, ανάμεσα~~ανά- μεσα~~ σε άλλα, χρησιμοποιείται για την υπό- δειξη~~πόδειξη~~ του αριθμού των επαναλήψεων που θα εκτελεστούν σε ένα βρόχο. Η δομή της είναι της μορφής range(αρχή, μέχρι, βήμα), όπου *αρχή, μέχρι, βήμα* ακέραιοι αριθμοί. Οι ενδείξεις της *αρχής* και του *βήματος* δεν είναι υποχρεωτικές και, αν δεν αναφέρονται, θα αρχίσει από 0 και θα συνεχίσει με βήμα 1. Αντίθετα η ένδειξη *μέχρι* πρέπει πάντα να αναφέρεται.

Παραδείγματα

- range(10), παράγει τη λίστα: [0,1,2,3,4,5,6,7,8,9].
- range(1, 8), παράγει τη λίστα: [1,2,3,4,5,6,7]
- range(8, -1, -1), παράγει τη λίστα [8, 7, 6, 5, 4, 3, 2, 1, 0]

Δομή Επανάληψης με while βρόχο

Η δομή while (~~ή~~ **Όσο <συνθήκη> επανάλαβε)**

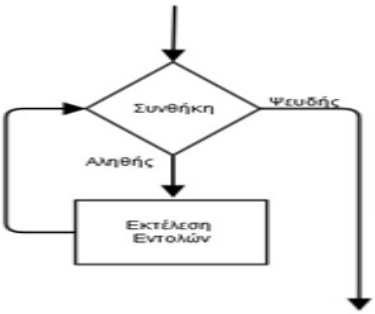
~~Η δομή while~~ χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων. Σε κάθε επανάληψη (και στην αρχική) πραγματοποιείται ο έλεγχος της συνθήκης, πριν από την εκτέλεση των εντολών του βρόχου, πράγμα που σημαίνει ~~ότι υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου.~~

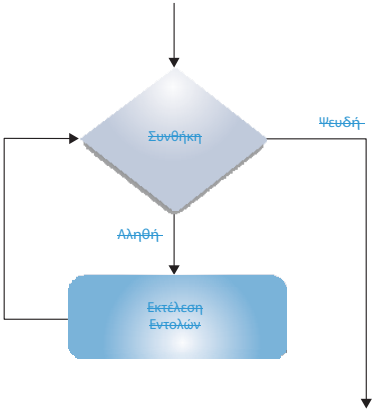
ότι υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου.

```
Αρχική τιμή
μεταβλητής
while ονομα_μεταβλητής
 <συνθήκη>: Εντολή_1
 Εντολή_2
 …
 .

 E
 v
 τ
 ο
 λ
 ή
 _
 κ
```

Σημείωση1: θα πρέπει μέσα στο μπλοκ εντολών να υπάρχει κατάλληλη εντολή, ώστε να εξασφαλίζεται ότι κάποια στιγμή η συνθήκη θα γίνει ψευδής και θα διακοπείδιακο- πεί ο βρόχος. Διαφορετικά ο βρόχος δε θα τερματίζει.

Σημείωση 2: πριν το βρόχο while θα πρέπει αρχικά να δώσουμε μία τιμή στη μεταβλητήμε- ταβλητή που ελέγχει τη συνθήκη του βρόχου, ώστε ανάλογα να εκτελεστεί ή όχι ο βρόχος.





Παράδειγμα

```
# Πρόγραμμα επανάληψης μεεπανάληψη while
x = 40              # αρχική τιμή στη μεταβλητή x
while x < 50 :   # έλεγχος της επανάληψης
    x = x + 1      # αύξηση του μετρητή
    print x
print x
```

Μια συνήθης εφαρμογή του while βρόχου είναι να ελέγχει την εγκυρότητα των δεδομένων εισόδου από το χρήστη.

Δραστηριότητα

Συμπλήρωσε κατάλληλα την παρακάτω συνθήκη, έτσι ώστε να εξασφαλίζεται ότι ο χρήστης θα εισάγει τελικά την τιμή 0-20 για το βαθμό στο μάθημα Εφαρμογές Πληροφορικής.

```
#Ελεγχος εισαγωγής δεδομένων
choice = input(' Δώστε το βαθμό που πήρατε στο μάθημα Εφαρμογές Πληροφορικής')
while …….. …….. ……..:
    choice = input ('Παρακαλώ δώστε έγκυρη τιμή ')
```

Δραστηριότητα: Δομή επανάληψης

Βρείτε τι κάνει το παρακάτω πρόγραμμα και δώστε με μία πρόταση ένα χαρακτηριστικό τίτλο.

Σημείωση: Η εντολή "import random" εισάγει μια βιβλιοθήκη συναρτήσεων για την παραγωγή τυχαίων αριθμών. Η συνάρτηση random επιστρέφει έναν τυχαίο δεκαδικό ανάμεσα στο 0.0 και στο 1.0 (συμπεριλαμβανομένου του 0.0, αλλά όχι του 1.0). Η συνάρτηση randint παίρνει ως παραμέτρους ένα κάτω και ένα άνω όριο και επιστρέφει έναν ακέραιο μεταξύ αυτών των ορίων, συμπεριλαμβανομένων και αυτών των δύο.

```
import random
thenum = random.randint(1,10)
print "Ψήφισα έναν από τους 10 υποψηφίους για πρόεδρο 15μελούς"
print "Μπορείς να μαντέψεις τον αύξοντα αριθμό αυτού που ψήφισα"
guess = 0
while guess != thenum:
```

Μέρος 1. Εμβάθυνση σε βασικές έννοιες

```
guess = input("Δώσε("Δώσε αριθμό: ")")
if guess>thenum:
  print "Εδωσες"Εδωσες μεγαλύτερο αριθμό"
   αριθμό" elif
guess<thenum:
  print "Εδωσες"Εδωσες μικρότερο αριθμό"
  αριθμό" else:
   print "Τον"Τον βρήκες! ""
```

Δραστηριότητα.: Εμφωλευμένη δομή επανάληψης

Σε έναν αθλητικό μαθητικό αγώνα στίβου, στο αγώνισμα του μήκουςμήκους, συμμετέχουν στους προκριματικούς 20 μαθητές από όλα τα σχολεία της Περιφέρειας. Στον τελικό περνούν όσοι μαθητές σημειώσουνσημειώσουν επίδοση μεγαλύτερη ή ίση από 4.5 μέτρα. Κάθε αθλητής έχει 3 προσπάθειες. Αν σημειώσει επίδοση ίση ή μεγαλύτερη από το όριο πρόκρισης, σταματάει τις προσπάθειες. Να γραφεί αλγόριθμοςαλγόριθμος και στη συνέχεια αντίστοιχο πρόγραμμα σε Python, που να διαβάζει τις επιδόσεις των αλμάτων κάθε αθλητή και να υπολογίζει την καλύτερη επίδοσή του. Να ελέγχει δίνοντας ανάλογο μήνυμα στην οθόνη αν ο αθλητής προκρίθηκε ή όχι στον τελικό και τελικά να εμφα νίζει στην οθόνη, πόσοι αθλητές προκρίθηκαν και ποια ήταν η καλύτερη επίδοση που σημειώθηκε.

να εμφανίζει στην οθόνη, πόσοι αθλητές προκρίθηκαν και ποια ήταν η καλύτερη επίδοση που σημειώθηκε.

```
# -*- coding: cp1253 -*-
# Πρόγραμμα

Αλγόριθμος πρόκριση στο στίβο και εύρεση καλύτερης επίδοσης
max_alma =← 0
count_athletes =←
0 for
Για i in range(=1,21):
 μέχρι 20 max_epidosi
        =← 0
        prospatheia =←
        1
        whileΌσο prospatheia<=3 andκαι max_epidosi<4.5 :επανάλαβε
print "Εμφάνισε ("αγωνίζεται ο", ",i, ""ος αθλητής στην ", ",prospatheia, ""η
```

```
προσπάθεια"
epidosi = input( "Δώσε την επίδοση του αθλητή:" ):") Διάβασε epidosi
if Αν max_epidosi<epidosi :τότε max_epidosi = epidosi
if Αν epidosi>=4.5: τότε
print 'Ο', Εμφάνισε 'Ο' i, 'ος' ος αθλητής προκρίθηκε με άλμα στα ', στα',epidosi, 'μέτρα' μ.'
count_athletes = =count_athletes + +1
Τέλος _αν Τέλος_αν
prospatheia = =prospatheia + +1
if max_ Τέλος_Επανάληψης
Αν epidosi < 4.5 :τότε
print 'ΔενΕμφάνισε ('Ο',i,'ος αθλητής δεν προκρίθηκε. Το καλύτερο άλμα του ήταν:',max_epidosi)
if Αν max_alma < <max_epidosi :
τότε max_alma = =max_epidosi
print 'Τελικά προκρίθηκαν ', Τέλος_Επανάληψης
Εμφάνισε ('Τελικά προκρίθηκαν' ,count_athletes , ,'αθλητές')
print 'ΗΕμφάνισε ( 'Η καλύτερη επίδοση που σημειώθηκε ήταν ', ήταν',max_alma , 'μέτρα',’m.’) Τέλος πρόκριση στο στίβο και εύρεση καλύτερης επίδοσης
```

```
#Πρόγραμμα πρόκριση στο στίβο και εύρεση καλύτερης επίδοσης
max_alma=0
count_athletes=0
for i in range(1,21):
    max_epidosi=0
    prospatheia=1
    while prospatheia<=3 and max_epidosi<4.5:
        print 'αγωνίζεται ο',i,'ος αθλητής στην',prospatheia,'η προσπάθεια'
        epidosi=input('Δώσε την επίδοση του αθλητή: ')
        if max_epidosi<epidosi:
            max_epidosi=epidosi
            if epidosi>=4.5:
                print 'Ο',i,'ος αθλητής προκρίθηκε με άλμα στα',epidosi,'μέτρα'
                count_athletes=count_athletes+1
        prospatheia=prospatheia+1
    if max_epidosi < 4.5:
        print 'ο',i,'ος αθλητής δεν προκρίθηκε. Το καλύτερο άλμα του ήταν:',max_epidosi
    if max_alma<max_epidosi:
        max_alma=max_epidosi
print 'Τελικά προκρίθηκαν',count_athletes,'αθλητές'
print 'Η καλύτερη επίδοση που σημειώθηκε ήταν',max_alma,'μέτρα'
```

4.2 Συναρτήσεις

Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Μας επιτρέπουν να δίνουμε ένα όνομα σε ένα σύνολο εντολών και να το εκτελούμε καλώντας το όνομα αυτό, από οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε, διαδικασία που ονομάζεται *κλήση* (calling) της συνάρτησης.

4.2.1 Δημιουργώντας δικές μας συναρτήσεις

Για να ορίσουμε μια δική μας συνάρτηση χρησιμοποιούμε τη χαρακτηριστική λέξη **def**, ακολουθεί ένα όνομα που ταυτοποιεί την εκάστοτε συνάρτηση και ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν ονόματα μεταβλητών, ενώ η γραμμή τελειώνει με άνω και κάτω τελεία (:).

Ας δούμε ένα απλό παράδειγμα:

```
def hellohellow ():
    print 'Γεια('Γεια σου
        κόσμε!')
# Τέλος της συνάρτησης
hello()    hellow() # κλήση της συνάρτησης
hello()    hellow() # κι άλλη μία κλήση της συνάρτησης
```

Αν θέλουμε μπορούμε να φτιάξουμε άλλη μία συνάρτηση, την epanalave_hellohellow(), που να καλεί την hellohellow δύο φορές. Σκεφτείτε τι θα εμφανιστεί στην οθόνη, όταν εκτελεστούν τα παρακάτω. ΔημιουργήστεΔημιουργήστε μια ακόμα συνάρτηση που να καλεί και να χρησιμοποιεί κατάλληλα την epanalave_hellohellow(), ώστε να εμφανίζει τα λόγια τέσσερις φορές.

```
def epanelave_hellohellow():
    hello()
    hello()
# hellow()
hellow()
#τέλος συνάρτησης
epanalave_hellohell
ow()
def epanalave_4fores():
    epanalave_hellohello
    w()
    epanalave_hellohello
    w()
#τέλος συνάρτησης
epanalave_4fores()
```

4.2.2 Παράμετροι συναρτήσεων

Μια συνάρτηση δέχεται δεδομένα μέσω των παραμέτρων και επιστρέφει τα αποτελέσματα μέσω άλλων ή και των ίδιων παραμέτρων στο πρόγραμμα ή σε άλλη συνάρτηση.

Όπως αναφέρθηκε, μια συνάρτηση μπορεί να δεχθεί παραμέτρους, οι οποίες χρη-

σιμοποιούνται για να παρέχουμε διάφορες τιμές σε αυτήν, έτσι ώστε να παράγει κάποιο αποτέλεσμα ή να εκτελεί κάποιες ενέργειες χρησιμοποιώντας τις τιμές αυτές. Οι παράμετροι μοιάζουν με τις μεταβλητές, με τη διαφορά ότι οι τιμές αυτών των μεταβλητών ορίζονται, όταν καλούμε τη συνάρτηση.

Οι παράμετροι καθορίζονται μέσα στο ζευγάρι των παρενθέσεων στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα. Όταν *καλούμε* τη συνάρτηση, δίνουμε και τις τιμές με τον ίδιο τρόπο, οι οποίες τιμές ονομάζονται ορίσματα.

Σημείωση για την ορολογία που χρησιμοποιείται: οι ονομασίες που δίνουμε στον ορισμό της συνάρτησης ονομάζονται *παράμετροι*, ενώ οι τιμές που δίνουμε, όταν καλούμε τη συνάρτηση, ονομάζονται *ορίσματα*.

Κάποιες από τις ενσωματωμένες συναρτήσεις που έχουμε ήδη συναντήσει, δεν απαιτούν ορίσματα, όπως, για παράδειγμα, όταν καλούμε τη math.pi, που χρησιμοποιείται για τη μαθηματική στα- θερά π 3.14…, όπου δεν έχουμε κάποιο όρισμα. Σε άλλες όμως, συναρτήσεις απαιτούνται ένα ή και περισσότερα ορίσματα, όπως στη math.pow, η οποία υπολογίζει την ύψωση σε δύναμη ακεραί- ων, που απαιτούνται δύο, ένα για τη βάση και ένα για τον εκθέτη.

```
def ginomeno (a,b):
    x = a * b
    return x
print (ginomeno(5,10))
```

Περισσότερα για τις συναρτήσεις και την εμβέλεια των παραμέτ- ρων θα έχουμε την ευκαιρία να μελετήσουμε στο κεφάλαιο 7.

Περιεχόμενα

Μέρος

Περιεχόμενα

5

Κλασικοί Αλγόριθμοι II

5. Κλασικοί Αλγόριθμοι II

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται ορισμένοι από τους βασικούς αλγορίθμους της Επιστήμης της Πληροφορικής~~ Επιστήμης~~. Η έννοια του αλ- γορίθμου~~αλγορίθμου~~ αποτελεί τον ακρογωνιαίο λίθο της Επιστήμης~~επιστήμης~~ της Πλη- ροφορικής~~Πληροφορικής~~, μια και το βασικό πρόβλημα το οποίο εξετάζει είναι~~εί- ναι~~ ο σχεδιασμός αλγορίθμων για την επίλυση προβλημάτων. Αυτό ό- μως~~όμως~~ δε σημαίνει~~σημαί- νει~~ ότι, αν επινοήσουμε έναν οποιονδήποτε αλγόριθ- μο~~αλγόριθμο~~ που να λύνει το πρόβλημα που θέλουμε, αυτό είναι αρκετό. Το βασικό, αλλά και αιώνιο ερώτημα της Πληροφορικής~~Πληρο φορικής~~, είναι “Μπο- ρούμε~~Μπορούμε~~ καλύτερα;~~”.;”~~ Μπορούμε να σχεδιάσουμε έναν αλγόριθμο ο οποίος να κάνει καλύτερη διαχείριση των υπολογιστικών πόρων του συστήματος που είναι η μνήμη(~~μνήμης~~ και ο επεξεργαστής;~~επεξεργαστή);~~ Σήμερα το βασικό πρόβλημα δεν είναι τόσο η χρήση της κύριας μνήμης, αλλά ο χρόνος εκτέλεσης του αλγορίθμου. Ο στόχος μας λοιπόν, δεν είναι να λύσουμε απλώς ένα πρόβλημα, αλλά να το λύσουμε με το βέλτιστο τρόπο. Πολλές φορές δεν υπάρχει ένας αλγόριθμος που να είναι βέλτιστος για όλες τις περιπτώσεις δεδομένων, αλλά εξαρ- τάται~~εξαρτάται~~ από την οργάνωση των δεδομένων. Για παράδειγμα η *σειρι- ακή~~σειριακή~~ αναζήτηση* σαρώνει όλα τα στοιχεία μιας ταξινομημένης λίσ- τας~~λίστας~~ χωρίς να εκμεταλλεύεται τη διάταξη των στοιχείων. Ο σχεδι- ασμός~~σχεδιασμός~~ ενός νέου αλγορίθμου που θα εκμεταλλεύεται την ιδιαίτερη δομή των δεδομένων, για να βρίσκει~~βρί- σκει~~ το ζητούμενο στοιχείο πιο γρήγορα, αποτελεί ένα από τα αντικείμενα αυτής της ενότητας.

Επειδή υπάρχουν διάφορες κατηγορίες δεδομένων, οι επιστήμο- νες~~επιστήμονες~~ της Πληροφορικής~~Πληροφο ρικής~~ σχεδιάζουν διάφορους αλγορίθμους ανά- λογα~~ανάλογα~~ με την ιδιαίτερη δομή των δεδομένων. Γι'δε- δομένων. Γι' αυτόν το λόγο δεν έχουμε μόνο έναν αλγόριθμο ταξινόμησης, αλλά αρκετούς, κάποι- ους~~κάποιους~~ από τους οποίους θα μελετήσουμε σε αυτή την ενότητα.

~~Κεφ.5: Κλασικοί αλγόριθμοι II~~

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου, θα μπορούμε να:

- αναλύουμε και να εφαρμόζουμε κατάλληλα κλασικούς αλγορίθμουςαλγορίθμους για προβλήματαπρο βλήματα ταξινόμησης και αναζήτησης
- υλοποιούμε σε μια γλώσσα προγραμματισμού κλασικούς αλ- γορίθμους ταξινόμησηςαλγορίθμους ταξι- νόμησης και αναζήτησης
- συγκρίνουμε και να επιλέγουμε τον κατάλληλο αλγόριθμο ανά- λογαανάλογα με το είδος του προβλήματος.

Λέξεις κλειδιά

Αλγόριθμοι, ταξινόμηση, αναζήτηση.

Διδακτικές Ενότητες

5.1 Δυαδική αναζήτηση

Στην προηγούμενη τάξη λύσαμε το πρόβλημα της αναζήτησης, με τον αλγόριθμο της *Σειριακής αναζήτησης*. Ο αλγόριθμος αυτός στην χειρότερη περίπτωση, όπου το στοιχείο δεν υπάρχει στο σύ- νολοσύνολο δεδομένων ή είναι το τελευταίο στη σειρά, θα ελέγξει όλα τα στοιχεία του συνόλου στο οποίο ψάχνουμε. Υπάρχει τρόπος να βρούμεβρού- με το ζητούμενο στοιχείο πιο γρήγορα; Μια ιδέα είναι να εκ- μεταλλευτούμεεκμεταλλευτούμε τη δομή του συνόλου των δεδομένων, αλλά θα πρέπει να ξέρουμε αν τα δεδομένα είναι τυχαία τοποθετημένα ή με βάση κάποια λογική. Για παράδειγμα: τα ονόματα σε μια λίστα μπορεί να είναι σε αλφαβητική σειρά ή οι απόφοιτοι μιας σχολής σε φθίνουσα σειρά με βάση τον βαθμό πτυχίου.

Ας παίξουμε το εξής παιχνίδι:

Σκεφτείτε έναν αριθμό από το 1 έως το 1000 και απαντήστε στην ερώτηση:

Ο αριθμός που σκεφτήκατε είναι μεγαλύτερος ή μικρότερος από το 500;

Ο αριθμός που σκεφτήκατε είναι μεγαλύτερος ή μικρότερος από το 500;

Ας υποθέσουμε ότι ο αριθμός που σκεφτήκαμε είναι μικρότερος του 500, άρα η απάντηση στην ερώτηση είναι μικρότερος. Αφού ο αριθμός ξέρουμε ότι είναι μικρότεροςμικρό τερος του 500, τότε θα βρίσκεται σίγουρα μεταξύ του 1 και του 500. Έτσι, ενώ στην
αρχή είχαμε να ψάξουμε μεταξύ 1000 αριθμών τώρα με μία ερώτηση καταφέραμε

να περιορίσουμε το χώρο αναζήτησης του προβλήματος στο μισό. Με μια δεύτερη ερώτηση για τη σχέση του αριθμού μας με το 250 και ανεξάρτητα από την απάντησηαπάντη- ση, περιορίζουμε πάλι το πρόβλη- μαπρόβλημα στο μισό του μισού, δηλαδή στο ¼ του αρχικού προβλήματος.

Μπορείτε να συνεχίσετε το παιχνίδι; Πόσα βήματα θα χρειαστούν για να βρούμε τον αριθμό;

Η παραπάνω μέθοδος είναι γνωστή ως **Δυαδική αναζήτηση** και εκμεταλλεύεται τη διάταξη των στοιχείων του συνόλου, διαμερίζον- ταςδιαμερίζοντας κάθε φορά το σύνολο σε δυο ίσα μέρη, και εφαρμόζοντας πάλι την ίδια μέθοδο στο υποσύνολο στο οποίο ανήκει ο ζητούμενος αριθμός.

Ας δούμε πώς μπορούμε να αποτυπώσουμε την παραπάνω ιδέα σε πρόγραμμα, υλοποιώντας τον αλγόριθμο αυτόν σε Python.

Δραστηριότητα: Μάντεψε τον αριθμό

Βήμα 1: Παραγωγή τυχαίων αριθμών

Πολλές φορές θέλουμε να παράγουμε αριθμούς με τυχαίο τρόπο. Η βιβλιοθήκη *random* περιέχει μια ποικιλία συναρτήσεων γι΄ αυτόν τον σκοπό. Δύο από αυτές που θα χρησιμοποιούμε κατά κόρον, είναι η **randint** και η **randrange**, που επιστρέφουνεπιστρέ- φουν τυχαίους ακέ- ραιουςακέραιους αριθμούς εντός κάποιων ορίων. Τα όρια στην randrange ακολουθούν την ίδια λογική με την range της Python, ενώ η randint συμπεριλαμβά- νει και το δεξιό άκρο.
randint συμπεριλαμβάνει και το άνω άκρο.

```
import random
number = random.randint(1, 10)
                # επιστρέφει έναν τυχαίο αριθμό στο [ 1, 10]
number = random.randrange(1,10)
                # επιστρέφει έναν τυχαίο αριθμό στο [ 1, 9]
number = random.randrange(10)
                # επιστρέφει έναν τυχαίο αριθμό στο [ 0, 9]
```

Βήμα 2: Ψάχνοντας στα τυφλά

Το παρακάτω πρόγραμμα επιλέγει έναν αριθμό στην τύχη στο δι- άστημαδιάστημα [1,100] και στη συνέχεια δίνει στο χρήστη όσες προσπάθε- ιες χρειάζεταιάπειρες προσπάθειες μέχρι να μαντέψει τον αριθμό. Ο χρήστης μαντεύει στην τύχη, χωρίς κάποια συγκεκριμένη στρατηγική, με το πρόγ- ραμμαπρό- γραμμα να χρησιμοποιεί έναν μετρητή, τον guesses, για να μετράει τις προσπάθειες του χρήστη.

```
import random
secret_number = random.randint(1, 100)

guesses = 0
found = False

while not found  :
  guess = input("Μάντεψε τον αριθμό : ")
  guesses = guesses + 1
  if  guess == secret_number :
     print "Μπράβο το βρήκες με ", guesses, " προσπάθειες"
     found = True
  else :
   print "Δυστυχώς δεν το βρήκες, Ξαναπροσπάθησε"
```

Βήμα 3: Καταστρώνοντας τη στρατηγική

Θα τροποποιήσουμε το πρόγραμμα έτσι, ώστε να δίνει μια βοήθε- ιαβοήθεια στον παίκτη. Θα ελέγχει αν ο αριθμός που μάντεψε είναι μικρό- τεροςμικρότερος ή μεγαλύτερος από τον μυστικό αριθμό και θα εμφανίζει κα- τάλληλοκατάλληλο μήνυμα. Για να γίνει αυτό, θα χρειαστεί να προσθέσουμε δύοπρο- σθέσουμε δυο ακόμα περιπτώσεις στη δομή επιλογής, όπως φαίνεται παρα- κάτωπαρακάτω:

```
if guess == secret_number :
  print “Μπράβο το βρήκες με ”, guesses, “ προσπάθειες”
  found = True
else :
  if guess < secret_number:
    print “Ο αριθμός σου είναι μικρότερος από τον ζητούμενο”
 else :
    print “ Ο αριθμός σου είναι μεγαλύτερος από τον ζητούμενο ”
```

Έτσι, αν ο αριθμός που ψάχνει ο παίκτης είναι μικρότερος από αυτόν που έδωσε, τότε ξέρει ότι πρέπει να ψάξει από εκεί και κά- τω~~κάτω~~, με αποτέλεσμα το μέγεθος του προβλήματος να μειωθεί στο μισό. Επίσης, το πρόγραμμα θα επιτρέπει στον παίκτη μέχρι 10 προσπάθειες ~~προ- σπάθειες~~ και οι αριθμοί θα είναι στο διάστημα [1, 1000]. ~~Παρατήρηση: αρκούν άρα γε 10 προσπάθειες για 1000 αριθμούς; Αν ναι, τότε για 1.000.000 αριθμούς πόσες προσπάθειες πιστεύετε ότι χρειάζονται;~~

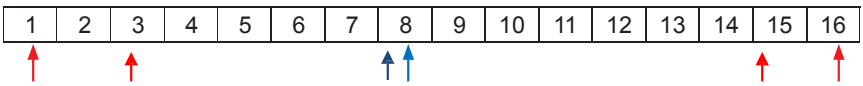
Παρατήρηση: αρκούν άραγε 10 προσπάθειες για 1000 αριθμούς; Αν ναι, τότε για 1.000.000 αριθμούς πόσες προσπάθειες πιστεύετε
ότι χρειάζονται;

```
import random
secret_number = random.randint(1, 1000)

guesses = 0
found = False

while not found and guesses < 10 :
    guess = input(“Μάντεψε τον αριθμό : ”)
    guesses = guesses + 1
    if  guess == secret_number :
      found = True
    else :
        if guess < secret_number :
```

```
        print "Ο αριθμός σου είναι μικρότερος"
    else :
        print " Ο αριθμός σου είναι μεγαλύτερος "
if  found == True :
    print "Μπράβο το βρήκες με ", guesses, " προσπάθειες"
else :
    print "Δυστυχώς δεν το βρήκες"
```

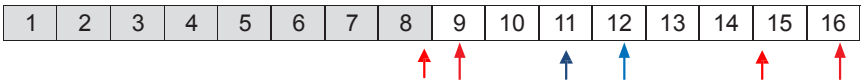
Βήμα 4: Σκέφτεται ο υπολογιστής;

Θα είχε ενδιαφέρον η εναλλαγή ρόλων μεταξύ υπολογιστή και πα- ίκτη~~παίκτη~~, όπου ο υπολογιστής~~υπο- λογιστής~~ θα ρωτάει σε κάθε βήμα και ο χρήστης θα απαντάει ανάλογα. Κάθε φορά που θα λαμβάνουμε μια απάν- τηση~~απάντηση~~ από το χρήστη πρέπει να περιορίζουμε το σύνολο~~σύ- νολο~~ των δεδο- μένων~~δεδομένων~~ στο οποίο γίνεται η αναζήτηση. Γ'~~Γι'~~ αυτό θα χρειαστούμε δύο μεταβλητές οι οποίες θα καθορίζουν το διάστημα στο οποίο βρίσκεται ο ζητούμενος αριθμός και μια τρίτη για το μέσο του διασ- τήματος.~~διαστήματος.~~ Ακολουθεί ένα παράδειγμα του παιχνιδιού όπου ο μυστι- κός~~μυστικός~~ αριθμός είναι το 11 και το διάστημα είναι το [1,16]. Αρχικά~~Αρ- χικά~~ το πρόγραμμα ρωτάει το χρήστη αν ο αριθμός είναι μικρότερος, με- ~~μεγαλύτερος ή ίσος με το μέσον του διαστήματος που είναι το 8.~~
γαλύτερος ή ίσος με το μέσον του διαστήματος που είναι το 8.



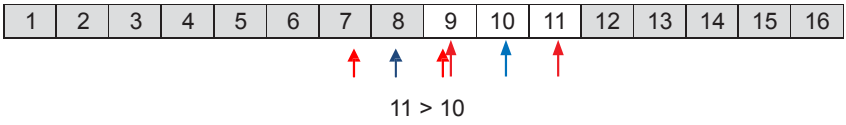
11~~15~~ >
8

Ο χρήστης απαντά ότι ο αριθμός είναι μεγαλύτερος από το 8, άρα το διάστημα στο οποίο θα πρέπει να ψάξουμε τώρα είναι από το ~~μέσον και πέρα, δηλαδή από το 9 και πάνω.~~
μέσον και πέρα, δηλαδή από το 9 και πάνω.



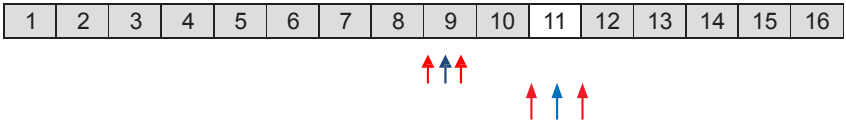
11 < 12

Συνεχίζουμε την ίδια διαδικασία υπολογίζοντας πάντα το μέσον~~μέσο~~ του διαστήματος που έχει απομείνει και αναπροσαρμόζοντας κα- τάλληλα~~κατάλληλα~~ τα άκρα του. Τώρα ο αριθμός που ψάχνουμε είναι μικρό- τερος~~μικρότερος~~ από το 12, άρα βρίσκεται αριστερά του και χρειάζεται~~χρειάζε- ται~~ να μεταφέρουμε αριστερά το δεξί άκρο.



11 > 10

Όμοια τώρα πρέπει το αριστερό άκρο να έρθει δεξιά.



Όπως διαπιστώσατε, η εύρεση του μυστικού αριθμού δε χρειάσ- τηκε~~χρειάστηκε~~ 16 συγκρίσεις αλλά μόλις 4. Ο αλγόριθμος που μόλις περιγ- ράψαμε~~περιγράψαμε~~ είναι γνωστός ως **αλγόριθμος της δυαδικής αναζήτη- σης~~αναζήτησης~~**, γιατί σε κάθε βήμα μειώνει~~χωρίζει~~ το χώρο αναζήτησης στο μισό. Στη συγκεκριμένη περίπτωση, που ο χώρος αναζήτησης ήταν το διάστημα [1,16], χρειάστηκαν 4 συγκρίσεις, αφού χρειάζονται τέσ- σερις~~τέσσερις~~ διαδοχικές διαιρέσεις με το 2, για να καταλήξουμε από 16 στοιχεία σε 1. Μπορείτε να υπολογίσετε πόσες συγκρίσεις θα θέ- λαμε~~συγκρίσεις θα θέλαμε~~, αν είχαμε 1024 = 2^{10} στοιχεία;

```
# Τώρα ο άνθρωπος σκέφτεται έναν αριθμό από 1 έως 1000
N = 1000
print "Σκέψου έναν αριθμό από το 1 έως το ", N

guesses = 0
found = False
first = 1
last = N
```

Κεφ.5: Κλασικοί αλγόριθμοι II

```
while not found and guesses < 10 :
    mid = ( first + last ) // 2
    answer = raw_input("Είναι ο αριθμός ο" + str(" " + mid) + " ? (N/O)")")
    guesses =
    guesses +
    1 if answer
    === "N" :
     f
    ou
    nd
    =
    Tr
    ue
    el
    se
    :
        answer = raw_input("Είναι μικρότερος του"+str(" " + mid)+"?" + " ? (N/O)" )
        if answer === "N" :
            last = mid – 1
        else :
          fir
        st =
        mid
        + 1
        els
        e :
          la
st =
```

```
mid –
1 if
found
==
True :
  print “Κέρδισα!!! Το βρήκα με ”, guesses, “ προσπάθειες”
else :
  print “Κέρδισες”
```

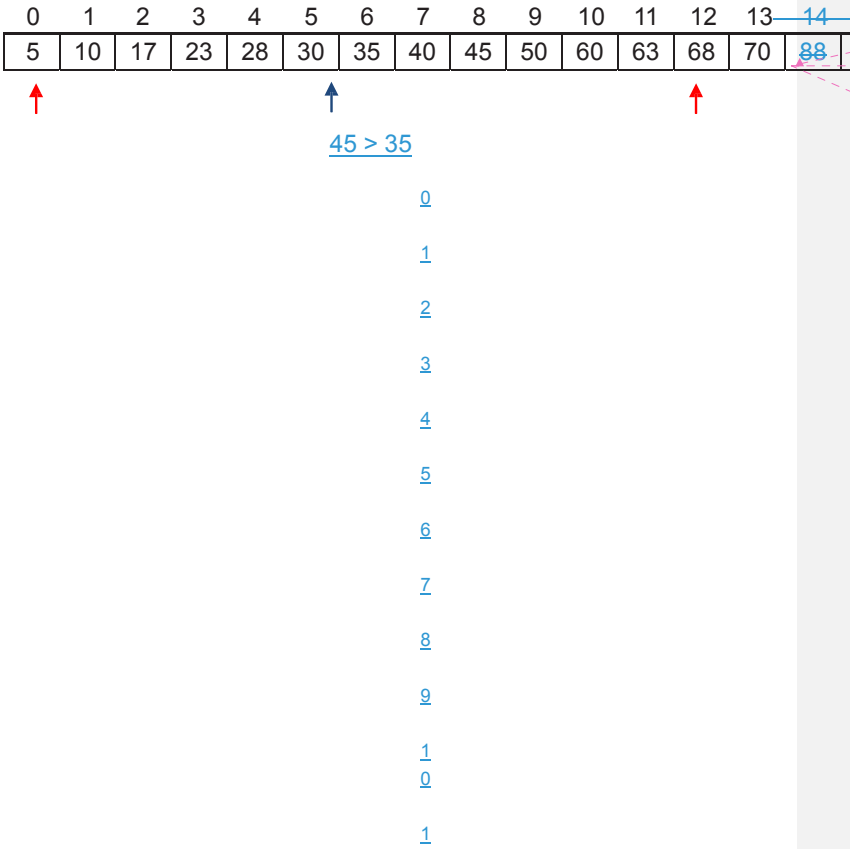
Στο παραπάνω πρόγραμμα καταχωρούμε στις μεταβλητές first και last καταχωρούμε τις θέσεις των δύοδυο άκρων του διαστήματος, ενώ στη και στην mid το μέσο του διαστήματος.

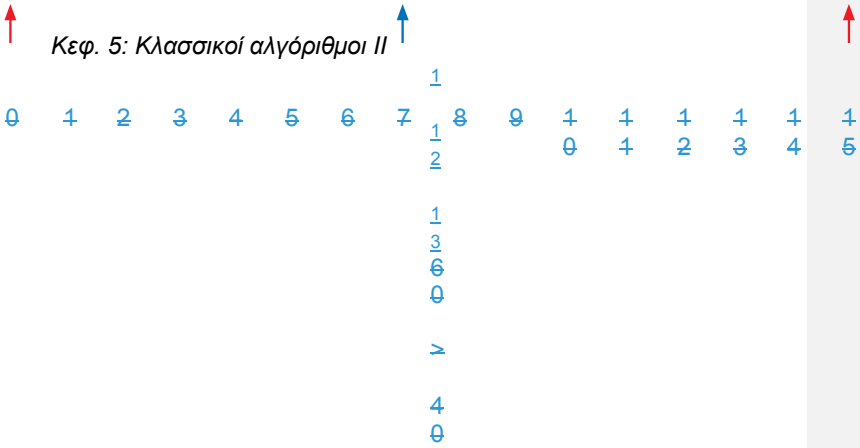
Δυαδική αναζήτηση σε λίστα

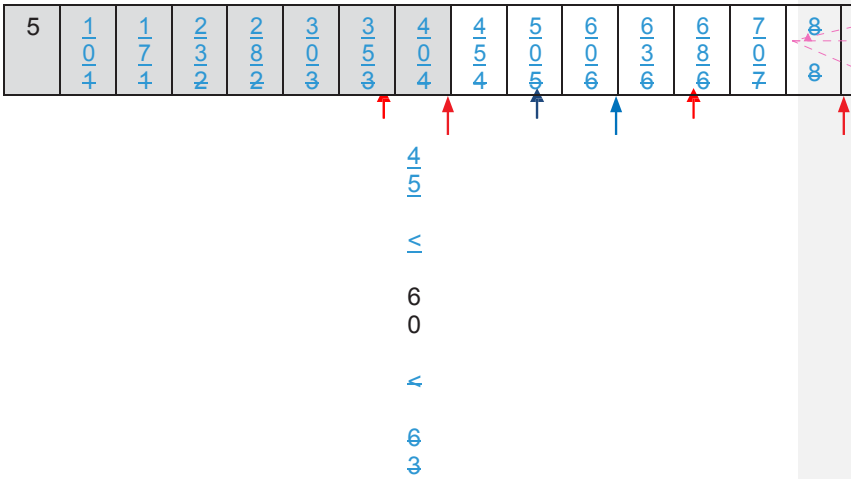
Προηγουμένως χρησιμοποιήσαμε τον αλγόριθμο της δυαδικής αναζήτησης για να βρούμε έναέναν μυστικό αριθμό μέσα σε κάποια όρια. Ο αλγόριθμος βασίζεται στο γεγονός ότι οι αριθμοί είναι διατεταγμένοιδιατεταγμένοι κατά αύξουσα σειρά. Για να εφαρμοστεί ο ίδιος αλγό- ριθμοςαλγόριθμος και σε άλλα δεδομένα, όπως για παράδειγμα σε ονόματα, θα πρέπει αυτά να είναι διατεταγμένα επίσης σε κάποιου είδους σειρά, όπως σε αλφαβητικήαλφα- βητική. Για παράδειγμα με χρήση της δυαδι- κήςδυαδικής αναζήτησης, μπορούμε να βρούμε ένα όνομα στον τηλεφωνι- κότηλεφωνικό κατάλογο, όπου όλα τα ονόματα είναι σε αλφαβητική σειρά. Έτσι μπορούμε να εφαρμόσουμε τη δυαδική αναζήτηση στα στοι-

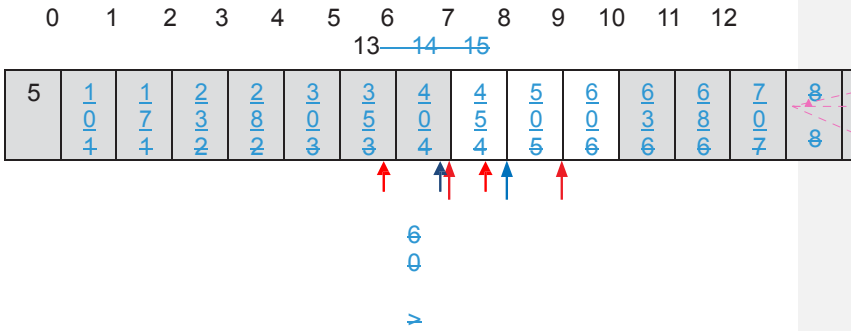
~~σειρά. Έτσι μπορούμε να εφαρμόσουμε τη δυαδική αναζήτηση στα στοιχεία~~ μιας λίστας τα οποία βρίσκονται σε κάποια λογική διάταξη, είτε αυτά είναι αριθμοί είτε αλφαριθμητικά.

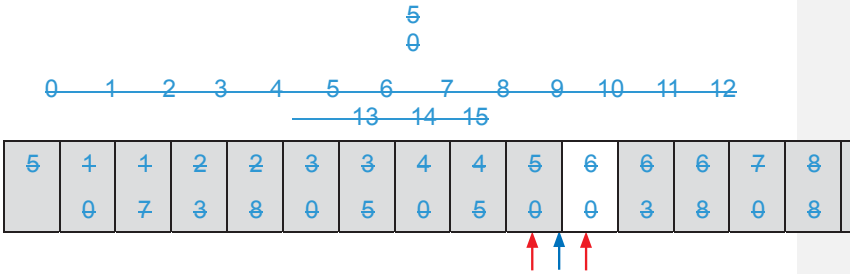
Παρακάτω φαίνεται η εκτέλεση του αλγόριθμου δυαδικής αναζήτησης~~αναζήτησης~~ σε μια λίστα 14 ~~16~~ αριθμών, για την αναζήτηση του αριθμού 45 ~~60~~. Η λογική εδώ είναι ακριβώς ίδια με το παιχνίδι “Βρες τον αριθμό~~αριθμό~~”.











Παρακάτω δίνεται ο αλγόριθμος της δυαδικής αναζήτησης σε Python με τη μορφή μιας συνάρτησης, η οποία δέχεται ως παράμετροπαράμετρο μια λίστα array σε αύξουσα σειρά και το ζητούμενο στοιχείο key.:

Να σημειωθεί ότι ο αλγόριθμος της δυαδικής αναζήτησης μπορεί να εφαρμοστεί μόνο όταν τα στοιχεία είναι διατεταγμένα σε αύξουσα ή φθίνουσα σειρά.

Η επόμενη έκδοση του αλγορίθμου έχει σχεδιαστεί για τα στοιχεία μιας λίστας που είναι ταξινομημένα σε αύξουσα σειρά.


74

~~*Κεφ.5: Κλασικοί αλγόριθμοι II*~~

Αλγόριθμος Δυαδικής αναζήτησης

```
def binarySearch( array, key ) :
    first = 0
    last =
    len(arra
    y) – 1
    found =
    False
    while first <= last and not found :
        mid = (
        first + last )
        // 2 if
        array[ mid
        ] == key :
            f
            o
            u
            n
            d
            =
            T
            r
            u
            e
        elif array[
        mid ] < key
            :
            fir
        st =
        mid +
        1 else
        :
            la
            st
```

=
mi
d
–
1

return found

Η ~~Παρατηρήστε ότι η~~ παραπάνω συνάρτηση επιστρέφει True αν το key υπάρχει στη λίστα array ή False διαφορετικά. Τις περισσότερες φορές όμως δεν θέλουμε μόνο να μάθουμε αν το στοιχείο υπάρχει αλλά και τη θέση στην οποία βρίσκεται.

Ο παρακάτω αλγόριθμος επιστρέφει τη θέση του στοιχείου αν υπάρχει, σε διαφορετική περίπτωση επιστρέφει –1. Η λογική μεταβλητή found έχει αντικατασταθεί από τη μεταβλητή pos η οποία εκτός από τη θέση του στοιχείου παίζει και τον ρόλο της λογικής μεταβλητής (True αν pos >= 0 και False αν pos < 0)

```
def binarySearch( array, key ) :
    first = 0
    last = len(array) – 1
    pos = –1
    while first <= last  and pos == –1  :
```

```
        mid = ( first + last ) / 2
        if  array[ mid ] == key :
            pos = mid
        elif  array[ mid ] < key :
            first = mid + 1
else :
            last = mid – 1
    return  pos
```

Παρατηρήστε ότι οι παραπάνω αλγόριθμοι ισχύουν ~~ισχύει~~ για όλους τους τύπους δεδομένων για τους οποίους ορίζονται~~ισχύουν~~ οι συγκριτικοί τε- λεστές~~τελεστές~~ ==, < . Δηλαδή, μπορούν~~μπορεί~~ να χρησιμοποιηθούν~~κληθεί~~ για ακέραιους~~ακέραιους~~ ή πραγματικούς αριθμούς, αλφαριθμητικά ή ακόμα και σύνθετους~~σύνθετους~~ τύπους για τους οποίους έχουμε ορίσει τους συγκεκριμένους τελεστές. Αυτό το χαρακτηριστικό που είναι γνωστό ως **πολυ- μορφισμός**~~χαρακτηριστικό~~ είναι ένα από τα βασικά πλεονεκτήματα της Python.

Συνοψίζοντας, μπορούμε να πούμε ότι η δυαδική αναζήτηση:

- εκμεταλλεύεται τη διάταξη των στοιχείων ενός συνόλου δεδομένων για τη γρήγορη~~γρή- γορη~~ εύρεση ενός στοιχείου
- χρησιμοποιείται μόνο σε ταξινομημένες συλλογές δεδομένων
- βρίσκει το ζητούμενο πολύ πιο γρήγορα από ότι η σειριακή αναζήτηση.

5.2 Ταξινόμηση Ευθείας ανταλλαγής

Στην προηγούμενη τάξη παρουσιάστηκε ο αλγόριθμος *ταξινόμησης*~~**ταξινόμησης**~~ **με επιλογή** (selection sort). Ένα από τα χαρακτηριστικά αυτού του αλγορίθμου είναι ότι εκτελεί πάντα τον ίδιο αριθμό συγκρίσεων για συλλογές δεδομένων με το ίδιο μέγεθος, ακόμα και για αυτές που είναι ήδη ταξινομημένες. Επειδή η ταξινόμηση είναι μια λειτο- υργία~~λειτουργία~~ που χρησιμοποιείται πολύ συχνά στην Πληροφορική, έχουν αναπτυχθεί διάφοροι αλγόριθμοι ταξινόμησης, κάποιοι από τους~~ οποίους έχουν πολύ καλή απόδοση για ορισμένα σύνολα δεδομένων, όπως για παράδειγμα αυτά που είναι σχεδόν ταξινομημένα. Ένας αλγόριθμος ο οποίος σταματάει, όταν διαπιστώσει ότι τα δεδομένα είναι ήδη ταξινομημένα, είναι ο αλγόριθμος της **ευθείας ανταλλαγής**~~.

οποίους έχουν πολύ καλή απόδοση για ορισμένα σύνολα δεδομέ- νων, όπως για παράδειγμα αυτά που είναι σχεδόν ταξινομημένα.

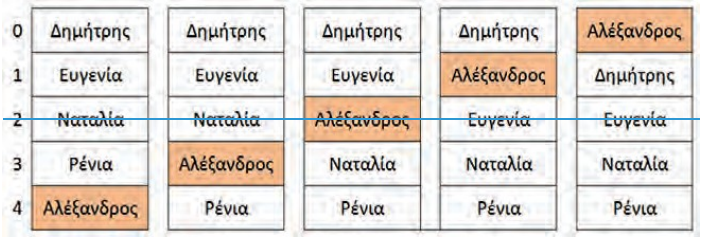
Εισαγωγική Δραστηριότητα

Σε μια πρόβα για την παρέλαση της 28ης Οκτωβρίου, τέσσερις μαθητές του τμήματος~~τμή ματος~~ Β4 έχουν παραταχθεί σε αλφαβητική σειρά. Ενώ περιμένουν να ξεκινήσει η πρόβα, ο Αλέξανδρος φτάνει κα- θυστερημένος~~καθυστερημένος~~ και ξεκινώντας από το τέλος της σειράς~~σει- ράς~~ προσπαθεί να βρει τη θέση του, συγκρίνοντας κάθε φορά το όνομά του με αυ- τό~~αυτό~~ του συμμαθητή ή της συμμαθήτριάς του που βρίσκεται μπροσ- τά~~μπροστά~~ του στη σειρά.

Θέλουμε να σχεδιάσουμε έναν αλγόριθμο ο οποίος θα προωθεί τον Αλέξανδρο με διαδοχικές εναλλαγές προς τα εμπρός, ώστε να, ~~βρεθεί μπροστά από τον Δημήτρη, στην αρχή της λίστας (θέση 0).~~

βρεθεί μπροστά από τον Δημήτρη, στην αρχή της λίστας (θέση 0).

0	Δημήτρης	Δημήτρης	Δημήτρης	Δημήτρης	Αλέξανδρος
1	Ευγενία	Ευγενία	Ευγενία	Αλέξανδρος	Δημήτρης
2	Ναταλία	Ναταλία	Αλέξανδρος	Ευγενία	Ευγενία
3	Ρένια	Αλέξανδρος	Ναταλία	Ναταλία	Ναταλία
4	Αλέξανδρος	Ρένια	Ρένια	Ρένια	Ρένια



Εικόνα 5-1

Αρχικά, ας θεωρήσουμε ότι καταχωρούμε τους παραπάνω μαθη- τέςμαθητές σε μια λίστα students με την παρακάτω εντολή:

```
students = [“Δημήτρης”, “Ευγενία”, “Ναταλία”, “Ρένια”, “Αλέξανδρος”]
```

Αρχικά, ο Αλέξανδρος θα συγκριθεί αλφαβητικά με τη Ρένια και αφού διαπιστωθεί ότι προηγείται αλφαβητικά, θα αλλάξει θέση με τη Ρένια. Η σύγκριση δύο αλφαριθμητικώνδυο αλφαριθ μητικών γίνεται με βάση την λεξικογραφική διάταξη ή όπως λέμε την αλφαβητική σειρά μεταξύ τους. Για παράδειγμα το αλφαριθμητικό “ΔΗΜΗΤΡΗΣ” έπεται του “ΔΗΜΗΤΡΑΔΗ- ΜΗΤΡΑ” αφού “Α” < “Η”.

Αφού διαπιστωθεί ότι τα στοιχεία της λίστας πρέπει να αλλάξουν αμοιβαία θέσεις, μπορεί να χρησιμοποιηθεί μια προσωρινή μεταβ- λητή temp, ή απευθείας η εντολή αντιμετάθεσης μεταβλητών τηςμεταβλητή temp.
Python, όπως θυμάστε από την Β Λυκείου.

Αμοιβαία αλλαγή των τιμών των μεταβλητών a, b	
Με επιπλέον μεταβλητή	**Χωρίς επιπλέον μεταβλητή**
temp = a a = b b = temp	a , b = b , a

Οι εντολές για τη σύγκριση των δυο τελευταίων στοιχείων είναι:

```
if students[4]  < students[3] :
   temp        = students[4]
   students[34] = students[3]
   students[4]
```

Στη συνέχεια, ο Αλέξανδρος θα συγκριθεί με τη Ναταλία και αφού ισχύει “Αλέξανδρος” < “Ναταλία”, θα αλλάξει θέσεις και με τη Να- ταλία. Ομοίως θα συγκριθεί και θα αλλάξει διαδοχικά θέσεις με την
Ευγενία και τον Δημήτρη, ανεβαίνοντας στην θέση 0.

```
   students[3] = temp
```

ή απευθείας η εντολή αντιμετάθεσης μεταβλητών της Python, με χρήση πλειάδων:

```
if students[4] < students[3] :
   students[4] , students[3] = students[3] , students[4]
```

Στη συνέχεια, ο Αλέξανδρος θα συγκριθεί με τη Ναταλία και αφού ισχύει “Αλέξανδρος” < “Ναταλία”, θα αλλάξει θέσεις και με τη Ναταλία. Ομοίως θα συγκριθεί και θα αλλάξει διαδοχικά θέσεις με την Ευγενία και τον Δημήτρη, ανεβαίνοντας στην θέση 0.

```
if students[4] < students[3] :
  students[4] , students[3] = students[3] , students[4]
if students[3] < students[2] :
  students[3] , students[2] = students[2] , students[3]
if students[2] < students[1] :
  students[2] , students[1] = students[1] , students[2]
if students[1] < students[0] :
  students[1] , students[0] = students[0] , students[1]
```

Οι παραπάνω εντολές μπορούν να γραφούν με πιο συνοπτικό τρόπο, με τη χρήση εντολής επανάληψης:

```
for j in range(4, 0, –1):
    if students[j] < students[j–1] :
        students[j], students[j–1] = students[j–1], students[j]
```

Υπενθύμιση: range(4, 0, –1) = [4, 3, 2, 1]

Άρα στην τελευταία επανάληψη το j παίρνει την αμέσως προηγο- ύμενη τιμή του 0, δηλαδή το 1 όπως κατεβαίνουμε, άρα έχουμε j=1 και j–1=0. Να θυμάστε ότι πάντα η τελική τιμή της range αναφέρε- ται στο στοιχείο της λίστας που θα συγκριθεί τελευταίο. Στην πε- ρίπτωση αυτή το στοιχείο είναι το students[0].

Αν εφαρμόσουμε τον ίδιο αλγόριθμο σε μια λίστα **A** με N αριθμούς, τότε παίρνει τη μορφή:

```
y[ j ]
```

	j=5	j=4	j=3	j=2	j=1	
0	21	21	21	21	21	2
1	13	13	13	13	2	21
2	8	8	8	2	13	13
3	5	5	2	8	8	8
4	3	2	5	5	5	5
5	2	3	3	3	3	3

ι αριθμού στην πρώτη θέση (0)

ray = [21, 13, 8, 5, 3, 2], αν

5.

	j=5	j=4	j=3	j=2	j=1	
0	21	21	21	21	21	2
1	13	13	13	13	2	21
2	8	8	8	2	13	13
3	5	5	2	8	8	8
4	3	2	5	5	5	5
5	2	3	3	3	3	3

Εικόνα 5-2

Τι θα συμβεί, αν εκτελέσουμε άλλη μια φορά τις ίδιες εντολές στη λίστα που προέκυψε προέ- κυψε; Αυτή τη φορά θα ανέβει στη δεύτερη θέση ο αμέσως μικρότερος αριθμός, δηλαδή δη- λαδή το 3. Κάθε φορά που θα εκ- τελείται εκτελείται

το συγκεκριμένο τμήμα κώδικα, το αμέσως

μικρότερο στο- ιχείοστοιχείο της λίστας θα ανεβαίνει στη σωστή θέση. Άρα, για να ταξινο- μηθείταξινομηθεί πλήρως η λίστα, πρέπει να εκτελέσουμε το παραπάνω τμή- ματμήμα κώδικα τόσες φορές όσες είναι τα στοιχεία της, δηλαδή N.

Έτσι καταλήγουμε στον παρακάτω αλγόριθμο:

```
N = len( array )
for i in range(1 , N ):          # range(0, N, 1)):
    for j in range(N–1 , 0, –1):
        if array[ j ] < array[ j–1 ] :
            array [ j ] , array [ j–1 ] = array[ j–1 ] , array[ j ]
```

Παρακάτω φαίνεται το Το περιεχόμενο της λίστας αμέσως μετά από για κάθε πέρασμα (εξωτερική επανάληψη) του αλγορίθμου για N=6φαίνεται παρακά- τω:

	i=0	i=1	i=2	i=3	i=4	i=5
0	2	2	2	2	2	2
1	21	3	3	3	3	3
2	13	21	5	5	5	5
3	8	13	21	8	8	8
4	5	8	13	21	13	13
5	3	5	8	13	21	21

	i=1	i=2	i=3	i=4	i=5
0	2				
1	21				
2	13				
3	8				
4	5				
5	3				

~~21~~	~~3~~	~~3~~	~~3~~
~~13~~	~~21~~	~~8~~	~~8~~
~~8~~	~~13~~	~~21~~	~~13~~
~~3~~	~~8~~	~~13~~	~~21~~

Εικόνα 5-3

Με πράσινο είναι τα στοιχεία που ήδη έχουν ανέβει στις τελικές θέσεις τους. Στο τελευταίο πέρασμα (i=5) δεν γίνεται καμιά αλλαγή. Το 21 δεν έχει που να πάει γιατί όλα τα υπόλοιπα στοιχεία έχουν ήδη ανέβει στις θέσεις τους στα προηγούμενα περάσματα. Άρα το 21 είναι ήδη στη σωστή θέση και το $6^{ο}$ πέρασμα για i=5 είναι περιττό. Αυτό μπορούμε να το γενικεύσουμε και στην περίπτωση των N στοιχείων όπου N-1 περάσματα αρκούν για την ταξινόμηση της λίστας (δηλαδή για i=0 μέχρι και i=N-2).

```
N = len( A )
for i in range( N–1 ):                    # range(0, N–1, 1)
    for j in range( N–1 , 0, –1 ):
        if  A[ j ] < A[ j–1 ]  :
            A[ j ] , A[ j–1 ] = A[ j–1 ] , A[ j ]
```

Κατά την εκτέλεση του αλγορίθμου, η λίστα χωρίζεται σε δύο τμή- ματατμήματα: το ταξινομημένο (πράσινο) και αυτό που δεν έχει ταξινομη- θείταξινομηθεί ακόμα. Είναι φανερό λοιπόν ότι, κάθε φορά, το αμέσως μικρό- τερομικρότερο στοιχείο θα ανέβει μέχρι το σημείο που αρχίζει το πράσινοπρά- σινο χρώμα και όχι παραπάνω. Άρα δεν υπάρχει λόγος στην εσωτερική επανάληψη να συγκρίνουμε κάθε φορά μέχρι και το πρώτο στοιχε- ίο, αλλά μέχρι τα δυο πιο πάνω στοιχεία που δεν έχουν ταξινομη-στοιχείο, αλλά μέχρι το στοιχείο στη θέση i–1. Επίσης φαίνεται ότι δεν χρειάζονται Ν (=6) επαναλήψεις, αλλά Ν–1 (=5), όταν τα Ν-1 στοιχεία τοποθετηθούν στις θέσεις τους, με το τελευταίο στοιχείο (21) να βρίσκεται στη μόνη θέση που περισσεύει, δηλαδή την τελευταία.

θεί ακόμα. Αυτά για κάθε πέρασμα είναι:

Πέρασμα	j	Τελευταία Σύγκριση
i=0	1	A[0] < A[1]
i=1	2	A[1] < A[2]
i=2	3	A[2] < A[3]
i=3	4	A[3] < A[4]
i=4	5	A[4] < A[5]

Παρατηρούμε ότι σε κάθε πέρασμα η τελευταία σύγκριση θα είναι για **j = i–1**, άρα η range της εσωτερικής επανάληψης γίνεται : *range(N-1, i, -1)*. Με αυτή την τελευταία βελτίωση ο αλγόριθμος παίρνει την τελική του μορφή:

Αλγόριθμος ταξινόμησης ευθείας ανταλλαγής (Bubble Sort)

```
N = len( Aarray )
for i in range(1, N–1 ):      # range(0, N–1, 1)):
    for j in range(N–1 , i , –1 ):      # μέχρι και i, –1):
        if Aarray[ j ] < array[ j –1 ] :
            array [ j ] < A, array [ j–1 ] := array[ j–1 ] , array[ j ]
                A[ j ] , A[ j–1 ] = A[ j–1 ] , A[ j ]
```

Ο αλγόριθμος αυτός ονομάζεται **αλγόριθμος ταξινόμησης ευθείας~~ευθείας~~ ανταλλαγής** (straight exchange sort), και είναι ευρύτερα γνωσ- τός~~γνωστός~~ ως αλγόριθμος **ταξινόμησης φυσαλίδας** (bubble sort). Αυτό διότι σε κάθε πέρασμα, το αμέσως μικρότερο στοιχείο~~στοιχείο~~ ανεβαίνει, όπως μια φυσαλίδα, στην επιφάνεια του νερού. Ο αλγόριθμος βα- σίζεται στη σύγκριση και αντιμετάθεση ζευγών που δεν ακολουθο- ύν~~ακολουθούν~~ τη διάταξη της ταξινόμησης.

Ο αλγόριθμος ευθείας ανταλλαγής, αν και θεωρείται από τους πιο αργούς αλγορίθμους~~αλγο- ρίθμους~~ ταξινόμησης, έχει ένα πολύ σημαντικό πλεο- νέκτημα~~πλεονέκτημα~~: ότι μπορεί να τροποποιηθεί~~τρο- ποποιηθεί~~, ώστε να τερματίσει μόλις διαπιστώσει ότι η λίστα έχει ταξινομηθεί. Έτσι αποφεύγονται πολ- λές~~πολλές~~ περιττές συγκρίσεις στην περίπτωση λιστών που είναι κατά ένα σημαντικό μέρος ήδη ταξινομημένες. Για παράδειγμα, αν δο- θούν~~δοθούν~~ οι αριθμοί~~: :~~
3, 5, 8, 13, 21, 34, 55, 2, αρκεί μόνο ένα πέρασ- μα~~πέρασμα~~, ώστε να έρθει το 2 στην πρώτη θέση, αφού τα υπόλοιπα στοι- χεία~~στοιχεία~~ είναι ήδη στη σωστή σειρά. Σε αυτή την περίπτωση προσθέ- τουμε~~περίπτωση προσθέτουμε~~ μια λογική μεταβλητή, η οποία θα παραμένει ψευδής (false), όσο η λίστα δεν είναι ταξινομημένη, ενώ αν γίνει αληθής ο αλγό- ριθμος~~αλγόριθμος~~ τερματίζει.

Στην εικόνα 5.4 φαίνεται το περιεχόμενο μιας λίστας επτά (7) α- ριθμών~~αριθμών~~ αναλυτικά για κάθε βήμα του αλγορίθμου ταξινόμησης ευ- θείας~~ευθείας~~ ανταλλαγής. Κάθε στήλη αντικατοπτρίζει~~αντικατοπτρίζει~~ τη λίστα των αριθ- μών~~αριθμών~~ για κάθε τιμή των μετρητών i, j των επαναλήψεων. Με πράσι- νο~~πράσινο~~ χρώμα είναι τα στοιχεία που είναι ήδη ταξινομημένα και με πορτοκαλί αυτά που συγκρίνονται στο συγκεκριμένο βήμα.

1ο πέρασμα για i=0

	j=6	j=5	j=4	j=3	j=2	j=1	
0	60	60	60	60	60	60	20
1	38	38	38	38	38	20	60
2	98	98	98	98	20	38	38
3	54	54	54	20	98	98	98
4	32	32	20	54	54	54	54
5	90	20	32	32	32	32	32
6	20	90	90	90	90	90	90

2ο πέρασμα για i=1

	j=6	j=5	j=4	j=3	j=2	
0	20	20	20	20	20	20
1	60	60	60	60	60	32
2	38	38	38	38	32	60
3	98	98	98	32	38	38
4	54	54	32	98	98	98
5	32	32	54	54	54	54
6	90	90	90	90	90	90

3ο πέρασμα για i=2

	j=6	j=5	j=4	j=3	
0	20	20	20	20	20
1	32	32	32	32	32
2	60	60	60	60	38
3	38	38	38	38	60
4	98	98	54	54	54
5	54	54	98	98	98
6	90	90	90	90	90

4ο πέρασμα για i=3

	j=6	j=5	j=4	
0	20	20	20	20
1	32	32	32	32
2	38	38	38	38
3	60	60	60	54
4	54	54	54	60
5	98	90	90	90
6	90	98	90	98

5ο πέρασμα για i=4

	j=6	j=5	
0	20	20	20
1	32	32	32
2	38	38	38
3	54	54	54
4	60	60	60
5	90	90	90
6	98	98	98

6ο πέρασμα για i=5

	j=6	
0	20	20
1	32	32
2	38	38
3	54	54
4	60	60
5	90	90
6	98	98

Κεφ.5: Κλασσικοί αλγόριθμοι II

1ο βήμα για i=1

	j=6	j=5	j=4	j=3	j=2	j=1
0	60	60	60	60	60	20
1	38	38	38	38	20	60
2	98	98	98	20	38	38
3	54	54	20	98	98	98
4	32	20	54	54	54	54
5	20	32	32	32	32	32
6	90	90	90	90	90	90

2ο βήμα για i=2

	j=6	j=5	j=4	j=3	j=2
0	20	20	20	20	20	20
1	60	60	60	60	60	32
2	38	38	38	38	32	60
3	98	98	98	32	38	38
4	54	54	32	98	98	98
5	32	32	54	54	54	54
6	90	90	90	90	90	90

3ο βήμα για i=3

	j=6	j=5	j=4	j=3
0	20	20	20	20	20
1	32	32	32	32	32
2	60	60	60	60	38
3	38	38	38	38	60
4	98	98	54	54	54
5	54	54	98	98	98
6	90	90	90	90	90

4ο βήμα για i=4

	j=6	j=5	j=4
0	20	20	20	20
1	32	32	32	60
2	38	38	38	38
3	60	60	60	54
4	54	54	54	60
5	98	90	90	90
6	90	98	98	98

5ο βήμα για i=5

	j=6	j=5
0	20	20	20
1	32	32	32
2	38	38	38
3	54	54	54
4	60	60	60
5	90	90	90
6	98	98	98

6ο βήμα για i=6

	j=6
0	20	20
1	32	32
2	38	38
3	54	54
4	60	60
5	90	90
6	98	98

Εικόνα 5-4

Εικόνα 5-4

Παρατηρούμε ότι:

- Στο 6ο βήμα, τα 6 πρώτα στοιχεία είναι στις τελικές θέσεις τους ~~τους~~ οπότε το 7ο (98) που μένει, βρίσκεται υποχρεωτικά στη θέση ~~θέση~~ που περισσεύει, που είναι η τελευταία. Για αυτό το λόγο ενώ έχουμε 7 στοιχεία, η εξωτερική επανάληψη είναι από ~~1~~ ~~μέχρι Ν–1 και όχι από~~ 0 μέχρι και Ν–2. Γι' αυτό δε ~~Ν–1. Έτσι δεν~~ χρειάζονται επτά (7), ~~)~~ αλλά έξι περάσματα ~~περάσματα~~.

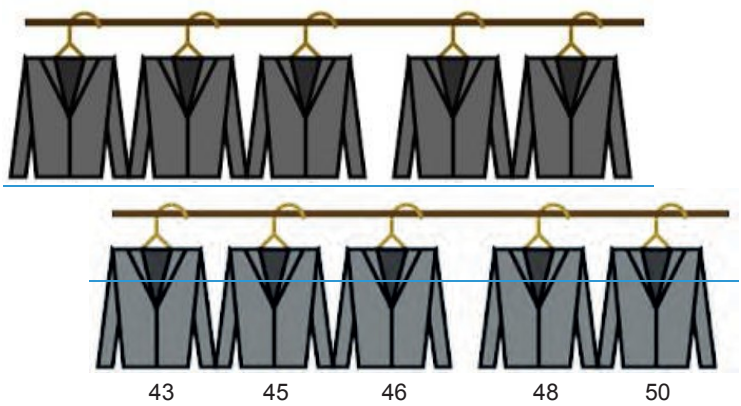
- Μετά το 4ο βήμα, οι αριθμοί είναι ταξινομημένοι, άρα δεν χρειάζονταιχρειάζονται άλλες συγκρίσεις. Τα δυο τελευταία βήματα για αυτή την περίπτωση είναι περιττά. Πώς θα μπορούσαμε να βελτιώσουμεβελτιώσουμε τον αλγόριθμο ευθείας ανταλλαγής, ώστε να σταματάει μόλις διαπιστώσει ότι δε χρειάζονται άλλες συγκρίσεις; Δείτε τη δραστηριότητα “βελτιωμένη φυσαλίδα” στο τέλος του κεφαλαί- ουκεφαλαίου.

5.3 Ταξινόμηση με Εισαγωγή

Σε μια βιοτεχνία ρούχων τα κοστούμια είναι τοποθετημένα σε αύ- ξουσααύξουσα σειρά με βάση το μέγεθός τους. Κάθε φορά που έρχεται ένα νέο κοστούμι ο ράφτης πρέπει να το τοποθετήσει στη σωστή θέ- σηθέση, χωρίς να χαλάσει τη σειρά των κοστουμιών και με προσοχή, για να μην τα τσαλακώσει. Για να τοποθετήσει ο ράφτης το κοστο- ύμι με μέγεθος 47 στο παρακάτω σχήμα, θα πρέπει να κάνει τρεις κινήσεις:

με προσοχή, για να μην τα τσαλακώσει. Για να τοποθετήσει ο ράφτης το κοστούμι με μέγεθος 47 στο παρακάτω σχήμα, θα πρέπει να κάνει τρεις κινήσεις:

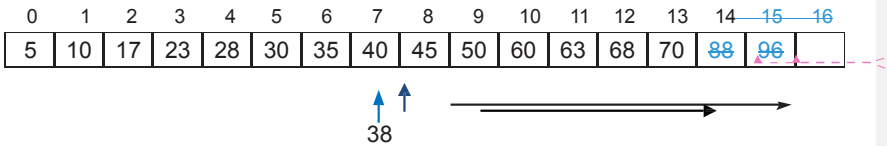
Βήμα 1: Να βρει σε ποιο σημείο πρέπει να μπει το νέο κοστούμι.
Βήμα 2: Να κάνει χώρο γι'αυτό μετακινώντας τα κοστούμια με μεγαλύτερο μέγεθος μια θέση δεξιά.
Βήμα 3: Να τοποθετήσει το κοστούμι στη θέση του.



Εικόνα 5-5

Πολλές φορές, χρειάζεται να έχουμε μια λίστα με στοιχεία η οποία ανανεώνεται συνεχώς, νεχώς και στην οποία τα στοιχεία πρέπει να βρίσκονται σε κάποια διάταξη. Για κάθε νέο στοιχείο που έρχεται στη λίστα, θα πρέπει να γίνουν δυο πράγματα, ώστε να διατηρήσουμε την επιθυμητή διάταξη των στοιχείων της λίστας:

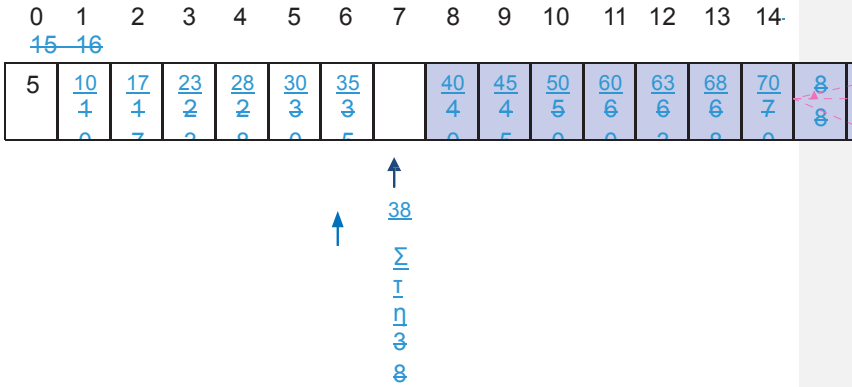
Βήμα 1: Βρίσκουμε τη θέση στην οποία πρέπει να εισαχθεί ο αριθμός.
Βήμα 2: Εισάγουμε τον αριθμό μεταφέροντας όλους τους μεγαλύτερους αριθμούς από αυτόν μία θέση δεξιά (shift).
Για παράδειγμα: ας υποθέσουμε ότι έχουμε την παρακάτω λίστα 16 αριθμών οι οποίοι είναι σε αύξουσα σειρά και θέλουμε να εισάγουμε έναν νέο αριθμό, το 38, έτσι ώστε οι αριθμοί να παραμείνουν ταξινομημένοι. Αυτό μπορεί να γίνει σε δυο βήματα:



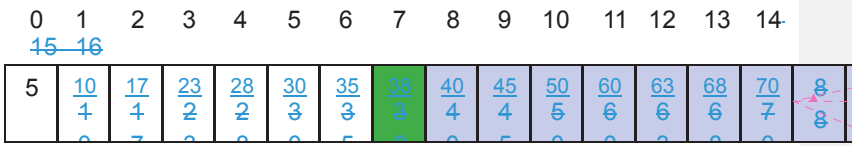
~~*Κεφ.5: Κλασικοί αλγόριθμοι II*~~

Πρώτα γίνεται αναζήτηση στη λίστα, ώστε να βρεθεί η θέση στην οποία θα εισαχθεί το 38. Η θέση αυτή είναι η 7. Για να εισαχθεί το ~~38 στη θέση αυτή, θα πρέπει πρώτα, όλα τα στοιχεία από το 40 και μετά να μετακινηθούν μια θέση δεξιά, για να κάνουν χώρο στο 38.~~

38 στη θέση αυτή, θα πρέπει πρώτα, όλα τα στοιχεία από το 40 και μετά να μετακινηθούν μια θέση δεξιά, για να κάνουν χώρο στο 38.



~~Στην~~ συνέχεια το 38 τοποθετείται στην θέση 7, έτσι ώστε η νέα λίστα~~λίστα~~ να παραμείνει ταξινομημένη.

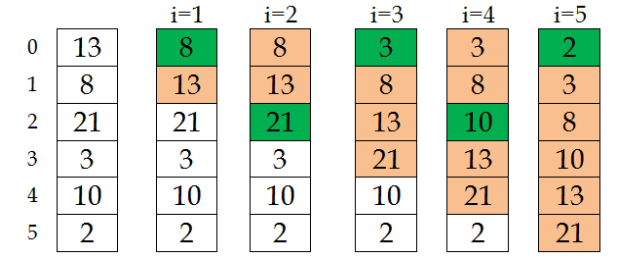


Μπορούμε να χρησιμοποιήσουμε την παραπάνω ιδέα για να ταξινομήσουμε~~ταξινομήσουμε~~ μια λίστα αριθμών. Κάθε φορά θα εισάγουμε και ένα νέο αριθμό στην ήδη ταξινομημένη λίστα, όπως κάναμε με το 38.

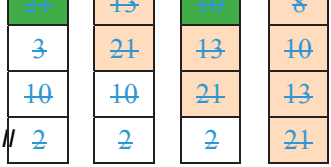
Ο αλγόριθμος ταξινόμησης που προκύπτει λέγεται **αλγόριθμος ταξινόμησης με εισαγωγή** (insertion sort) και είναι γνωστός και ως **ταξινόμηση παρεμβολής**, αφού ο νέος αριθμός παρεμβάλλεται~~παρεμβάλλεται~~ στο ταξινομημένο τμήμα της λίστας.

Στην ταξινόμηση με εισαγωγή μπορούμε να χωρίσουμε τα στοιχεία~~στοιχεία~~ σε 3 ομάδες:

- Αυτά που έχουν ήδη ταξινομηθεί και είναι σε πλήθος ίδια με τον αριθμό των βημάτων του αλγορίθμου μέχρι εκείνη τη στιγ- μή~~στιγμή~~ (πορτοκαλί χρώμα).
- Το στοιχείο που εισάγεται~~θα εισαχθεί~~ (πράσινο χρώμα).
- Τα στοιχεία που δεν έχουν ταξινομηθεί ακόμα.



		i=1	i=2	i=3	i=4	i=5
0	13	8				
1	8	13				
2	1	21				
3	3	3				
4	10	10				
5	2	2				



Εικόνα 5-6

Σε αντίθεση με τον αλγόριθμο της ευθείας ανταλλαγής, τα στοιχεία με πορτοκαλί~~πράσινο~~ είναι ταξινομημένα μεταξύ τους, αλλά δε βρίσκονται εξαρχής στις τελικές θέσεις.

Αλγόριθμος Ταξινόμησης με Εισαγωγή (Insertion Sort)

```
def insertionSort( array ) :
    for i in range(1, len( array ) ) :
        value = array[ i ]
        pos = i
        # αναζήτηση
        while pos > 0 and array[ pos–1 ] > value :
            pos = pos – 1
        # μετακίνηση των στοιχείων μια θέση δεξιά
        for j in range(i-1, pos-1, -1) :
            array[ j+1 ] = array[ j ]
        # το στοιχείο τοποθετείται στη θέση pos.
        array[ pos ] = value
```

Μπορεί να εμφανιστεί ένα μικρότερο στοιχείο και να τοποθετηθεί στην πρώτη θέση σπρώχνοντάς τα όλα μια θέση προς τα κάτω, όπως, για παράδειγμα, συμβαίνει με τον αριθμό 2 στο τελευταίο βήμα του αλγορίθμου.

Αλγόριθμος Ταξινόμησης με Εισαγωγή (Insertion Sort)

```
def insertionSort( array ) :
    for i in range(1, len( array ) ) :
        value = array[ i ]
        pos = i
        # αναζήτηση
        while pos > 0 and array[ pos - 1 ] > value :
            pos = pos - 1
            # μετακίνηση των στοιχείων μια θέση δεξιά
            for j in range(pos, i) :
                array[ j+1 ] = array[ j ]
            # το στοιχείο τοποθετείται στη θέση pos.
            array[ pos ] = value
```

Ο αλγόριθμος μπορεί να γραφτεί πιο συνοπτικά, όπως φαίνεται παρακάτω:

Αλγόριθμος Ταξινόμησης με Εισαγωγή (Insertion Sort) II

```
def insertionSortOpt~~insertionSort~~( array ) :
    for i in range(1, len( array ) ) :
        value = array[ i ]
        j = i
        while j > 0 and array[ j – 1 ] > value :
            array[ j ] = array[ j–1 ]
        j = j
        – 1
        arra
        y[ j ]
        =
        valu
        e
```

Η λειτουργία του αλγορίθμου έχει ως εξής:

Σαρώνουμε το ταξινομημένο τμήμα της λίστας, από δεξιά προς τα αριστερά, μέχρι να συναντήσουμε το πρώτο στοιχείο που είναι μικρότερο αυτού που θέλουμε να εισάγουμε (value=array[i]). Ενώ διατρέχουμε τη λίστα, παράλληλα μετακινούμε και τα στοιχεία μία~~μια~~ θέση προς τα δεξιά *(array[j] = array[j–1])*. Δηλαδή η μετακίνηση και η αναζήτηση γίνονται παράλληλα σε αυτήν την περίπτωση.

Επίσης, μόλις η μεταβλητή pos γίνει 0 η έκφραση pos > 0 είναι ψευδής, άρα όλη η συνθήκη είναι ψευδής, αφού χρησιμοποιείται ο λογικός τελεστής και, ανεξάρτητα από την τιμή της δεύτερης λογι-κής~~λογικής~~ έκφρασης. Σε αυτήν την περίπτωση, η έκφραση A[j–1] > value δε θα αποτιμηθεί.

Παρακάτω φαίνεται το περιεχόμενο μιας λίστας 7 αριθμών, αναλυτικάαναλυτικά για κάθε βήμα του αλγορίθμου ταξινόμησης με εισαγωγή. Κά- θεΚάθε στήλη αντικατοπτρίζει το περιεχόμενοπεριεχόμενο της λίστας για μια μετακί- νησημετακίνηση του στοιχείου προς εισαγωγή. Με πράσινο χρώμα, είναι τα στοιχεία που είναι ήδη ταξινομημένα και με πορτοκαλί, αυτά που συγκρίνονται στο συγκεκριμένο βήμα.

στοιχεία που είναι ήδη ταξινομημένα και με πορτοκαλί, αυτά που συγκρίνονται στο συγκεκριμένο βήμα.

Από την εκτέλεση του αλγορίθμου που φαίνεται στην εικόνα 5-7, παρατηρούμε

1ο βήμα για i=1

j=1

0	60	38
1	38	60
2	98	98
3	54	54
4	32	32
5	90	90
6	20	20

2ο βήμα για i=2

j=2

0	38	38
1	60	60
2	98	98
3	54	54
4	32	32
5	90	90
6	20	20

3ο βήμα για i=3

j=3 j=2 j=1

0	38	38	38	38
1	60	60	54	54
2	98	54	60	60
3	54	98	98	98
4	32	32	32	32
5	90	90	90	90
6	20	20	20	20

4ο βήμα για i=4

j=4 j=3 j=2

0	38	38	38	38
1	60	60	54	54
2	98	54	60	60
3	54	98	98	98
4	32	32	32	32
5	90	90	90	90
6	20	20	20	20

4ο βήμα για i=4

j=4 j=3 j=2 j=1

0	38	38	38	38	32
1	54	54	54	32	38
2	60	60	32	54	54
3	98	32	60	60	60
4	32	98	98	98	98
5	90	90	90	90	90
6	20	20	20	20	20

5ο βήμα για i=5

j=5 j=4

0	32	32	32
1	38	38	38
2	54	54	54
3	60	60	60
4	98	90	90
5	90	98	98
6	20	20	20

6ο βήμα για i=6

j=6 j=5 j=4 j=3 j=2 j=1

0	32	32	32	32	32	32	20
1	38	38	38	38	38	20	32
2	54	54	54	54	20	38	38
3	60	60	60	20	54	54	54
4	90	90	20	60	60	60	60
5	98	20	90	90	90	90	90
6	20	98	98	98	98	98	98

Εικόνα 5-7

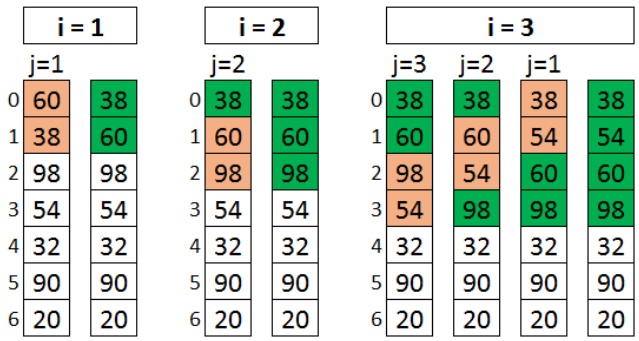
Παρατηρούμε ότι:

- _____

- Το πλήθος των συγκρίσεων και αντιμεταθέσεων σε κάθε βήμα δεν είναι σταθερόσταθε- ρό ή αυξανόμενο, αλλά εξαρτάται από τις θέσε- ιςθέσεις των στοιχείων.

- Ο αλγόριθμος ταξινόμησης με εισαγωγή είναι κατάλληλος για λίστες που είναι σχεδόν ταξινομημένες, σε αντίθεση με τον αλ- γόριθμοαλγόριθμο της επιλογής ή με τον αλγόριθμο ευθείας ανταλλαγής.

- Ο αλγόριθμος ταξινόμησης με εισαγωγή μπορεί να εφαρμοστεί σε λίστες οι οποίες δεν έχουν σταθερό μέγεθος και υλοποιούν συνεχή ροή δεδομένων σε πρακτικές εφαρμογές, όπως είναι π.χ. τα δίκτυα αισθητήρων (sensor networks). Το είδος αυτό των αλγορίθμων που επεξεργάζονται συνεχείς ροές δεδομέ- νωνδεδομένων ονομάζονται **άμεσοι αλγόριθμοι** (online algorithms).



i = 4

	j=4	j=3	j=2	j=1	
0	38	38	38	38	32
1	54	54	54	32	38
2	60	60	32	54	54
3	98	32	60	60	60
4	32	98	98	98	98
5	90	90	90	90	90
6	20	20	20	20	20

i = 5

	j=5	j=4	
0	32	32	32
1	38	38	38
2	54	54	54
3	60	60	60
4	98	90	90
5	90	98	98
6	20	20	20

i = 6

	j=6	j=5	j=4	j=3	j=2	j=1	
0	32	32	32	32	32	32	20
1	38	38	38	38	38	20	32
2	54	54	54	54	20	38	38
3	60	60	60	20	54	54	54
4	90	90	20	60	60	60	60
5	98	20	90	90	90	90	90
6	20	98	98	98	98	98	98

Εικόνα

Δραστηριότητες - Ερωτήσεις - Ασκήσεις κεφαλαίου

5.4 Δραστηριότητες - Άλυτες

Δραστηριότητα 1

Να αντιστοιχήσετε τους παρακάτω αλγόριθμους με τις κατάλληλες λειτουργίες:

Αλγόριθμος	Στοιχειώδ
Ταξινόμηση με επιλογή	Α. Αντιμετάθεση ζευγών
Ταξινόμηση ευθείας αν-	Β. Εισαγωγή στοιχείου σε ταξινο-
Ταξινόμηση με εισαγωγή	Γ. Εύρεση ελαχίστου

Δραστηριότητα 2

Να τροποποιήσετε τον αλγόριθμο της ταξινόμησης με επιλογή, ώστε να ταξινομεί μια λίστα ακεραίων σε φθίνουσα σειρά. Υπάρχει τρόπος να το πετύχετε, χωρίς να κάνετε καμία απολύτως αλλαγή στον κύριο αλγόριθμο που δίνεται στην ενότητα 6.3 του βιβλίου της Β' τάξης; Σε τι οφείλεται αυτό;

Δραστηριότητα 3

Να γράψετε μια συνάρτηση σε Python, η οποία θα δέχεται μια λίσ- τα, θα ελέγχει αν τα στοιχεία της είναι σε αύξουσα σειρά και θα επιστρέφει αντίστοιχα True ή False. Υπόδειξη: Χρησιμοποιήστε μια λογική μεταβλητή.

Δραστηριότητα 4. (Βελτιωμένη φυσαλίδα)

Να αναπτύξετε τη βελτιωμένη έκδοση του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής η οποία τερματίζει, όταν διαπιστώσει ότι η λίστα είναι ταξινομημένη, ώστε να αποφεύγονται περιττές συγκρίσεις.

Υπόδειξη: Χρησιμοποιήστε μια λογική μεταβλητή η οποία θα αλ- λάζει τιμή, αν υπάρχουν τουλάχιστον δύο στοιχεία τα οποία δεδεν βρίσκονται στην επιθυμητή σειρά, καθώς η "φυσαλίδα ανεβαίνει στην επιφάνεια".

Δραστηριότητα 5

Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίστα~~λίστα~~ με λογικές τιμές True/False και θα διαχωρίζει τις τιμές αυτές, τοποθετώντας τα True πριν από τα False.

Δραστηριότητα 6

Να γράψετε ένα πρόγραμμα σε Python το οποίο θα δέχεται μια λίστα με λογικές τιμές True/False και στη συνέχεια θα καλεί την συνάρτηση του προηγούμενου ερωτήματος, ώστε να τοποθετηθο~~ερω- τήματος, ώστε να τοποθετηθούν τα True πριν από τα False. Στη συνέχεια θα τοποθετεί τις τιμές αυτές εναλλάξ, δηλαδή True, False, True, False, κλπ, εκτελώντας τις λιγότερες δυνατές συγκρίσεις.~~

ύν τα True πριν από τα False. Στη συνέχεια θα τοποθετεί τις τιμές αυτές εναλλάξ, δηλαδή True, False, True, False, κ.λπ.

Δραστηριότητα 7

Ας υποθέσουμε ότι σας δίνεται μια λίστα στην Python η οποία πε- ριέχειπεριέχει λογικές τιμές True/False εναλλάξ. Επίσης, το πλήθος των True είναι ίσο με το πλήθος των False. Να γράψετε αλγόριθμο, σε Python, ο οποίος δεδομένης της παραπάνω δομής της λίστας, θα τοποθετεί τα True πριν από τα False. εκτελώντας, τις λιγότερες δυνατές μετακινήσεις. Δεν επιτρέπεται να κάνετε καμία σύγκριση ούτε να χρησιμοποιήσετε τη δομή if. Θεωρήστε ότι το πρώτο στοιχείο της λίστας έχει την τιμή True.

Δραστηριότητα 8

Το πρόβλημα της ολλανδικής σημαίας αναφέρεται στην αναδιάτα- ξηαναδιάταξη μιας λίστας γραμμάτων, η οποία περιέχει μόνο τους χαρακτή- ρεςχαρακτήρες R, W, B. (Red, White, Blue), έτσι ώστε όλα τα R να βρίσκονται πριν από τα W και όλα τα W να βρίσκονται πριν από B. Να τρο- ποποιήσετετροποποιήσετε έναν από τους αλγορίθμους ταξινόμησης που παρου- σιάστηκανπαρουσι- άστηκαν σε αυτήν την ενότητα, ώστε να επιλύει αυτό το πρόβ- λημαπρόβλημα.

Δραστηριότητα 9

Να γράψετε μια συνάρτηση σε Python η οποία διαβάζει αριθμούς από το χρήστη μέχρι να δοθεί η τιμή None, τους οποίους τοποθε- τείτοποθετεί σε μια λίστα σε φθίνουσα σειρά, την οποία και επιστρέφει. Κάθε φορά που διαβάζει έναν νέο αριθμό τον τοποθετεί στη σωστή θέ- σηθέση στην ήδη ταξινομημένη λίστα, ώστε να διατηρείται η φθίνουσα διάταξη των στοιχείων της λίσταςλί- στας. Ποιον αλγόριθμο ταξινόμησης σας θυμίζει η παραπάνω λειτουργία; Σε τι διαφέρειδια- φέρει η συνάρτηση που θα αναπτύξετε από τον αλγόριθμο αυτόν;

~~*Κεφ.5: Κλασικοί αλγόριθμοι II*~~

Δραστηριότητα 10

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει από το χρήστη δύο~~δυο~~ λίστες αριθμών~~αριθμών~~ Α και Β και θα ταξινομεί σε αύξουσα σειρά τους αριθμούς της λίστας Α. Στη συνέχεια θα εμφανίζει πόσοι από τους αριθμούς της λίστας Β εμφανίζονται στην λίστα Α.

Υπόδειξη: Να θεωρήσετε ότι οι αριθμοί της λίστας Β είναι όλοι διαφορετικοί~~διαφορετικοί~~ μεταξύ τους. Επίσης, να εκμεταλλευτείτε το γεγονός ότι τα στοιχεία της λίστας η λίστα~~~~ Α είναι ταξινομημένα~~ταξινομημένη~~ σε αύξουσα σειρά.

Η εισαγωγή των

Δραστηριότητα 11

Δίνονται οι παρακάτω λίστες αριθμών.

55	34	21	13	8	5	3	2	1	1

4	6	8	16	28	32	64	2	1	1

28	28	28	28	28	28	28	28	28	28

28	28	28	28	28	6	6	6	6	6

Να απαντήσετε στα παρακάτω ερωτήματα:

1. Ποιος αλγόριθμος είναι κατάλληλος για την ταξινόμηση κάθε λίστα σταματάει όταν δοθεί η τιμή None.λίστας σε αύξου- σα σειρά και γιατί;
2. Ποιος θα κάνει τις λιγότερες συγκρίσεις και ποιος τις λιγότερες αντιμεταθέσεις στοιχείων;
3. Ποιον ή ποιους αλγόριθμους ταξινόμησης θα επιλέγατε αν θέλατε να ταξινομήσετε μόνο τα 4 μικρότερα στοιχεία κάθε λίστας, χωρίς να ταξινομήσετε όλη τη λίστα;

5.5 Ερωτήσεις - Ασκήσεις

1. Να περιγράψετε τη λειτουργία του αλγορίθμου της δυαδι- κήςδυαδικής αναζήτησης, δίνοντας και ένα παράδειγμα εκτέλεσής του.
2. Να περιγράψετε τη λειτουργία του αλγορίθμου της ταξινό- μησης-ταξινόμησης- ευθείας ανταλλαγής και να δώσετε ένα παράδειγμα εκτέλεσής του.
3. Να περιγράψετε τη λειτουργία του αλγορίθμου της ταξινό- μησηςταξινόμησης με εισαγωγή και να δώσετε ένα παράδειγμα εκτέλε- σήςεκτέλεσής του.
4. Αν έχουμε συνεχή ροή δεδομένων, αλλά θέλουμε τα δεδο- μέναδεδομένα μας να είναι μονίμως ταξινομημένα, ποιον αλγόριθμο θα χρησιμοποιήσουμε και γιατί;
5. Ποια είναι η απαραίτητη προϋπόθεση για να χρησιμοποι- ήσουμεχρησιμοποιήσουμε τη δυαδική αναζήτηση;

Σύνοψη

Μια μέθοδος για την αναζήτηση ενός στοιχείου σε μια λίστα είναι η σειριακή αναζήτησηαναζή- τηση. Στην περίπτωση που η λίστα είναι ταξινομη- μένηταξινομημένη, χρησιμοποιείται η δυαδική αναζήτηση η οποία εκμεταλλεύε- ταιεκμεταλλεύεται τη διάταξη των στοιχείων, για να βρει το ζητούμενοζή- τούμενο στοιχείο πιο γρήγορα. Για να χρησιμοποιηθεί η δυαδική αναζήτηση σε μη ταξι- νομημέναταξινομημένα δεδομένα πρέπει προφανώς πρώτα αυτά να ταξινομη- θούνταξινομηθούν.

Σε αυτό το κεφάλαιο παρουσιάστηκαν δύοδυο αλγόριθμοι ταξινόμη- σηςταξινόμησης. Ο αλγόριθμος της ευθείας ανταλλαγής που βασίζεται στη σύγκριση γειτονικών ζευγών της λίστας και ο αλγόριθμος ταξινό- μησηςταξινόμησης, που με εισαγωγή ταξινομεί μια λίστα βαθμιαία, εισάγοντας- σάγοντας κάθε φορά το επόμενο στοιχείο, έτσι ώστε η λίστα να παραμένει ταξινομημένηταξινο- μημένη.

6

Διαχείριση αρχείων

6. Διαχείριση Αρχείων

Εισαγωγή

Στο κεφάλαιο αυτό θα αναπτυχθούν οι τρόποι δημιουργίας και χει- ρισμούχειρισμού αρχείων. Επίσης, θα παρουσιαστούν οι βασικές λειτουργί- εςλειτουργίες, όπως το άνοιγμα, κλείσιμο ενός αρχείου, το διάβασμα και γρά- ψιμογράψιμο μιας εγγραφής, μέσω της γλώσσας προγραμματισμού Py- thonπρογραμμα- τισμού Python.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- δημιουργούμε αρχεία
- χειριζόμαστε απλά αρχεία, όπως για παράδειγμα αρχεία κειμένου, μέσω βασικώνβα- σικών εντολών (open, close, read, write) που χρησιμοποιεί η γλώσσα προγραμματισμούπρο- γραμματισμού Python
- εκτελούμε πρόσθετες λειτουργίες σε αρχεία, όπως η αναζήτηση.

Λέξεις κλειδιά

Αρχείο κειμένου, άνοιγμα αρχείου, ανάγνωση αρχείου, εγγραφή σε αρχείο, κλείσιμο αρχείου, διαδρομή.

Διδακτικές Ενότητες

6.1 Εισαγωγή - δημιουργία, άνοιγμα, κλείσιμο αρχείων

Η δυνατότητα να δημιουργούμε, να διαβάζουμε και να γράφουμε σε αρχεία αποτελεί βασική εργασία σε πολλά προγράμματα και παρέχεται από όλες τις γλώσσες προγραμματισμού.προ- γραμματισμού. Έτσι, στο κε- φάλαιοκεφάλαιο αυτό θα ασχοληθούμε με τη διαχείριση αρχείων στη γλώσσα προγραμματισμού Python.

Τα περισσότερα προγράμματα που έχουμε δει μέχρι τώρα μπο- ρούνμπορούν να χαρακτηριστούνχαρακτηρι- στούν ως *προσωρινά*. Αυτό, με την έννοια ότι τρέχουν για ένα μικρό χρονικό διάστημαδιάστη- μα και παράγουν κάποια έ- ξοδοέξοδο, με τα δεδομένα τους (ως είσοδος από το πληκτρολόγιοπληκτρο- λόγιο ή ως έξοδος στην οθόνη), να χάνονται. Αυτό συμβαίνει, διότι τα δεδομέ- ναδεδομένα ήταν αποθηκευμένα προσωρινά στην κύρια μνήμη του υπολο- γιστήυπολογιστή, οπότε διαρκούσαν μόνο κατά την εκτέλεση του προγράμμα- τοςπρογράμματος, λογική που έχει ως αποτέλεσμα, να απαιτείται να δίνουμε τα δεδομένα από την αρχή σε κάθε εκτέλεση. Σε πολλές όμως περιπ- τώσειςπεριπτώσεις θέλουμε τα δεδομένα να μη χάνονται. Θέλουμε να διαβά- ζουμε δεδομέναδιαβάζουμε δεδο-

~~μένα~~ από ένα αρχείο του υπολογιστή στο οποίο βρίσ- κονταιβρίσκονται~~βρίσκονται~~ αποθηκευμένα και να γράφουμε ένα αποτέλεσμα στο ίδιο ή εναλλακτικά σε άλλο αρχείο. Αυτή η διεργασία ανάγνωσης και εγ- γραφήςεγγραφής, ονομάζεται Είσοδος/Έξοδος Αρχείου και στην Python υ- λοποιείταιυλοποιείται~~υλοποιείται~~ μέσω ενσωματωμένων συναρτήσεων (μεθόδων).

Γενικά μπορούμε να πούμε ότι χρησιμοποιούμε δύο τύπους αρχε- ίωναρχείων~~αρχείων~~: αυτά που περιέχουνπε- ριέχουν~~πε- ριέχουν~~ το πρόγραμμα που θα εκτελέσουμε και εκείνα που περιέχουν τα δεδομένα, τα οποία, όταν εκτελεστεί το πρόγραμμα, τα διαβάζει και πιθανά τα ενημερώνει.

Ως αρχεία δεδομένων χρησιμοποιούνται συνήθως αρχεία κειμένο- υκειμένου~~κειμένου~~. Ένα **αρχείο κειμένου** είναι ένα αρχείο το οποίο περιέχει μια ακολουθία χαρακτήρων και βρίσκεταιβρίσκε- ται~~βρίσκε- ται~~ αποθηκευμένο σε ένα μέσο μόνιμης αποθήκευσης, όπως ο σκληρός δίσκος. Στα περιεχόμενα ενός τέτοιου αρχείου μπορούμε να έχουμε πρόσβαση μέσω ενός προγράμματοςπρο- γράμματος~~προ- γράμματος~~ που περιέχει τις κατάλληλες εντολές της γλώσσας Python. Ένα αρχείο για να χρησιμοποιηθεί, πρέπει να το ανοίξου- μεανοίξουμε~~ανοίξουμε~~ με την ενσωματωμένη συνάρτηση open() και στο τέλος να κλεί- σεικλείσει~~κλείσει~~ με τη συνάρτηση close().

Η συνάρτηση open() μας επιστρέφει ένα αντικείμενο του αρχείου και μπορούμε να το χρησιμοποιήσουμε για να εκτελέσουμε διάφο- ρεςδιάφορες~~διάφορες~~ λειτουργίες σε αυτό. Η συνάρτησησυ- νάρτηση~~συ- νάρτηση~~ open() είναι ενσωματωμένη στην Python, δε χρειάζεται να φορτώσουμε κάποια βιβλιοθήκη, ενώ η σύνταξή της είναι:

open (“όνομα_αρχείου”, “τρόπος προσπέλασης”)

Παρατηρούμε ότι η συνάρτηση open() δέχεται δύο ~~δυο~~ ορίσματα: Το πρώτο είναι το όνομα του αρχείου, με το οποίο το αναγνωρίζει το λειτουργικό σύστημα. Το δεύτερο είναι ένα ειδικό σύμβολο (ση- μαίασημαία~~σημαία~~ - flag) που καθορίζει τον τρόπο προσπέλασης του αρχείου, του οποίου οι επιτρεπόμενες τιμές της παραμέτρου για τον τρόπο προσπέλασηςπροσπέ- λασης~~προσπέ- λασης~~ φαίνονται στον παρακάτω πίνακα 6.1. Αν δε χρησι- μοποιηθείχρησιμοποιηθεί~~χρησιμοποιηθεί~~ το δεύτερο αυτό όρισμα, τότε θεωρείται εξ ορισμού (προεπιλογή) ότι είναι το “r”.

Παράδειγμα

```
>>> fin = open('words.txt','w')
>>> print fin
```

Στην πρώτη γραμμή ανοίγουμε το αρχείο κειμένου “words.txt” και αποδίδουμε το αποτέλεσμα στον περιγραφέα αρχείου fin. Στη δεύτερη γραμμή τυπώνουμε τα χαρακτηριστικά του περιγραφέα fin. Όλες οι λειτουργίες που πρόκειται να εκτελεστούν στο αρχείο words.txt θα εκτελεστούν μέσω του περιγραφέα fin.

Πίνακας 6-1. Ορίσματα τρόπου προσπέλασης του αρχείου

Ορίσματα τρόπου προσπέλασης του αρχείου	**Λειτουργία**
“r”	Ανάγνωση
“w”	Εγγραφή (διαγραφή προηγούμενων περιεχομένων, αν υπάρχουν)
“a”	Προσθήκη (append) (διατήρηση προηγούμενων περιεχομένων)
“r+”	Άνοιγμα αρχείου και για ανάγνωση και για εγγραφή
“b”	Αρχείο δυαδικής μορφής

Μπορούμε να **δημιουργήσουμε** ένα αρχείο δεδομένων είτε χρησιμοποιώντας ένα συντάκτη (editor), όπως το notepad είτε τη συνάρτηση open στο περιβάλλον της γλώσσας, με το όρισμα **“w”**, όπως για παράδειγμα: open(“words.txt”, “w”). Η συνάρτηση αυτή, αν δεν υπάρχει το αρχείο words.txt, τότε το δημιουργεί, ενώ αν αυτό υπάρχει, τότε θα έχουμε ως αποτέλεσμα να χαθούν τα περι- εχόμενα του. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση με το όρισμα **“a”**, για παράδειγμα open(“words.txt”, “a”), η οποία, αν δεν υπάρχει το αρχείο, το

δημιουργεί, ενώ, αν ~~υπάρχει, το ανοίγει σε κατάσταση προσθήκης δεδομένων στο τέλος του.~~

υπάρχει, το ανοίγει σε κατάσταση προσθήκης δεδομένων στο τέ- λος του.*Κεφ.6: Διαχείριση Αρχείων*

Η γλώσσα Python διαθέτει μεθόδους για να επεξεργαστούμε τα αρχεία. Μπορούμε να ανοίξουμε και να χρησιμοποιήσουμε αρχεία χρησιμοποιώντας τις αντίστοιχες μεθόδους για ανάγνωση **read()** ή **readline()** και για εγγραφή **write()**, στο αρχείο. Η δυνατότητα να διαβάζουμε ή να γράφουμε στο αρχείο εξαρτάται από τον τρόπο προσπέλασης που έχουμε καθορίσει στη συνάρτηση open() κατά το άνοιγμά του.

Όταν ολοκληρώσουμε τις λειτουργίες που θέλουμε να εκτελεστούν στο αρχείο, καλούμεκα- λούμε απαραίτητα τη συνάρτηση **close()**. Αυτό δη- λώνειδηλώνει ότι τελειώσαμε με τη χρήση του και οδηγεί το Λειτουργικό Σύστημα στην αποθήκευση δεδομένων που βρίσκονταιβρίσκο- νται ακόμη στη μνήμη και την περάτωση των δικών του διεργασιών με το αρχείο. Η αντίστοιχη εντολή είναι:

>>> fin.close()

Η close κλείνει το αρχείο, παρόμοια με την επιλογή File→Save στον επεξεργαστή κειμένου.

Για να ελέγξουμε, αν ένα αρχείο όντως έκλεισε, πρέπει να χρησι- μοποιήσουμεχρησιμοποιήσουμε την ιδιότητα (attribute) closed, η οποία επιστρέφει την τιμή True ή False, ανάλογα, αν είναι κλειστό ή ανοιχτό το αρ- χείοαρχείο.

Παράδειγμα

```
>>> fin = open("workfile.txt",
"w")("workfile.txt")
>>> fin.closed
#-False
>>> fin.close()
>>> fin.closed
#-True
```

6.2 Ανάγνωση και εγγραφή σε αρχείο

Στην ενότητα αυτή θα χρησιμοποιήσουμε για τα παραδείγματα ένα αρχείο, το words.txt, το οποίο έχει ως περιεχόμενο το:

This is line 1

This is line 2

This is line 3

Εγγραφή σε αρχείο

Υπενθυμίζουμε ότι για να γράψουμε σε ένα αρχείο, πρέπει πρώτα να το ανοίξουμε με το κατάλληλο όρισμα. Έτσι, με τη χρήση του ορίσματος 'w''w', θα διαγραφούν τα υπάρχοντα περιεχόμενα του αρ- χείουαρχείου αν αυτό υπάρχει, ενώ με τη χρήση του 'a' τυχόν υπάρχοντα περιεχόμενα θα διατηρηθούν. Για την εγγραφή χρησιμοποιούμε τη μέθοδο **write**() και ως όρισμα τη συμβολοσειρά που θέλουμε να
εισάγουμε στο αρχείο. Για παράδειγμα:

```
>>> fin = open(¨words.txt¨, “a”)
>>> fin.write(“This is line 4\n”)
```

Το αρχείο words.txt έχει τώρα τη μορφή:

This is line 1

This is line 2

This is line 3

This is line 4

Σε ένα αρχείο κειμένου, αν θέλουμε να υπάρχουν γραμμές και όχι συνεχόμενοι χαρακτήρεςχα- ρακτήρες, πρέπει να σημειώνουμε την αλλαγή γραμμής με το χαρακτήρα **“\n”**.

Το όρισμα της μεθόδου write πρέπει να είναι **συμβολοσειρά**. Αν θέλουμε να εισάγουμεεισά- γουμε κάτι διαφορετικό, όπως για παράδειγμα έ- νανέναν αριθμό, ο ευκολότερος τρόπος είναι, να μετατρέψουμε τον
αριθμό σε συμβολοσειρά, με χρήση της συνάρτησης str.

```
>>> x = 52
>>> keimeno.write(str(x))
```

Ανάγνωση περιεχομένων ενός αρχείου

Οι πιο διαδεδομένες μέθοδοι για διάβασμα των περιεχομένων ε- νόςενός αρχείου, είναι η **read()** και η **readline()**, όπου διαβάζει ένα πλήθος χαρακτήρων από την αρχή του αρχείου.

Σύνταξη: fileObject_despriptor.read([count]);

Εδώ, η παράμετρος count καθορίζει τον αριθμό των χαρακτήρων που θα διαβασθούνδιαβα- σθούν από το αρχείο, ξεκινώντας την ανάγνωση από την αρχή του μέχρι το πλήθοςπλή- θος των χαρακτήρων που ορίζονται με την count. Στην περίπτωση που η παράμετροςπαράμε-

τρος count λείπει, τότε γίνεται ανάγνωση μέχρι να διαβαστεί η ένδειξη τέλους του αρχείου.

Παράδειγμα 1: Ανάγνωση και εμφάνιση ενός μόνο χαρακτήρα

```
>>> fin = open('words.txt', 'r')
>>> print fin.read(1)
T
```

Παράδειγμα 2: Διάβασμα και εμφάνιση των επόμενων 13 χαρακτήρωνχαρακτήρων

```
>>> print fin.read(13)
his is line 1
```

Παράδειγμα 3: Εμφάνιση ολόκληρου του αρχείου

```
>>> print fin.read()
This is line 1
This is line 2
This is line 3
```

Αν η εντολή print fin.read() εκτελεστεί, αφού έχουν προηγηθεί οι προηγούμενες δύο εντολές, τότε αυτή θα επιστρέψει το υπόλοιπο τμήμα του αρχείου, δηλαδή τις δύο τελευταίες γραμμές.

Η **readline** διαβάζει μια γραμμή του αρχείου, δηλαδή διαβάζει δια- δοχικούς χαρακτήρεςδιαδοχικούς χαρα- κτήρες από ένα αρχείο μέχρι να συναντήσει το χαρακτήραχαρακτήρα νέας γραμμής και επιστρέφειεπι- στρέφει το αποτέλεσμα:

Παράδειγμα

```
>>> fin.close()
>>> fin = open("words.txt")
>>> print fin.readline()
This is line 1
```

Το fin καταγράφει τη θέση που βρίσκεται μέσα στο αρχείο και έτσι αν ξανακαλέσουμεξανακαλέσου- με τη readline θα πάρουμε την επόμενη γραμμή:

```
>>> print fin.readline()
This is line 2
```

Η συνάρτηση close() αναλαμβάνει να κλείσει το αρχείο και να απελευθερώσει έτσι πόρους του συστήματος.

Παράδειγμα

Για ανάγνωση αλλά και εγγραφή, κατά γραμμές, από ένα αρχείο μπορούμε να σαρώσουμε με μια δομή επανάληψης το αρχείο. Για παράδειγμα, για να τυπώσουμε τις γραμμές του αρχείου γράφουμε:

```
>>> for line in fin:
  print line,
```

Εντοπισμός θέσης στο αρχείο

Η μέθοδος **fin.tell()** επιστρέφει έναν ακέραιο που περιέχει την τρέχουσα θέση στο αρχείο, υπολογισμένη σε χαρακτήρες (bytes) από την αρχή του αρχείου. Με άλλα λόγια, η επόμενη ανάγνωση ή εγγραφή θα γίνει σε εκείνη τη θέση του αρχείου.

Για να αλλάξουμε την τρέχουσα θέση του αρχείου, μπορούμε να χρησιμοποιήσουμε την **fin.seek() (offset[, from_what])**. Η θέση υπολογίζεται προσθέτοντας offset (πλήθος bytes) σε ένα σημείο αναφοράς, το οποίο επιλέγεται από το from_what όρισμα. Αν το from_what έχει τιμή 0, μετρά από την αρχή του αρχείου, αν έχει 1, χρησιμοποιεί την τρέχουσα θέση του αρχείου και αν έχει 2, χρησιμοποιεί το τέλος του αρχείου.

```
>>> fin = open("workfile", "r+")
>>> fin.write("0123456789abcdef")
>>> fin.seek(5) # πηγαίνει στο 6ο byte στο αρχείο
>>> fin.read(1)
    5
'5'
>>> fin.seek(-3, 2) # πηγαίνει στο 3ο byte πριν το τέλος
>>> fin.read(1)
    d

'd'
```

6.3 Πρόσθετες λειτουργίες σε αρχεία

Όταν ένα αρχείο έχει ανοίξει, μπορούμε να ανακτήσουμε διάφορες χρήσιμες πληροφορίεςπλη- ροφορίες σχετικά με αυτό. Ακολουθεί μια λίστα με όλα τα σχετικά χαρακτηριστικά:

όλα τα σχετικά χαρακτηριστικά:

Χαρακτηριστικό	Περιγραφή
file.closed	Επιστρέφει true, αν το αρχείο είναι κλειστό, false, σε διαφορετική περίπτωσηπερί- πτωση
file.mode	Επιστρέφει τον τρόπο προσπέλασης στο αρ- χείοαρχείο που έχουμε ανοίξει
file.name	Επιστρέφει το όνομα του αρχείου

Παράδειγμα

```
fin= open("foo.txt", "w")("foo.txt", "wb")
print "Όνομα"Όνομα του αρχείου: ",", fin.name
print "Κλειστό: ","Κλειστό: ", fin.closed
print "Τρόπος"Τρόπος προσπέλασης: ",",
fin.mode Αυτό παράγει το ακόλουθο
αποτέλεσμα: Όνομα του αρχείου: foo.txt
Κλειστό: False
Τρόπος προσπέλασης: wwb
```

Όνομα αρχείου και η θέση του στο μέσο

Τα αρχεία αποθηκεύονται στο φυσικό μέσο από το Λειτουργικό Σύστημα (ΛΣ) οργανωμέναοργα- νωμένα σε καταλόγους, που είναι γνωστοί και ως φάκελοι-folders.

Κάθε δε αρχείο διαθέτει από το ΛΣ *μια μοναδική διαδρομή* του τύπου τύπου:

Φάκελος/φάκελος\υποφάκελος/.. ./... .\όνομα_αρχείου.

Για την Python, αυτό αποτελεί μια συμβολοσειρά και προσδιορίζει το κάθε αρχείο. Μια τέτοια διαδρομή, η οποία ξεκινάει από τον τρέχοντα κατάλογο, καλείται *σχετική*, ενώ αυτή η οποία ξεκινάει από τον ανώτατο κατάλογο στο σύστημα αρχείων καλείται απόλυτη διαδρομή. Όπως κάθε περιβάλλον, έτσι και η Python, έχει έναν προεπιλεγμένο κατάλογο, για αναζήτηση αρχείων, που ονομάζεται τρέχων φάκελος. Έτσι, για παράδειγμα, όταν ανοίγουμε ένα αρχείο για διάβασμα, τότε η Python το αναζητά στον τρέχοντα κατάλογο.

Η **μονάδα λογισμικού os** (os: από το operating system) παρέχει συναρτήσεις, για να μπορούμε να δουλέψουμε με αρχεία και καταλόγους. Για να χρησιμοποιήσουμε αυτό το άρθρωμα (module) απαιτείται πρώτα να το εισάγουμε και έπειτα να καλέσουμε όλες τις σχετικές συναρτήσεις.

```
>>> import os
```

Μετονομασία και διαγραφή αρχείων. Η μέθοδος **rename**()

Η **rename**() αποτελεί μέθοδο της Python για μετονομασία αρχείου και δέχεται δύο ορίσματα, ένα για το τρέχον και ένα για το νέο όνομα του αρχείου.

Σύνταξη: os.rename(current_file_name, new_file_name)

Παράδειγμα

```
import os
# Μετονομασία ενός αρχείου από test1.txt σε test2.txt
os.rename( "test1.txt", "test2.txt" )
```

Η μέθοδος **remove()**

Αποτελεί μέθοδο για τη διαγραφή αρχείων και δέχεται ως όρισμα το όνομα του αρχείου προς διαγραφή.

Σύνταξη: os.remove(file_name)

Παράδειγμα

```
import os

os.remove("text2.txt")   #
Διαγραφή αρχείου test2.txt-
os.remove("text2.txt")
```

Διαχείριση καταλόγων

Η Python χειρίζεται τους καταλόγους (φακέλους) με τη βοήθεια του os module παρέχοντας παρέχοντας αρκετές μεθόδους, όπως για τη δημιουργία δημιουργία, διαγραφή και την αλλαγή καταλόγων.

Η μέθοδος **mkdir**()

Δημιουργεί (υπο)καταλόγους στον τρέχοντα κατάλογο και δέχεται ως όρισμα το όνομα του καταλόγου που θα δημιουργηθεί.

Σύνταξη:
os.mkdir("newdir")("newdir")

Παράδειγμα

```
import os

os.mkdir("test")(
"test")
```

Η μέθοδος chdir(), αλλάζει τον τρέχοντα κατάλογο.

Σύνταξη: os.chdir("newdir")("newdir")

Παράδειγμα

```
import os

os.chdir("/("/home/newdir
")newdir")
```

Η μέθοδος **getcwd():** Επιστρέφει τον τρέχοντα κατάλογο εργασί- ας. εργασίας. Σύνταξη: os.getcwd()

Παράδειγμα

```
import os

os.getcwd()

'C:\\Python27'
```

Η μέθοδος **rmdir():** Διαγράφει το φάκελο που δίνουμε ως όρισμα. Πριν διαγραφεί ο φάκελος, όλα τα περιεχόμενά του διαγράφονται.
Σύνταξη: os.rmdir('dirname')

Παράδειγμα

```
import os
os.rmdir( "/tmp/test" )
```

Δραστηριότητες - Ερωτήσεις - Ασκήσεις κεφαλαίου

6.4 Δραστηριότητες

Δραστηριότητα 1

Ας πειραματιστούμε γράφοντας κώδικα σε Python, ο οποίος διαβάζει το αρχείο “input_file.txt” και το ξαναγράφει στο νέο αρχείο “output_file.txt”, όπου πριν από κάθε γραμμή θα προσθέτει τον αύξοντα αριθμό της. Τα αρχεία θεωρούμε ότι βρίσκονται στον τρέχοντα κατάλογο “TxtFiles”.

αύξοντα αριθμό της. Τα αρχεία θεωρούμε ότι βρίσκονται στον τρέ- χοντα κατάλογο “TxtFiles”.

```
inputfile =
open("TxtFiles("TxtFiles/input_file.txt","r")txt","r"
) outputFile =
open("TxtFiles("TxtFiles/output_file.txt","w")txt",
"w") linecounter = 1

for line in inputfile:
        outputFile.write(str(linecounter)+". "+)+".
        "+line) linecounter = linecounter + 1

inputfile.close()
outputFile.close()
```

Δραστηριότητα 2

Υποθέτουμε ότι έχουμε ένα πρόγραμμα σε Python που έχει απο- θηκευθείαποθηκευθεί στο αρχείοαρ- χείο με όνομα demo.py και με τον ακόλουθο κώδι- κακώδικα:

```
def linecount(filename):
    count = 0
    for line in open(filename):
        count += 1
    return count
print linecount('demo.py')
```

Εκτελέστε το παραπάνω πρόγραμμα και συζητήστε το αποτέλεσμα. Υπόδειξη: Το αποτέλεσμα είναι ότι θα εμφανίσει το πλήθος των γραμμών του αρχείου.

Δραστηριότητα 3

Να γράψετε πρόγραμμα στη γλώσσα Python, το οποίο θα δέχεται ως είσοδο το όνομα ενός αρχείου, θα εμφανίζει τα περιεχόμενά του κατά γραμμή και στη συνέχεια θα γράφει σε ένα άλλο αρχείο, τις γραμμές του αρχείου με την αντίστροφη σειρά.

Δραστηριότητα 4

Τι πιστεύετε ότι θα συμβεί, όταν θα εκτελεστούν τα παρακάτω σενάρια. Τεκμηριώστε την άποψή σας.

Παράδειγμα

```
my_list = [i**2 for i in range(1,11)]
f = open("output.txt", "w")("output.txt", "w")
for item in my_list:
  f.write(str(item) + "\n")"\n") # δέχεται string όρισμα η
write f.close()

f = open("output.txt", "r")("output.txt", "r")
print f.readline()
print f.read()
f.close()
```

6.4 Ερωτήσεις - Ασκήσεις

1. Σε τι χρησιμεύει ένα αρχείο δεδομένων;
2. Ποιες βασικές λειτουργίες - - μεθόδους υποστηρίζει η γλώσσα προγραμματισμούπρογραμματι- σμού Python για τη διαχείριση αρχε- ίωναρχείων;
3. Με ποια συνάρτηση (μέθοδο) ανοίγουμε ένα αρχείο για ανάγνωση;
4. Πώς ανοίγουμε ένα αρχείο για εγγραφή;
5. Με ποια συνάρτηση κλείνουμε ένα αρχείο;
6. Με ποια συνάρτηση γράφουμε περιεχόμενο σε ένα αρχείο;
7. Ποια συνάρτηση μπορούμε να χρησιμοποιήσουμε για να διαβάσουμε δεδομέναδεδο- μένα από μία γραμμή, από ένα αρχείο;

Σύνοψη

Στο κεφάλαιο αυτό ασχοληθήκαμε με τη διαχείριση αρχείων δεδο-μένωνδεδομένων, κυρίως από τη σκοπιά της γλώσσας προγραμματισμού Python. Επίσης, με θέματα ταυτοποίησής τους από το Λειτουργικό Σύστημα, όπως το όνομαόνο- μα, η θέση στο αποθηκευτικό μέσο και με βασικές λειτουργίες σε αυτά.

7

Προηγμένα στοιχεία γλώσσας προγραμματισμού

7. Προηγμένα στοιχεία γλώσσας προγραμματισμού

Εισαγωγή

Στο κεφάλαιο αυτό θα αναπτυχθούν προηγμένα χαρακτηριστικά της γλώσσας προγραμματισμούπρο- γραμματισμού Python και συγκεκριμένα. Συγκεκριμένα θα αναπτυχθούν οι ιδιό- τητεςιδιότητες των υποπρογραμμάτωνυποπρο- γραμμάτων, η εμβέλεια μεταβλητών, τα αρθρώ- ματααρθρώματα και τα πακέτα.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- αναφέρουμε τα χαρακτηριστικά των υποπρογραμμάτων και τον τρόπο κλήσης τους, με την προσαρμογή τους στη γλώσσα προγραμματισμού
- εντοπίζουμε και απομονώνουμε κατάλληλα τμήματα κώδικα από έτοιμο πρόγραμμαπρό- γραμμα για μετατροπή σε υποπρογράμματα
- σχεδιάζουμε τη λύση προγραμματιστικού προβλήματος με αρθρωτό τρόπο, όπου αυτό είναι σκόπιμο
- αναγνωρίζουμε την εμβέλεια μεταβλητών και παραμέτρων σε σύνθετα προγράμματαπρο- γράμματα.

Λέξεις κλειδιά

Συνάρτηση, παράμετρος, εμβέλεια, άρθρωμα, τοπική μεταβλητή.

Διδακτικές Ενότητες

7.1 Υποπρογράμματα και τρόποι κλήσης τους

7.1.1 Υποπρογράμματα

Μεταξύ των θεμάτων της αλγοριθμικής επίλυσης προβλημάτων που έχουμε ήδη εξετάσει, είναι και εκείνα της ανάγκης επανάληψης~~επανάληψης~~ κώδικα με τη χρήση μιας δομής επανάληψης ή ομαδοποίησης και χειρισμού πολλών δεδομένων με τη χρήση άλλων δομών δε- δομένων~~δεδομένων~~, όπως η Λίστα.

Προχωρώντας σε πιο πολύπλοκα προβλήματα, θα δούμε ότι συχνά~~συχνά~~ κάποια ενέργεια ή και ολόκληρο αλγοριθμικό τμήμα, είναι αναγ- καίο~~αναγκαίο~~ να επαναλαμβάνεται, σχεδόν αυτούσια~~αυτούσια~~ ως λογική επεξεργα- σία~~επεξεργασία~~, σε πολλά διαφορετικά σημεία του αλγορίθμου, αλλά κάθε φο- ρά~~φορά~~ να επεξεργάζεται διαφορετικά δεδομένα. Ο τρόπος αυτός "ε-

πανάληψηςεπανάληψης" δεν μπορεί να υλοποιηθεί με μια δομή επανάληψης, λόγω της δυνατότητας επεξεργασίας επεξερ- γασίας διαφορετικών δεδομένων κάθε φορά. Φυσικά δεν είναι προγραμματιστικά ορθό να επανα-λαμβάνουμεεπαναλαμβάνουμε το τμήμα κώδικα με άλλα δεδομένα κάθε φορά, αλλά να επιτρέπει η γλώσσα προγραμματισμού τη δημιουργία ομάδας εντολών, ως οντότηταοντότη- τα στο πρόγραμμα, που να δέχεται ως είσοδο παραμέτρους και να μπορεί να καλείταικαλεί-

~~τα~~ από σημεία του προγράμ- ματος~~προγράμματος~~ και με συγκεκριμένες τιμές για τις παραμέτρους ανά κλήση. Αυτό έχει ως αποτέλεσμα τη δυνατότητα πολλαπλής διενέργειας μιας επεξεργασίας με συγκεκριμένα δεδομένα κάθε φορά.

Μια από τις βασικότερες τεχνικές του διαδικαστικού προγραμματισμού~~προγραμματισμού~~ είναι ο **Τμηματικός~~Τμη- ματικός~~ Προγραμματισμός**. Σύμφωνα με την τεχνική αυτή, μπορούμε να γράψουμε~~γράψου- με~~ ένα πρόγραμμα ως ένα σύ- νολο~~σύνολο~~ από μικρότερα κομμάτια προγράμματος. Με αυτόν τον τρό- πο~~τρόπο~~ δεν είναι απαραίτητο να ξαναγράφουμε τον ίδιο κώδικα σε διαφορετικές~~διαφο- ρετικές~~ υλοποιήσεις. Τον γράφουμε μία~~μια~~ φορά, δημιουργώντας ένα *υποπρόγραμμα* και τον χρησιμοποιούμε, όσες φορές θέλουμε σε ένα πρόγραμμα, καλώντας το αντίστοιχο~~αντί- στοιχο~~ υποπρόγραμμα. Ένα υποπρόγραμμα λοιπόν είναι ένα κομμάτι προγράμματος~~προγράμ- ματος~~ που έχει γραφεί ξεχωριστά από το υπόλοιπο πρόγραμμα και επιτελεί ένα αυτόνομο έργο.

Κάθε υποπρόγραμμα πρέπει να έχει τα παρακάτω βασικά χαρακτηριστικά~~χαρακτηριστικά~~:

α) Έχει μόνο ένα σημείο εισόδου από το οποίο δέχεται τα δεδομέ- να~~δεδομένα~~ του.

β) Το (υπο)πρόγραμμα το οποίο καλεί ένα άλλο υποπρόγραμμα σταματάει την εκτέλεσή του όσο εκτελείται το καλούμενο υποπ- ρόγραμμα. Μόνο ένα υποπρόγραμμα~~. Μόνο ένα υπο- πρόγραμμα~~ μπορεί να εκτελείται σε μια χρονική στιγμή.

γ) Ο έλεγχος επιστρέφει στο (υπο)πρόγραμμα το οποίο καλεί, ό- τανόταν~~ταν~~ το καλούμενο~~καλούμε- νο~~ υποπρόγραμμα σταματήσει να εκτελείται.

Καλές πρακτικές

Πριν ξεκινήσουμε να γράφουμε ένα πρόγραμμα, μελετάμε πώς αυτό μπορεί να *αναλυθεί* σε επιμέρους τμήματα και αποφασίζου- μεαποφασίζουμε για τα αντίστοιχα υποπρογράμματαυπο- προγράμματα.

Εξετάζουμε αν κάποια υποπρογράμματα, τα οποία έχουμε ήδη γράψει ή υπάρχουνυπάρ- χουν σε έτοιμες *βιβλιοθήκες* προγραμμάτων, μπο- ρούνμπορούν να χρησιμοποιηθούν, για να κερδίσουμε χρόνο και να απο- φύγουμεαποφύγουμε λάθη.

Προσπαθούμε κάθε υποπρόγραμμα να είναι όσο το δυνατόν πιο *ανεξάρτητο* από τα άλλα. Αυτό μας προφυλάσσει από λάθη στο πρόγραμμά μας και επιτρέπειεπι- τρέπει τη χρήση του σε άλλα προγράμμα-προγράμματα αργότερα.
τα αργότερα.

Πολλά από τα ανθρώπινα τεχνουργήματα, φυσικά ή τεχνητά, στη- ρίζονταιστηρίζονται στην ιδέα της επαναχρησιμοποίησης – συνδυασμού μικ- ρότερωνμικρότερων δομικών μονάδων, με αποτέλεσμα την απλότητα και την ποικιλία αντικειμένων και χαρακτηριστικών. Στην Python, η τεχνική αυτή υλοποιείται με τις συναρτήσεις, δίνοντας καλύτερο έλεγχο και *υψηλό επίπεδο αφαίρεσης*.

7.1.2 Συναρτήσεις στην Python

Κάθε γλώσσα προγραμματισμού διαθέτει ένα δικό της λεξιλόγιο για τις εντολές και τις ενσωματωμένες συναρτήσεις, ενώ προγ- ράμματαπρογράμματα που γράφονται πρέπει να ακολουθούνακο- λουθούν αυστηρούς γραμ- ματικούςγραμματικούς και συντακτικούς κανόνες για τη δόμηση των εντολών, τη δημιουργία συναρτήσεων χρήστη κ.ά. Στη συνέχεια θα δούμε πώς υλοποιείταιυλο- ποιείται ο τμηματικόςΤμηματικός προγραμματισμός στη γλώσσα προγ- ραμματισμούπρογραμματισμού Python.

Η Python παρέχει ένα μόνο τύπο υποπρογραμμάτων, τις **συναρ- τήσειςσυναρτήσεις**, τις οποίες τις θεωρεί ως *αντικείμενα*.

Για να εξασκηθούμε μπορούμε να δημιουργήσουμε, σε πρώτο επίπεδο, τις δικές μας συναρτήσεις και έτσι, να κατανοήσουμε τη λειτουργία και τη χρήση τους. Σε υψηλότερο επίπεδο, μπορούμε να χρησιμοποιούμε και να ενσωματώνουμε στο πρόγραμμα έτοι- μεςέτοιμες βιβλιοθήκες, που είτε παρέχει το περιβάλλον ή υπάρχουν σε

άλλα συστήματα. Έτσι, δεν είναι ανάγκη να ξαναγράφουμε κώδικα που υπάρχει διαθέσιμος, μειώνοντας την πολυπλοκότητα και την πιθανότητα σφαλμάτων (bugs).

Ορισμός και κλήση Συνάρτησης

Ο ορισμός μιας συνάρτησης περιλαμβάνει το όνομά της και τις παραμέτρους εισόδου και καλείται από σημεία του προγράμματος μέσω της λειτουργίας που ονομάζεται **κλήση** (calling) της συνάρτησης. Ορίζεται με τη λέξη κλειδί **def** που την ακολουθεί ένα όνομα το οποίο την ταυτοποιεί και ένα ζεύγος παρενθέσεων που μπορούν να περιέχουν ονόματα μεταβλητών, ενώ η δήλωση τελειώνει με διπλή τελεία (:). Κάτω από τη γραμμή αυτή τοποθετούνται, σε *σειρά*, οι εντολές που καθορίζουν τη λειτουργία της συνάρτησης. Ο ορισμός συναρτήσεων στην Python είναι απλός σε σχέση με άλλες γλώσσες (ο τύπος των παραμέτρων δε δηλώνεται).

Παράδειγμα ορισμού συνάρτησης χωρίς χρήση παραμέτρων

```
def fun_name():
    print “hello”
```

Η συνάρτηση αυτή, όταν κληθεί, εμφανίζει στην οθόνη τη λέξη hello.

Παράδειγμα, ορισμού συνάρτησης με χρήση δύο παραμέτρων

```
def find_sum(par1, par2):
    result = par1+par2
    return result
```

Η συνάρτηση αυτή επιστρέφει το άθροισμα των τιμών των δύο παραμέτρων.

Μια συνάρτηση μπορεί να καλείται από διάφορα σημεία του κύριου προγράμματος ή και μέσα από μια άλλη συνάρτηση, γράφοντας το όνομά της και τις κατάλληλες παραμέτρους μέσα σε παρενθέσεις.

Παράδειγμα 1: Κλήση συνάρτησης

```
>>> type(45)
<type 'int'>
```

Εδώ, το όνομα της συνάρτησης είναι type και η έκφραση μέσα στις παρενθέσεις, που είναι το όρισμα~~ονομάζεται παράμετρος~~ της συνάρτησης, είναι το 45, ενώ η συνάρτηση υπολογίζει~~υπο- λογίζει~~ και επιστρέφει τον τύπο του ορίσμα- τος~~ορίσματος~~. Είναι σύνηθες να λέμε πως η συνάρτηση~~συ- νάρτηση~~ δέχεται ορίσματα και επιστρέφει ένα αποτέλεσμα. Το αποτέλεσμα αυτό ονομάζεται **επιστρεφόμενη**~~**επιστρεφόμενη**~~ **τιμή** (return value).

Παράδειγμα 2

Κλήση της συνάρτησης find_sum, που δημιουργήσαμε παραπά- νωπαραπάνω:

```
>>>
find_s
um(3,
4)
```

7

Εδώ, η συνάρτηση καλείται με το όνομά της και με ορίσματαπαραμέτρους τις τιμές 3 και 4, ενώ επιστρέφει το άθροισμά τους που είναι η τιμή 7.

Αν καλέσουμε πάλι την ίδια συνάρτηση "περνώντας" τώρα διαφο-ρετικέςδιαφορετικές τιμές στις παραμέτρους, για παράδειγμα δύο συμβολοσει- ρέςσυμβολοσειρές, η συνάρτηση επιστρέφει τη συνένωσησυ- νένωση των συμβολοσειρών αυτών ως αποτέλεσμα του *πολυμορφισμού* που ισχύει στον τελεσ-τελεστή + στην Python.
τή + στην Python.

```
>>>
find_sum("well","
come")
```

"wellcome"

Σημείωση: Μια συνάρτηση πρέπει να έχει οριστεί πριν χρησιμο-ποιηθεί.χρησιμοποιηθεί. Είναι προφανέςπρο- φανές ότι οι δηλώσεις (statements) μέσα στη συνάρτηση δεν εκτελούνται μέχρι αυτή να κληθεί.

Ροή εκτέλεσης

Ένας τρόπος να διαβάζουμε προγράμματα είναι να ακολουθούμε τη ροή εκτέλεσης των εντολών, κάτι το οποίο μπορεί να γίνει γρή-γοραγρήγορα πολύπλοκο και δυσανάγνωστο.δυσανάγνω- στο. Ένας εναλλακτικός τρόπος είναι, όταν βρεθούμε σε μια κλήση συνάρτησης, αντί να ακολου-

θήσουμεακολουθήσουμε τη *ροή της εκτέλεσης* στον εσωτερικό κώδικα της συνάρτησης, να υποθέτουμε ότι η συνάρτηση δουλεύει σωστά και επισ- τρέφειεπιστρέφει το σωστό αποτέλεσμα. Οπότε, να συνεχίζουμε με την επό- μενηεπόμενη εντολή του κύριου προγράμματος.κυρίου προγράμ-

~~ματος.~~ Στην πραγματικότητα ε- φαρμόζουμε~~εφαρμόζουμε~~ κάτι ανάλογο με τη χρησιμοποίηση των ενσωματω- μένων~~ενσωματωμένων~~ συναρτήσεων. Έτσι, όταν καλούμε τη math.cos ή τη math.exp, δεν εξετάζουμε τον εσωτερικό κώδικα αυτών των συ- ναρτήσεων~~συναρτήσεων~~, αλλά θεωρούμε ότι δουλεύουν σωστά και δεχόμαστε το αποτέλεσμα. Το ίδιο ισχύει και όταν καλούμε μια από τις δικές μας συναρτήσεις.

Μια κλήση συνάρτησης είναι σαν μια παράκαμψη στη ροή της εκ- τέλεσης~~εκτέλεσης~~. Αντί η ροή να πάει στην επόμενη δήλωση, περνάει στο σώμα της συνάρτησης, εκτελεί όλες τις δηλώσεις εκεί και μετά ε- πιστρέφει~~επιστρέφει~~ για να συνεχίσει από εκεί που σταμάτησε.

Κατηγορίες συναρτήσεων

Οι συναρτήσεις μπορούν να κατηγοριοποιηθούν με πολλούς τρό- πους.

~~τρόπους.~~ Μια πρώτη κατηγορία συναρτήσεων είναι:

α) αυτές οι οποίες δεν τροποποιούν το αντικείμενο στο οποίο ε- φαρμόζονται~~εφαρμόζονται~~, όπως:

```
>>>
a = 'Python'~~'~~
~~Python'~~
>>>
print(a.upper( ))
PYTHON
>
>
>
```

print(a)
Python

β) εκείνες που μπορούν να αλλάξουν το αντικείμενο στο οποίο καλούνται, όπως στο παράδειγμα 2.

```
>>> b = [ 'a' , 'b' , 'c' , 'd''a ' , 'b ' , 'c ' , 'd ' ]
>>> print (b.append( 'e')reverse( )) None
>
>
>

print (b)
```

['a', 'b', 'c', 'd', ‘e’]

['d ', ' c ', 'b ', 'a ']

Μια άλλη κατηγορία συναρτήσεων είναι:

α) αυτές οι οποίες, όταν κληθούν, επιστρέφουν αποτέλεσμα (κάποιακάποια τιμή)

β) εκείνες που δεν επιστρέφουν κάποια τιμή (κενές/void συναρτήσειςσυναρτήσεις), αλλά εκτελούνεκτε- λούν ενέργειες μέσω των εντολών τους.

Οι *κενές* αυτές συναρτήσεις μπορεί να εμφανίζουν αποτέλεσμα στην οθόνη ή να έχουν κάποιο άλλο αποτέλεσμα, αλλά δεν επιστρέφουν κάποια~~κάποια~~ τιμή.

7.2 Μεταβλητές και παράμετροι

Όπως έχουμε αναφέρει, ένα υποπρόγραμμα αποτελεί ένα ανεξάρτητο~~ανεξάρτητο~~ τμήμα προγράμματος~~προ- γράμματος~~ που μπορεί να καλείται από σημεία του προγράμματος. Δέχεται τιμές από το τμήμα προγράμματος που το καλεί (το κύριο πρόγραμμα ή άλλο υποπρόγραμμα~~υποπρό- γραμμα~~) και μετά την εκτέλεση των εντολών του επιστρέφει σε αυτό νέες τιμές, δηλαδή τα αποτελέσματα. Οι τιμές αυτές που μεταβιβάζονται από το ένα υποπρόγραμμα~~υποπρό γραμμα~~ στο άλλο λέγονται **παράμετροι** και γενικά σε γλώσσες προγραμματισμού διακρίνονται σε *παραμέτρους Εισό- δου~~Εισόδου~~ και Εξόδου*. Οι παράμετροι είναι μεταβλητές οι οποίες χρησιμοποιούνται~~χρησιμοποιούνται~~ για τη μεταβίβαση τιμών μεταξύ υποπρογραμμάτων ή υποπρογραμμάτων και κύριου προγράμματος.

7.2.1 Παράμετροι συναρτήσεων

Μια συνάρτηση μπορεί να δεχθεί παραμέτρους, οι οποίες είναι τιμές που δίνονται/ εισέρχονται στη συνάρτηση, έτσι ώστε αυτή να μπορεί να λειτουργήσει αξιοποιώντας~~αξιοποιώ- ντας~~ αυτές τις τιμές. Οι παράμετροι~~παράμετροι~~ μοιάζουν με τις μεταβλητές, καθορίζονται μέσα στο ζευγάρι των παρενθέσεων, στον ορισμό της συνάρτησης και διαχωρίζον- ται~~διαχωρίζονται~~ με κόμμα, ενώ οι τιμές αυτών των μεταβλητών ορίζονται, όταν καλούμε τη συνάρτηση.

Παράδειγμα

```
def printMax(a, b):
    if a > b:
        print(a, 'είναι το μέγιστο')
    elif a == b:
        print(a, 'είναι ίσο με το', b)
    else:
        print(b, 'είναι το μέγιστο')
```

```
printMax(3, 4) # δίνουμε απ' απ. ευθείας σταθερές
τιμές x = 5
y = 7
printMax(x, y) # δίνουμε μεταβλητές ως ορίσματα
Έξοδος:
4 είναι το μέγιστο
7 είναι το μέγιστο
Τα
```

Μέσα στη συνάρτηση, τα ορίσματα εκχωρούνται σε μεταβλητές οι οποίες ονομάζονται ονομάζο νται *παράμετροι*. Αυτό είναι ένα παράδειγμα συνάρτησης ορισμένης από το χρήστη, η οποία παίρνει ένα όρισμα:

```
def print_twice(x):
    print x
    print x
```

Αυτή η συνάρτηση εκχωρεί το όρισμα σε μια παράμετρο με όνομα x. Όταν καλείται η συνάρτηση, εμφανίζει την τιμή της παραμέτρου, όποια κι αν είναι αυτή, δύο φορές. Η συνάρτηση αυτή δουλεύει με οποιαδήποτε τιμή μπορεί να εμφανιστεί.
οποιαδήποτε τιμή μπορεί να εμφανιστεί.

```
>>> print_twice('Spam')
Spam
Spam
>>> print_twice(25)
25
25
>>> print_twice(math.pi)
3.14159265359
3.14159265359
```

Όταν οι παράμετροι περνάνε με **αναφορά**, σημαίνει ότι αν αλλάξο- υμε αλλάξουμε μια παράμετρο παράμε τρο μέσα στη συνάρτηση, η αλλαγή είναι μόνιμη και μετά την κλήση της συνάρτησης συνάρτη σης. Για παράδειγμα:

```
# Ορισμός συνάρτησης
def changeme( mylist ):
  # "-Αλλάζει τη λίστα που περνά στη συνάρτηση" ."
  mylist.append([1,2,3,4]);
```

```
  print "Τιμές Τιμές μέσα στη συνάρτηση:
  ", mylist return

# Τώρα μπορούμε να καλέσουμε τη changeme συνάρτηση
mylist = [10,20,30];
changeme( mylist );
print "Τιμές έξω από τη συνάρτηση: ",
mylist
```

Οπότε, θα παραχθεί το ακόλουθο αποτέλεσμα: Τιμές μέσα στη συνάρτηση: [10, 20, 30, [1, 2, 3, 4]] Τιμές έξω από τη συνάρτηση: [10, 20, 30, [1, 2, 3, 4]]

Ας προσέξουμε το ακόλουθο παράδειγμα, όπου το όρισμα παρακάμπτεται παρακάμπτεται μέσα στην καλούμενη συνάρτηση.

```
# Ορισμός συνάρτησης
def changeme( mylist ):
   mylist = [1,2,3,4]; # Δημιουργεί νέα αναφορά στη mylist
   print "Τιμές μέσα στη συνάρτηση: ", mylist
   return
# Τώρα μπορούμε να καλέσουμε την τη changeme
συνάρτηση mylist = [10,20,30];
changeme( mylist );
print "Τιμές Τιμές έξω από τη συνάρτηση:
", mylist
```

Η παράμετρος mylist είναι τοπική στη συνάρτηση changeme. Αλλάζοντας Αλλάζοντας την τιμή της mylist μέσα στη συνάρτηση, η αλλαγή αυτή εφαρμόζεται μόνο στην "περιοχή" της συνάρτησης και δεν αλλάζει την τιμή της παραμέτρου mylist στο πρόγραμμα. Οπότε τελικά θα παραχθεί το ακόλουθο αποτέλεσμα:

Τιμές μέσα στη συνάρτηση: [1, 2, 3, 4]
Τιμές έξω από τη συνάρτηση: [10, 20, 30]

7.2.2 Εμβέλεια των μεταβλητών

Όλες οι μεταβλητές σε ένα πρόγραμμα δεν μπορούν να είναι προσπελάσιμες από όλα τα μέρη του προγράμματος. Η **εμβέλεια** (scope) μιας μεταβλητής αναφέρεται στο τμήμα του προγράμμα- τος προγράμματος που μπορεί αυτή να έχει πρόσβαση. Οι γλώσσες προγραμμα- τισμού επιτρέπουν τη χρήση των

μεταβλητών, όχι μόνο στο τμήμα προγράμματος που δηλώνονται, αλλά και σε άλλα υποπρογράμ- ματα και πιο συγκεκριμένα ισχύει:

προγραμματισμού επιτρέπουν τη χρήση των μεταβλητών, όχι μόνο στο τμήμα προ γράμματος που δηλώνονται, αλλά και σε άλλα υποπρογράμματα και πιο συγκεκρι μένα ισχύει:

Απεριόριστη εμβέλεια: Όλες οι μεταβλητές είναι ορατές και μπορούνμπορούν να χρησιμοποιούνταιχρησιμο- ποιούνται σε οποιοδήποτε τμήμα του προγράμ- ματοςπρογράμματος, ανεξάρτητα από το πού δηλώθηκανδη- λώθηκαν. Αυτού του τύπου οι μεταβλητές χαρακτηρίζονται ως **καθολικές** (global). Το μειονέκτη- μαμειονέκτημα είναι ότι περιορίζεται έτσι η ανεξαρτησία των υποπρογραμμά- τωνυποπρογραμμάτων.

Περιορισμένη εμβέλεια: Αυτές οι μεταβλητές είναι **τοπικές** (local), ισχύουν δηλαδήδηλα δή για το υποπρόγραμμα στο οποίο δηλώθη- κανδηλώθηκαν. Η περιορισμένη εμβέλεια απαιτεί όλες οι μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, να δηλώνονταιδηλώ- νονται σε αυτό το τμήμα.

Θα αναλύσουμε στη συνέχεια τους δύο βασικούς τύπους εμβέλει- αςεμβέλειας, τις καθολικές και τις τοπικές μεταβλητές, στην Python.

Καθολικές και Τοπικές μεταβλητές

Οι μεταβλητές που έχουν οριστεί μέσα στο σώμα της συνάρτησης, έχουν τοπική εμβέλεια, ενώ αυτές που ορίστηκαν έξω από αυτό έχουν καθολική εμβέλεια. Αυτό σημαίνει ότι οι τοπικές μεταβλητές μπορούν να προσπελαστούν μόνο μέσα στη συνάρτηση στην ο-

ποίαεποία δηλώθηκαν, ενώ οι καθολικές μεταβλητές μπορεί να είναι προσβάσιμες από όλες τις συναρτήσεις. Ακολουθεί ένα απλό πα-παράδειγμα.
ράδειγμα.

```
total = 0; # Αυτή είναι μια καθολική μεταβλητή.

def sum( arg1, arg2 ):
   total = arg1 + arg2; # Η total είναι τοπική μεταβλητή.
   print "Μέσα"Μέσα στη συνάρτηση η τοπική total : ",",
   total return total;

# Κλήση της sum συνάρτησης
sum( 10, 20 );
print "Έξω"Έξω από τη συνάρτηση η καθολική total :
",", total
```

Όταν εκτελεσθεί ο παραπάνω κώδικας, παράγεται το ακόλουθο αποτέλεσμα: Μέσα στη συνάρτηση η τοπική total : 30
Έξω από τη συνάρτηση η καθολική total : 0

Τοπικές μεταβλητές

Όταν δηλώνουμε μεταβλητές μέσα σε έναν ορισμό συνάρτησης, αυτές δεν έχουν καμία σχέση με άλλες μεταβλητές που έχουν την ίδια ονομασία και χρησιμοποιούνται χρησιμοποιού- νται έξω από αυτήν τη συνάρτηση. Έτσι, σε μια συνάρτηση τα ονόματα των μεταβλητών χρησιμοποι- ούνται μετα- βλητών χρησιμοποιούνται μόνο τοπικά. Όλες οι μεταβλητές έχουν την εμβέλεια του τμήματος κώδικα όπου έχουν δηλωθεί, αρχίζοντας από το σημείο στο οποίο ορίζεται το όνομα. Όπως αναφέρθηκε, η εμβέλεια μιας μεταβλητής ή μιας συνάρτησης, ορίζειορί- ζει την περιοχή του κώδικα στον οποίο αυτή έχει πρόσβαση. Οι τοπικές μεταβλητές που ορί- ζονταιορίζονται σε μια συνάρτηση, χάνονται όταν τελειώσει η εκτέλεση της συνάρτησηςσυνάρτη- σης, ενώ η κλήση μιας συνάρτησης δημιουργεί νέες τοπι- κέςτοπικές μεταβλητές. Άλλωστε και οι παράμετροι είναι τοπικές μεταβλη- τέςμεταβλητές.

Παράδειγμα

```
x = 50
def func(x):
        print('Το x είναι', x)
        x = 2
        print('Το τοπικό x άλλαξε σε', x)
func(x)
    print('Το x είναι ακόμα', x


#) Έξοδος:
Το x είναι 50
Το τοπικό x άλλαξε σε 2
Το x είναι ακόμα 50
```

Χρήση της εντολής global

Οι μεταβλητές που δηλώνονται έξω από τις συναρτήσεις του προγράμματος, είναι **καθολικές μεταβλητές** (global) και προσπελαύνονται από οποιοδήποτε σημείο

μέσα στο πρόγραμμα. Εάν θέλουμε μέσα σε μια συνάρτηση να αλλάξουμε την τιμή μιας κα- θολικήςκαθολικής μεταβλητής, δηλαδή μιας μεταβλητής η οποία ορίζεται στο κορυφαίο επίπεδο του προγράμματος (δηλαδή όχι μέσα σε κάποιου είδους εμβέλεια, όπως σε συναρτήσεις ή κλάσεις), τότε πρέπει να δηλώσουμε στην Python ότι η μεταβλητή αυτή δεν είναι τοπική αλλά καθολική. Αυτό γίνεται με τη χρήση της εντολής **global**, με την οποία γίνεται ξεκάθαρο ότι η μεταβλητή βρίσκεται σε ένα εξωτερικό τμήμα εντολών.
Ας πειραματιστούμε με τα ακόλουθα:-
Παράδειγμα

Παράδειγμα

```
x = 50
def func():
        global x
        print('Το x είναι', x)
        x = 2
        print('Το καθολικό x άλλαξε σε', x)
func()
print('Η τιμή του x είναι', x)

# Έξοδος:
Το x είναι 50
Το καθολικό x άλλαξε σε 2
Η τιμή του x είναι 2
```

Παράδειγμα επίδειξης χειρισμού της Python των μεταβλητών τοπικής και καθολικήςκαθολι- κής εμβέλειας.
Έστω ότι έχουμε το εξής σενάριο:

```
>>> myGlobal = 5

def func1():
        myGlobal = 42
def func2():
        print myGlobal
>>>
```

```
func1()
    global myGlobal
    print myGlobal
    myGlobal = 42

func2()
    print myGlobal
```

```
5
```

Τυπώνει, όχι την τιμή 42, αλλά το 5. Αν βέβαια προσθέσουμε μία δήλωση 'global' ~~'global'~~
στη func1(),όπως στο παρακάτω παράδειγμα,
τότε η func2() θα τυπώσει 42.

```
>>> myGlobal = 5
def func1():
    global myGlobal
    print myGlobal
    myGlobal = 42

def func2():
    print myGlobal

func1()
func2()
```

```
5
42
```

my
Glo
bal
my
Glo
bal
= 42

```
def func2():
    print myGlobal
>>> func1()
>>> func2()
42
```

Αυτό που συμβαίνει, είναι ότι η Python θεωρεί ότι αν μέσα σε μια συνάρτηση εκχωρηθείεκ- χωρηθεί σε οποιαδήποτε μεταβλητή μια τιμή, η με- ταβλητήμεταβλητή αυτή είναι τοπική και η τιμή της ισχύει μόνο για αυτή τη συνάρτηση, εκτός αν δηλωθεί ρητά διαφορετικά. Αν απλώς διαβά- ζειδιαβάζει μια τιμή από τη μεταβλητή και αυτή δεν υπάρχει τοπικά, τότε προσπαθεί να την αναζητήσει σε οποιαδήποτε δυνατή εμβέλεια, για παράδειγμα καθολική. Όταν εκχωρούμε το 42 στη myGlobal, η Python δημιουργεί μια τοπική μεταβλητή, η οποία υπερκαλύπτει την καθολική μεταβλητή που έχει την ίδια ονομασίαονο- μασία. Αυτή η τοπική μεταβλητή βγαίνει εκτός εμβέλειας και χάνεται, όταν η func1() ο- λοκληρώνεταιολοκληρώνεται. Η func2() δε μπορεί ποτέ να δει τι έγινε μέσα σε άλλη συνάρτηση και βλέπει μόνο την καθολική μεταβλητή, (η οποία δεδεν μεταβλήθηκε.).

7.3 Αρθρώματα (Modules)

7.3.1 Εισαγωγή

Έχουμε δει πώς μπορούμε να επαναχρησιμοποιήσουμε κώδικα στο πρόγραμμά μας, ορίζοντας συναρτήσεις. Τι κάνουμε όμως, αν θέλουμε να επαναχρησιμοποιήσουμεεπαναχρησιμοποι- ήσουμε έναν αριθμό συναρτήσεων σε άλλα προγράμματα που γράφουμε; Η λύση είναι τα **αρθρώμα- τα** (modules**αρθρώματα (module)**). Ένα άρθρωμα μας επιτρέπει να οργανώσουμε με λογικό τρόπο έναν κώδικα Python. Η ομαδοποίηση σχετικού κώ- δικακώδικα σε ένα module κάνει τον κώδικα ευκολότερο στην κατανόηση και χρήση. Ένα άρθρωμα είναι ένα

Python αντικείμενο, όπως όλα στη γλώσσα αυτή. Με απλά λόγια, ένα *άρθρωμα είναι ένα αρχείο αποτελούμενο από κώδικα Python* και μπορεί να ορίσει συναρτή- σεις~~συναρτήσεις~~, κλάσεις και μεταβλητές.

Υπάρχουν διάφορες μέθοδοι για να γράφουμε αρθρώματα, αλλά ο απλούστερος τρόπος είναι δημιουργώντας ένα αρχείο με επέκτα- ση~~επέκταση~~ **.py**, το οποίο θα περιέχει συναρτήσεις~~συ- ναρτήσεις~~ και μεταβλητές.

Για να χρησιμοποιήσουμε ένα άρθρωμα στο πρόγραμμά μας, θα πρέπει να το "εισάγουμε"~~"ει- σάγουμε"~~ σε αυτό. Αυτό γίνεται με τη δήλωση im- port~~import~~.

Η

~~Η~~ **δήλωση import**

Η import έχει την ακόλουθη σύνταξη:

import module1[, module2[,... moduleN]

Η **from...import**

Η from...import δίνει τη δυνατότητα εισαγωγής συγκεκριμένων χαρακτηριστικών~~χαρακτηριστικών~~ - δυνατοτήτων του αρθρώματος, χωρίς να εισάγει όλο το άρθρωμα και έχει την ακόλουθη~~ακό- λουθη~~ σύνταξη:

```
from modname import name1[, name2[, ... nameN]]
```

Η δήλωση from...import *

Είναι δυνατό να εισάγουμε όλα τα περιεχόμενα από ένα άρθρωμα, με χρήση της ακόλουθης δήλωσης import:

```
from modname import *
```

Όμως δεν ενδείκνυται η συχνή χρήση του.

Για να έχουμε πρόσβαση σε μια από τις συναρτήσεις, θα πρέπει να προσδιορίσουμε ~~προσδιορίσου- με~~ το όνομα της μονάδας και το όνομα της συνάρτησης~~συνάρτησης~~, χωρισμένα με μία τελεία. Αυτή η μορφή ονομάζεται **συμβολισμός με τελεία** (dot notation).

Σε ένα πρόγραμμα μπορούν να συνυπάρχουν τρεις κατηγορίες συναρτήσεων. Μπορούν να υπάρχουν συναρτήσεις *ενσωματωμένες* στο περιβάλλον~~περιβάλλον~~ (built-in)
που είναι πάντα διαθέσιμες για χρήση, συναρτή- σεις~~συναρτήσεις~~ που περιέχονται σε *εξωτερικά*

αρθρώματα, τα οποία πρέπει πρώτα να εισαχθούν και τέλος συναρτήσεις που ορίζονται **από τον προγραμματιστή** (με το def).

Για παράδειγμα στο παρακάτω πρόγραμμα εκτελούνται ορισμένες μαθηματικές πράξεις:

```
from math import sqrt

def cube(x):
  return x * x * x

  >>> print abs(-1)
  >>> print cube(9)
>>>       print
sqrt(81)
```

Για τις ακόλουθες δηλώσεις παρατηρούμε τα:

from math import sqrt

Η sqrt() συνάρτηση εισάγεται από το math module.

```
def cube(x):
    return x * x * x
```

Η cube() συνάρτηση είναι μια συνάρτηση οριζόμενη από τον προγραμματιστή.

```
print abs(-1)
    print abs(-1)
```

Η abs() συνάρτηση είναι μια ενσωματωμένη built-in συνάρτηση, προσβάσιμη καθολικά και αποτελεί μέρος του πυρήνα (core) της γλώσσας.

7.3.2 Σύντομη περιγραφή της Πρότυπης βιβλιοθήκης (Standard Library)

Μια βιβλιοθήκη (library), σε οποιαδήποτε γλώσσα προγραμματισμού, είναι μια συλλογή εργαλείων που μπορεί να έχουν γραφτεί και από άλλους προγραμματιστές, προκειμένου να εκτελούνται συγκεκριμένες λειτουργίες. Οι βιβλιοθήκες είναι πολύ σημαντικές στον προγραμματισμό, γιατί μας δίνουν τη δυνατότητα να χρησιμοποιούμε τα εργαλεία που περιλαμβάνονται σε αυτές.

Η *πρότυπη βιβλιοθήκη* της Python περιέχει έναν τεράστιο αριθμό χρήσιμων αρθρωμάτωναρ θρωμάτων και είναι μέρος κάθε πρότυπης εγκατάστασηςεγκατάστασης Python. Είναι σημαντικό να εξοικειωθούμε με την πρότυπη βιβλιοθήκη, επειδή πολλά προβλήματα μπορούν να λυθούν γρή- γοραγρήγορα, αν αξιοποιηθεί το εύρος των δυνατοτήτων που έχουν οι βιβ- λιοθήκεςβιβλιο θήκες. Περιλαμβάνει τμήματα για προγραμματισμό γραφικών (Tkinter), αριθμητική επεξεργασία, web συνδεσιμότητα, βάσεις δεδομένων (Sqlite3, Anydbm), Βιοπληροφορική (Biopython) κ.ά. Επίσης, βιβλιοθήκες από πολλές άλλες γλώσσες προγραμματισ- μούπρογραμματι- σμού, μπορούν να χρησιμοποιηθούν στην Python.

Η Python διαθέτειέχει μια μαθηματική μονάδα λογισμικού (**math module**), η οποία περιέχει τις δημοφιλέστερες μαθηματικές συναρ- τήσεις.συναρτήσεις. Προτού χρησιμοποιήσουμε μια μονάδαμο- νάδα, άρα και τη math, πρέπει να την εισάγουμε.

Παράδειγμα 1. Μαθηματική μονάδα λογισμικού

α)

Α)

```
>>> import math
>>> print math
<module 'math'‘math’ (built-in)>
>>> print math.pi
3.14159265359
>>> print math.cos(math.pi / 4.0)
0.70710678118654757
```

Αν εισάγουμε τη math, θα πάρουμε ένα αντικείμενο το οποίο πε- ριέχειπεριέχει σταθερές, όπως η pi και συναρτήσεις, όπως η sin και η exp. Αλλά, αν προσπαθήσουμε να αποκτήσουμε απευθείας πρόσβαση στην pi, θα πάρουμε ένα μήνυμα λάθους.

β

Β) Εναλλακτικά, μπορούμε να εισάγουμε ένα αντικείμενο από μια μονάδα ως εξής:

```
>>> from math import pi
```

Τώρα μπορούμε να έχουμε απευθείας πρόσβαση στην pi, χωρίς το συμβολισμό τελείας.

```
>>> print pi
3.14159265359
```

Επίσης μπορούμε να χρησιμοποιήσουμε τον τελεστή αστεράκι για να τα εισάγουμε όλα από τη μονάδα:

```
>>> from math import *
>>> cos(pi)
-1.0
```

Παράδειγμα 2. Μονάδα Λογισμικού random

Το **random module** διαθέτει εργαλεία δημιουργίας ψευδοτυχαίων αριθμών:

```
>>> import random
>>> random.choice(['apple', 'pear', 'banana'])
'apple'
# τυχαία δειγματοληψία
'apple'
>>> random.sample(xrange(100), 10) # τυχαία δειγματοληψία
[30, 83, 16, 4, 8, 81, 41, 50, 18, 33]
>>> random.random() # τυχαίος κινητής υποδιαστολής (float)
>>> random.random()
0.17970987693706186
>>> random.randrange(6) # τυχαίος ακέραιος στο διάστημα range(6)
>>> random.randrange(6)
5
```

7.3.3 Πακέτα (Packages)

Μέχρι τώρα, πρέπει να έχουμε αρχίσει να παρατηρούμε την ιε- ραρχία ιεραρχία με την οποία οργανώνονται τα προγράμματά μας. Οι με- ταβλητές μεταβλητές, συνήθως, πηγαίνουν μέσα στις συναρτήσεις, ενώ οι συ- ναρτήσεις συναρτήσεις και οι καθολικές μεταβλητές πηγαίνουν συχνά

μέσα στα αρθρώματα. Τι θα γινόταν όμως, αν θέλαμε να οργανώσουμε τα αρθρώματααρθρώ- ματα;

Τα πακέτα είναι ένα εργαλείο για την ιεραρχική οργάνωση των αρθρωμάτωναρθρωμάτων. Η Python παρέχει ένα απλό σύστημα δημιουργίας πα- κέτωνπακέτων, ως επέκταση του μηχανισμούμηχανι- σμού των αρθρωμάτων. Κάθε κα-

τάλογοςκατάλογος με ένα __init__.py αρχείο αναφέρεται ως ένα Python πακέτο.πακέτο.

Δραστηριότητες - Ερωτήσεις - Ασκήσεις κεφαλαίου

7.4 Δραστηριότητες

Δραστηριότητα 1

Να εκτελεστούν και να συζητηθούνκατανοηθούν τα παρακάτω παραδείγματα:

Παράδειγμα 1

```
>>> name = "black"

def f():
    name = "white"
    print "Μέσα στη συνάρτηση",name

>>> print "Έξω από τη συνάρτηση",name
>>> f()
```

Έξοδος:

Έξω από τη συνάρτηση black

Μέσα στη συνάρτηση white

Εξήγηση: ΜιαΜία μεταβλητή που ορίζεται στο σώμα της συνάρτησης έχει τοπική εμβέλειαεμβέλεια, δηλαδή έχει ισχύ μόνο μέσα στο σώμα της συνάρτησης.

Παράδειγμα 2

```
>>> name = "black"

def f( ):
  print "Μέσα στη συνάρτηση", name

>>> print "Έξω από τη συνάρτηση", name
>>> f( )
Έξοδος:
Έξω από τη συνάρτηση black
Μέσα στη συνάρτηση black
```

Μπορούμε να πάρουμε τα περιεχόμενα μιας καθολικής global μεταβλητής μέσα στο σώμα μιας συνάρτησης. Αλλά αν θέλουμε να αλλάξουμε μία καθολική μεταβλητή σε μία συνάρτηση, πρέπει να χρησιμοποιήσουμε τη δήλωση global.

Παράδειγμα 3

```
>>> name = "black"

def f( ):
  global name
  name =
  "white"
  print "Μέσα στη συνάρτηση", name

    >>> print "Έξω από τη συνάρτηση", name
>>> f( )
>>> print "Έξω από τη συνάρτηση", name
    Έξω από τη συνάρτηση black Μέσα στη
    συνάρτηση white Έξω από τη συνάρτηση
    white
```

Το παράδειγμα αυτό αλλάζει το περιεχόμενο της global μεταβλητής name μέσα στη συνάρτηση f( ).
Με τις εντολές:

```
global name
name =
"white"
```

Με χρήση της global, αναφερόμαστε στη μεταβλητή name που ορίζεται έξω από το σώμα της συνάρτησης. Ακολούθως στη με- ταβλητήμεταβλητή αυτή δίνεται μια νέα τιμή.Έξοδος:
Έξω από τη συνάρτηση black
Μέσα στη συνάρτηση white
Έξω από τη συνάρτηση white

7.5 Ερωτήσεις

1. Ποια είναι η διαφορά παραμέτρων και απλών μεταβλητών;
2. Τι ονομάζεται εμβέλεια μεταβλητών;
3. Τι χαρακτηρίζει τις καθολικές μεταβλητές;
4. Ποια η διαφορά της περιορισμένης από την απεριόριστη εμβέλεια;
5. Ποια είναι η λειτουργία της εντολής global;
6. Τι γνωρίζετε για τα αρθρώματα;
7. Ποια είναι η χρησιμότητα της δήλωσης import;
8. Τι γνωρίζετε για τα πακέτα της Python;
9. Ποια είναι η χρησιμότητα των βιβλιοθηκών;
10. Τι ορίζει η εντολή def;

Σύνοψη

Όπως οι συναρτήσεις είναι επαναχρησιμοποιούμενα μέρη προγ- ραμμάτωνπρογραμμάτων, τα αρθρώματααρ- θρώματα είναι επαναχρησιμοποιούμενα προγ- ράμματαπρογράμματα. Τα πακέτα είναι μια άλλη ιεραρχία, για να οργανώνουμε αρθρώματα. Η πρότυπη βιβλιοθήκη που συνοδεύει την Python είναι ένα παράδειγμα τέτοιων πακέτων και αρθρωμάτων.

Ένα άρθρωμα (module) είναι μια συλλογή σχετικών συναρτήσεων και αποθηκεύεται σε αρχείο με κατάληξη .py. Μπορούμε, να γρά- ψουμεγράψουμε και τα δικά μας αρθρώματα, αποθηκεύοντας τις συναρτήσε- ιςσυναρτήσεις μας σε ένα αρχείο .py. Για να χρησιμοποιήσουμε ένα άρθρωμα σε ένα πρόγραμμα, θα πρέπει να το εισάγουμε με την εντολή import.

~~ένα άρθρωμα σε ένα πρόγραμμα, θα πρέπει να το εισάγουμε με την εντολή import.~~

Εκτός από τις ενσωματωμένες βιβλιοθήκες (μονάδες) συναρτήσε- ων ~~συναρτήσεων~~ που περιλαμβάνονται~~περιλαμ- βάνονται~~ στη γλώσσα Python, μπορούμε να βρο- ύμε~~βρούμε~~ στους δικτυακούς τόπους υποστήριξης~~υπο στήριξης~~ της γλώσσας και εξω- τερικές~~εξωτερικές~~ μονάδες λογισμικού με πληθώρα επιπλέον συναρτήσεων για τη δημιουργία ισχυρών προγραμμάτων.

Ένα από τα καθοριστικά πλεονεκτήματα της γλώσσας Python, είναι ότι υποστηρίζεται~~υποστη- ρίζεται~~ από μια παγκόσμια κοινότητα προγραμμα- τιστών~~προγραμματιστών~~ που συνεισφέρουν με τη δημιουργία πολλών εξωτερικών βιβλιοθηκών. Τις βιβλιοθήκες αυτές μπορούμε να χρησιμοποιή- σουμε~~χρησιμοποιήσουμε~~, για να δημιουργήσουμε γρήγορα εντυπωσιακές και φιλικές προς το χρήστη εφαρμογές. Τις συλλογές αυτές μπορούμε να τις θεωρήσουμε ως πρόσθετες εργαλειοθήκες με έτοιμα εργαλεία, τα οποία μπορούμε να χρησιμοποιήσουμε~~χρησιμοποι- ήσουμε~~ μέσα στον κώδικά μας, αφού οι περισσότερες εργαλειοθήκες της Python ανήκουν στην κατηγορία του ελεύθερου λογισμικού~~.~~

8

Δομές Δεδομένων II

8. Δομές Δεδομένων ΙΙ

Εισαγωγή

Μια **δομή δεδομένων** μπορεί να οριστεί ως ένα σχήμα οργάνω- σηςοργάνωσης σχετικών μεταξύμετα- ξύ τους στοιχείων δεδομένων. Η επιλογή της κατάλληλης δομής δεδομένων παίζει σημαντικό ρόλο στην ανάπ- τυξηανάπτυξη του αλγορίθμου, για την επίλυση ενός προβλήματοςπροβλήμα- τος.

Στο κεφάλαιο αυτό θα αναπτυχθούν οι *ενσωματωμένες* δομές της Python, όπως τα αλφαριθμητικά, οι λίστες, οι πλειάδες και τα λεξι- κάλεξικά. Σχετικά με τις λίστες και τα αλφαριθμητικάαλ- φαριθμητικά, που είχαν καλυφθεί και στην ύλη της Β' Λυκείου, θα γίνει μια πιο εκτενήςεκτε- νής παρουσίαση των βασικών λειτουργιών τους, μέσα από διάφορα παραδείγματα. Επίσης, με χρήση των ενσωματωμένων δομών της Python θα παρουσιαστούν υλοποιήσειςυλο- ποιήσεις σύνθετων δομών δεδομένων, όπως είναι η στοίβα και η ουρά. Τέλος θα παρουσιαστούν θεωρητικά, οι δομές των γράφων και των δέντρων.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου, θα μπορούμε να:

- αναλύουμε τρόπους αναπαράστασης στη μνήμη βασικών στατικών και δυναμικώνδυνα- μικών δομών
- αναφέρουμε τα ιδιαίτερα χαρακτηριστικά των δομών:

πίνακας, λίστα, στοίβα πίνακα, λίστας, στοίβας και ουράουράς

- κατονομάζουμε τη χρησιμότητα των δομών: δέντροδέντρα και γράφοςγράφοι
- επιλέγουμε και χρησιμοποιούμε κατάλληλες δομές στα προγράμματα που υλοποιούμε.

Λέξεις κλειδιά

Δομές δεδομένων, λίστα, στοίβα, ουρά, λεξικό, γράφος.

Διδακτικές Ενότητες

8.1 Συμβολοσειρές (strings)

Τα **αλφαριθμητικά** ή **συμβολοσειρές** στην Python είναι ακολου- θίεςακολουθίες από χαρακτήρεςχαρακτή- ρες που έχουν σταθερό μέγεθος και μη *μεταβαλ- λόμενα**μεταβαλλόμενα* περιεχόμενα. Δηλαδή, δεν

μπορούμε να προσθέσουμε ή να αφαιρέσουμε χαρακτήρες, ούτε να τροποποιήσουμε~~τροποποιήσου με~~ τα περιε- χόμενα~~περιεχόμενα~~ του αλφαριθμητικού. Γι'Γι' αυτό λέμε ότι η δομή αυτή ανήκει στις *μη μεταβαλλόμενες* (immutable) δομές της Python. Η αρίθμη- ση~~αρίθμηση~~ των χαρακτήρων σε ένα αλφαριθμητικό ξεκινάει από το 0. Για παράδειγμα: αν **word = "PYTHON"**, η αναπαράσταση του αλφα- ριθμητικού~~αλφαριθμητικού~~ μοιάζει με το παρακάτω σχήμα:

0	1	2	3	4	5
'P'	**'Y'**	**'T'**	**'H'**	**'O'**	**'N'**

Για παράδειγμα: το στοιχείο word[3] αναφέρεται στο 4ο γράμμα ('H'), ενώ το στοιχείο~~στοι χείο~~ word[0], στο 1ο γράμμα ('P') . Ο τύπος των αλφαριθμητικών στην Python ονομάζεται~~ονο- μάζεται~~ **str**, από τα αρχικά της λέξης string. Παρακάτω δίνονται μερικά παραδείγματα~~παραδείγμα- τα~~ εντολών με αλφαριθμητικά στον~~στο~~ διερμηνευτή της Python~~, όπου φαίνεται ότι~~:

```
>>> word = "PYTHON"
>>> len( word )
6
>>> print word[5]+word[0]
NP
>>> str(28) == '28'
True
>>> print int( '496' ) + 4
500
```

- η συνάρτηση len επιστρέφει το μήκος, δηλαδή το πλήθος των χαρακτήρων του αλφαριθμητικού
- ο τελεστής + όταν εφαρμόζεται σε αντικείμενα τύπου string, έχει σαν αποτέλεσμα ~~αποτέλεσμα~~ τη συνένωσή τους σε μια συμβολοσειρά
- η συνάρτηση str μετατρέπει μια τιμή σε συμβολοσειρά

- με τη συνάρτηση int μπορούμε να μετατρέψουμε ένα αλφαριθμητικόαλφαριθμητικό στον ακέραιο αριθμό που αναπαριστά.

Έλεγχος ύπαρξης

Ένας σημαντικός τελεστής που θα συναντήσουμε και αργότερα στις λίστες, είναι ο *υπαρξιακός τελεστής* **in**, ο οποίος ελέγχει, αν ένα αντικείμενο ανήκει σε ένα σύνολο αντικειμένων. Δεδομένου ότι οι συμβολοσειρές μπορούν να θεωρηθούν ως σύνολα χαρακτή- ρων, μπορούμε να τον χρησιμοποιήσουμε όπως φαίνεται παρακάτω.

```
>>> "Py" in "Python"
True
>>> "a" in "Python"
False
>>> "a" not in "Python"
True
>>> 'antonis' > 'antonia'
True
>>> '1000' < '2'
True
>>> 'babylon5' > 'babylon4'
True
```

χαρακτήρων, μπορούμε να τον χρησιμοποιήσουμε όπως φαίνεται παρακάτω.

>>> "Py" in "Python"	>>> 'antonis' > 'antonia'
True	True
>>> "a" in "Python"	>>> '1000' < '2'
False	True
>>> "a" not in "Python"	>>> 'babylon5' > 'babylon4'
True	True

Επίσης, οι γνωστοί *συγκριτικοί τελεστές* (<, <=, >, >=, ==, !=) ισχύ- ουν=) ισχύουν και στις συμβολοσειρέςσυμβο- λοσειρές, η λειτουργία των οποίων βασίζεται στη λεξικογραφική διάταξη των χαρακτήρωνχαρα- κτήρων.

Παράδειγμα 1.÷ Σάρωση των χαρακτήρων μιας συμβολοσειράς

Μια συμβολοσειρά είναι μια ακολουθία (sequence) από αντικείμε- ναντικείμενα, τα οποία, στην προκειμένη περίπτωση, είναι χαρακτήρες. Αν θέλουμε να σαρώσουμε τα αντικείμεναντικεί- μενα αυτά ένα-ένα, μπορούμε να το κάνουμε με μια εντολή επανάληψης for. Το παρακάτω τμήμα κώδικα δημιουργεί μια νέα συμβολοσειρά η οποία είναι όμοια με την αρχική, αλλά χωρίς τα κενά ανάμεσα στις λέξεις:

την αρχική, αλλά χωρίς τα κενά ανάμεσα στις λέξεις:

```
def trimSpaces( sentence ):
    result = ""
    for char in sentence :
        if  char != " " :
            result += char
    return result

>>> phrase = "Houston we have a problem"
>>> trimSpaces( phrase )
'Houstonwehaveaproblem'
```

Παράδειγμα 2. Καταμέτρηση φωνηέντων μιας φράσης

Η παρακάτω συνάρτηση υπολογίζει πόσα φωνήεντα έχει η λέξη και επιστρέφει το πλήθος των φωνηέντων της λέξης που δέχεται ως παράμετρο.

```
  word.

def count_vowels( word ):
    vowels = "AEIOUaeiou"
    count = 0
  for letter in word : if
     letter in vowels:
        count += 1
    return count
```

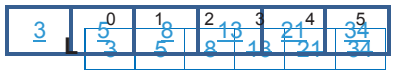
Σημείωση: Παρατηρήστε το διαφορετικό τρόπο με τον οποίο χρησιμοποιείται χρησιμοποιείται ο τελεστής τε λεστής **in**, στη μία μια περίπτωση για τον καθορισμό του εύρους της επανάληψης και στην άλλη για τον έλεγχο ύπαρξης ύπαρξης του γράμματος letter στη λέξη vowels.

Παραπάνω κάνουμε χρήση του ιδιώματος for…in για να επεξεργαστούμε επεξεργαστούμε τους χαρακτήρες χα- ρακτήρες της συμβολοσειράς, έναν κάθε φορά.

8.2 Λίστες

Η **λίστα** είναι μια διατεταγμένη ακολουθία αντικειμένων, όχι απαραίτητα απαραίτητα του ίδιου τύπου και αποτελεί τη **βασική δομή δεδομένων της Python**. Η λίστα, σε αντίθεση με τη συμβολοσειρά, είναι μια *δυναμική δομή* στην οποία μπορούμε να προσθέτουμε προσθέτου- με ή να αφαι- ρούμε αφαιρούμε στοιχεία (mutable). Κάθε αντικείμενο της λίστας χαρακτηρί- ζεται χαρακτηρίζεται από ένα μοναδικό αύξοντα αριθμό, ο οποίος ορίζει τη θέση του στη λίστα, ενώ η προσπέλαση στα στοιχεία της λίστας γίνεται όπως στις συμβολοσειρές, με το όνομα της λίστας και τον αύξοντα αριθμό του αντικείμενου μέσα σε αγκύλες.

Η εντολή L = [3, 5, 8, 13, 21, 34] δημιουργεί τη μεταβλητή L που αναφέρεται στη λίστα [3, 5, 8, 13, 21, 34], όπως φαίνεται στην εικόνα:
εικόνα:



Η αρίθμηση των στοιχείων, όπως στις συμβολοσειρές, έτσι και στις λίστες, ξεκινάει από το 0. Άρα το 1ο στοιχείο της λίστας είναι το L[0], το οποίο είναι ίσο με το 3, το 2ο το L[1] και το τελευταίο το
L[5].

```
>>> L = [ 3, 5, 8, 13, 21, 34 ]
>>> print L[ 0 ]
3
>>> print L[ 5 ]
34
```

Μπορεί κανείς ανά πάσα στιγμή να *προσθέσει, να αφαιρέσει ή να τροποποιήσει* οποιοδήποτε στοιχείο της λίστας.

```
>>> daysofweek = ["Δευτέ-
ρα","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή"]["Δευτέρ
α","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή"]
>>> print daysofweek[ 0 ] + daysofweek[ 4 ]
ΔευτέραΠαρασκευή
>>> daysofweek = daysofweek + ["Σάββατο"]["Σάββατο']
>>> daysofweek[ 0 ] = daysofweek[ 1 ] = "Κυριακή""Κυριακή"
>>> print daysofweek
  ["Κυριακή","Κυριακή","Τετάρτη", "Πέμπτη", "Παρασκευή", “Σάββα- το”]
```

["Κυριακή","Κυριακή","Τετάρτη", "Πέμπτη", "Παρασκευή",
“Σάββατο”]

Έτσι, αν θέλουμε να προσθέσουμε ένα στοιχείο στο τέλος μιας λίστας, γράφουμε:

Λίστα = Λίστα + [στοιχείο]

ενώ στην αρχή της λίστας

Λίστα = [στοιχείο] + Λίστα

Οι λίστες στην Python:

Στην πραγματικότητα όμως οι παραπάνω εντολές δεν προσθέτουν το στοιχείο στην ήδη υπάρχουσα λίστα αλλά δημιουργούν μια νέα λίστα κάθε φορά. Η λειτουργία αυτή έχει σημαντικό υπολογιστικό κόστος. Για αυτό αν θέλουμε να προσθέσουμε ένα στοιχείο στο τέλος της λίστας προτιμούμε τον τελεστή **+=** , όπως φαίνεται παρακάτω:

Λίστα += [στοιχείο]

Οι λίστες στην Python:

- Δεν έχουν σταθερό μέγεθος, δηλαδή μπορούν να αυξάνονται και να μειώνονται κατά την εκτέλεση του προγράμματος.
- Η αρίθμηση των δεικτών ξεκινάει από το 0, όπως ακριβώς στις συμβολοσειρές.
- Είναι *δυναμικές δομές*, και χαρακτηρίζονται από μεγάλη ευελιξία. Έτσι για παράδειγμα, μπορούμε να έχουμε σε μια λίστα ακόμα και στοιχεία διαφορετικού τύπου.

```
>>> mix = [6, 3.14159 , True, “Guido Van Rossum”]
```

```
>>> len(mix)
4
```

Στις λίστες μπορούμε να χρησιμοποιήσουμε τον υπαρξιακό τελεσ- τήτελεστή **in**, τη συνάρτησησυνάρτη ση **len** ή και τον τελεστή συνένωσης '+', ακριβώς όπως στις συμβολοσειρές.

```
>>> fruits = ['apple', 'orange']
>>> len(fruits)
2
>>> print fruits[0]
apple
>>> 'apple' in fruits
True
>>> powers = [2, 4, 8, 16]
>>> fib = [3, 5, 8, 13, 21]
>>> fib + powers
[3, 5, 8, 13, 21, 2, 4, 8, 16]
>>> powers + fruits
[2, 4, 8, 16, 'apple', 'orange']
>>> fib = fib + [ fib[3] + fib[4] ]
>>> print fib
[3, 5, 8, 13, 21, 34]
```

Επειδή οι λίστες και οι συμβολοσειρές ανήκουν και οι δύο σε μια πιο γενική κατηγορίακα τηγορία δομών, τις ακολουθιακές δομές (sequences) της Python, ισχύουν οι ίδιοι τελεστές:

item in List: επιστρέφει True, αν το στοιχείο item υπάρχει μέσα στη λίστα List, αλλιώς επιστρέφει False.

item not in List: επιστρέφει True, αν το στοιχείο item δεν υπάρχει μέσα στη λίστα List, αλλιώς, αν υπάρχει επιστρέφει False.

Εκτός από τους τελεστές που αναφέραμε παραπάνω, υπάρχουν και συναρτήσεις που παίρνουν ως όρισμα μια λίστα. Οι συναρτήσεις των λιστών που θα χρησιμοποιήσουμε σε αυτό το κεφάλαιο είναι η len και η list.

len (List): Επιστρέφει το πλήθος των στοιχείων (ή μέγεθος) της λίστας.

list (String): Επιστρέφει μια λίστα με στοιχεία τους χαρακτήρες της συμβολοσειράς string. Η συνάρτηση αυτή μπορεί να μετατρέψει και άλλα είδη δομών σε λίστα, όπως είναι οι πλειάδες και τα λεξικά.

Οι λίστες, όπως και οι συμβολοσειρές, διαθέτουν μεγάλη ποικιλία μεθόδων, η χρήσηχρή ση των οποίων μπορεί να επεκτείνει, σε μεγάλο βαθμό, τη λειτουργικότητά τους. Στο κεφάλαιο αυτό θα χρησιμο- ποιήσουμεχρησιμοποιήσουμε μόνο τις παρακάτω μεθόδους των λιστών, όπου L το όνομα της λίστας:

L.**append**(object): προσθήκη του στοιχείου object στο τέλος της λίστας L.

L.**insert**(index, object): προσθήκη του στοιχείου object, στη θέση index της λίσταςλί- στας L, μετακινώντας όλα τα στοιχεία από τη θέση index και μετά, κατά μίαμια θέση.

L.**pop**([index]): Αφαίρεση από τη λίστα του στοιχείου που βρίσκεται στη θέση index. Αν δεν δοθεί θέση, τότε θα αφαιρεθεί το τελευταίο στοιχείο της λίστας.

Παρακάτω δίνονται μερικά παραδείγματα κλήσεων των μεθόδων που θα χρησιμοποιήσουμε και δίπλα, μέσα σε σχόλια, η νέα μορφή της λίστας.

```
>>> fib = [5, 8, 13, 21, 34]
>>> fib.pop(1)                    # fib = [5, 13, 21, 34 ]
>>> fib.append(55)                # fib = [5, 13, 21, 34, 55]
>>> fib.pop( )                    # fib = [5, 13, 21, 34]
>>> fib.insert( 2, 89 )           # fib = [5, 13, 89, 21, 34]
```

Αν και η Python ορίζει μια μεγάλη ποικιλία μεθόδων για την επεξεργασία και διαχείριση λιστών, στο βιβλίο αυτό θα χρησιμοποιηθούν μόνο οι παραπάνω, για την επίλυση προβλημάτων.

Διάσχιση Λίστας

Μπορούμε να επεξεργαστούμε τα στοιχεία μιας λίστας, ένα κάθε φορά, κάνοντας χρήση του παρακάτω ιδιώματος της δομής επανάληψης for:

```
for item in List :
  <Εντολές Επεξεργασίας του αντικειμένου item>
```

Παράδειγμα 1. Δημιουργία και εμφάνιση στοιχείων λίστας

Οι παρακάτω εντολές αρχικά κατασκευάζουν μια λίστα η οποία περιέχει όλους τους αριθμούς από το 1 έως και το 6 και στη συνέχεια εμφανίζουν κάθε αριθμό σε διαφορετική γραμμή.

```
L = [1, 2, 3, 4, 5, 6]
for number in L :
    print number
```

Παράδειγμα 2. Μέσος όρων των στοιχείων μιας λίστας

Για να υπολογίσουμε το μέσο όρο των στοιχείων μιας λίστας, πρώτα χρειάζεται να υπολογίσουμε το άθροισμα των στοιχείων, χρησιμοποιώντας μια μεταβλητή στην οποία προσθέτουμε, κάθε φορά, το επόμενο στοιχείο της λίστας:

οποία προσθέτουμε, κάθε φορά, το επόμενο στοιχείο της λίστας:

```
sum = 0.0                # το sum είναι πραγματικός (float)
for number in L :
    sum = sum + number
average = sum / len( L )  # δεν θα γίνει ακέραια διαίρεση
print average
```

Παράδειγμα 3. Μέγιστη τιμή σε μια λίστα

```
maximum = L[0]
for number in L :
    if number > maximum :
    maximum = number
print maximum
```

Παράδειγμα 4. Ρέστα

Καλούμαστε να σχεδιάσουμε το λογισμικό μιας ταμειακής μηχα- νής, το οποίο θα διαβάζει τα ποσό που έδωσε ο πελάτης και το κόστος των αγορών του και θα εμφανίζει το ελάχιστο πλήθος κερ- μάτων ή χαρτονομισμάτων που θα δοθούν για ρέστα. Θεωρήστε ότι όλες οι τιμές είναι σε ακέραια πολλαπλάσια του ευρώ:

```
values = [100, 50, 20, 10, 5, 2, 1]
cost = input( “Δώσε το κόστος των αγορών” )
payment = input( “Δώσε το ποσό της πληρωμής” )
change = payment – cost
counter = 0
for value in values :
    counter = counter + ( change / value )
    change = change % value
print counter
```

Εφαρμογή.: Διαχωρισμός λίστας

Η λειτουργία του διαχωρισμού μιας λίστας σε δύο μέρη με βάση κάποια κριτήρια, αποτελεί μια τυπική επεξεργασία των λιστών.

```
positives = [ ]
negatives = [ ]
for number in numbers :        # για κάθε αριθμό της λίστας
   if number > 0 :             # αν είναι θετικός
      positives.append( = positives + [ number )]
      # πρόσθεσέ τον στη λίστα με τους θετικούς
   else :
      negatives.append( = negatives + [ number )]
      # αλλιώς στη λίστα με τους αρνητικούς
print  positives
print  negatives
```

Το παραπάνω πρόγραμμα διαχωρίζει τους αριθμούς μιας λίστας σε αρνητικούς και θετικούς. Υποθέτουμε ότι όλοι οι αριθμοί είναι διάφοροι του μηδενός. Να σημειωθεί ότι η αρχική λίστα παραμένει ανέπαφη.

Εφαρμογή.: Συγχώνευση διατεταγμένων λιστών

Ένας από τους πιο γνωστούς και χρήσιμους αλγορίθμους της Πληροφορικής είναι ο αλγόριθμος της συγχώνευσης των στοιχεί- ωνστοιχείων δυο ταξινομημένων λιστών σε μία νέα, επίσης ταξινομημένη, λίστα. Ο αλγόριθμος αξιοποιεί το γεγονός ότι οι αρχικές λίστες εί- ναιείναι ήδη ταξινομημένες, ώστε να μη χρειαστεί να ταξινομήσει από την αρχή την τελική λίστα, κάτι το οποίο έχει σημαντικό υπολογισ- υπολογιστικό κόστος για πολύ μεγά- λες λίστες.
τικό κόστος για πολύ μεγάλες λίστες.

```
def merge( A, B ) :
   L = [ ]
   while A != [ ] and B != [ ] :  # όσο και οι δυο λίστες έχουν στοιχεία
   while  A != [ ]  and  B != [ ] :
```

```
if A[0] < B[0] :  # Αν το 1ο~~πρώτο~~ στοιχείο της Α είναι το μικρότερο
    L.append(~~= L +~~ [ A.pop(0) ~~]~~# το μεταφέρουμε στο τέλος της~~στην~~ L
```

```
    else :
        L.append( L + [ B.pop(0) ])
        # αλλιώς μεταφέρουμε το πρώτο στοιχείο της B στην L
return L + A + B
# στο τέλος προσθέτουμε τα στοιχεία που έχουν μείνει
```

Αφού οι δύο λίστες είναι ταξινομημένες σε αύξουσα σειρά το ελάχιστοελάχιστο στοιχείο και των δύο λιστών είναι το μικρότερο από τα A[0], B[0]. Στη συνέχεια μεταφέρουμε αυτό το στοιχείο στην τελική λίστα.

Κάποια στιγμή, μία από τις δυο λίστες θα αδειάσει, οπότε η επανάληψηεπανάληψη θα σταματήσειστάμα- τήσει. Η άλλη λίστα όμως θα έχει κάποια στοιχε-

ίαστοιχεία τα οποία πρέπει να προστεθούν στο τέλος της τρίτης λίστας. Έτσι θα έπρεπε να γράψουμε:

```
if A == [ ] :
   L = L + B
else :
   L = L + A
```

L = L + B + A

Οπότε αν η A είναι κενή το αποτελέσματα είναι L + B + [ ], ενώ αν η B είναι κενή το αποτέλεσμα είναι L + [ ] + A.

Η Ή συνάρτηση range (έχει αναλυθεί και στην 4.1.3)

Η συνάρτηση range επιστρέφει, αν δώσουμε την αρχική, την τελι- κήτελική τιμή και το βήμα, μια λίστα από αριθμούς, όπως φαίνεται πα- ρακάτωπαρακάτω:

```
>>> range( 4 )
[0, 1, 2, 3]
>>> range( 0, 4 )
[0, 1, 2, 3]
>>> range( 0, 4, 1 )
[0, 1, 2, 3]
>>> range( 10, 30, 5 )
[10, 15, 20, 25]
>>> range( 30, 10, –5 )
[30, 25, 20, 15]
>>> range( 1, 2, 100 )
[1]
>>> range(
1, 1, 100 ) [ ]
>>> range(
1, 5, –1 ) [ ]
>>> range( 0
)
[ ]
```

```
>>> range( 1, 1, 100 )
[]
>>> range( 1, 5, -1 )
[]
>>> range( 0 )
[]
```

Η συνάρτηση **range**(A, M, B) επιστρέφει μια λίστα αριθμών ξεκινώντας με τον αριθμό A μέχρι το M με βήμα B. Το M δεν συμπεριλαμβάνεται στη λίστα.

Έτσι τα παρακάτω τμήματα κώδικα εκτελούν την ίδια λειτουργία:

```
>>> L = [ 6, 28, 496, 8128 ]
>>> for item in L :
        print item ,
6 28 496 8128
```

```
>>> L = [ 6, 28, 496, 8128 ]
>>> for index in range(0,4) :
        print L[index] ,
6 28 496 8128
```

```
>>> L = [ 6, 28, 496, 8128 ]
>>> for index in [ 0, 1, 2, 3 ] :
        print L[index] ,
6 28 496 8128
```

Η range μας διευκολύνει επίσης στην εκτέλεση ενός τμήματος εντολώνεντολών για έναν προκαθορισμένο αριθμό επαναλήψεων, όπως φαίνεταιφαίνεται παρακάτω:

```
>>> sum = 0
>>> for i in range(101100) :
        sum = sum + i
>>> print sum
5050
```

```
>>> for index in range(2,11,2) :
        print index ,
2 4 6 8 10
```

Ο τελεστής διαμέρισης :

Ένας πολύ χρήσιμος τελεστής είναι ο **τελεστής διαμέρισης** (slice operator) που συμβολίζεται με την άνω κάτω τελεία **:**, και ο οποίος έχει αναφερθεί στις παραγράφους 5.2, 5.3 του βιβλίου της Β' τάξης. Ο τελεστής διαμέρισης μπορεί να μας επιστρέψει ένα τμήμα μιας συμβολοσειράς ή μιας λίστας. Η έκφραση word[a **:** b] μας επιστρέφει το τμήμα της συμβολοσειράς ή της λίστας από το στοιχείο word[a] μέχρι και το στοιχείο word[b-1]. Μπορούμε να παραλείψουμε την αρχή ή το τέλος όπως φαίνεται στα παραδείγματα

παρακάτω αν θέλουμε ένα τμήμα από την αρχή ή το τέλος της λίστας/συμβολοσειράς αντίστοιχα.

Δίνονται τα παρακάτω παραδείγματα:

```
>>> word = “zanneio gymnasio”
>>> word[ :7 ]
‘zanneio’
>>> word[ 8: ]
‘zanneio’
>>> word[ : ]
‘zanneio gymnasio’
>>> word[ 3:11 ]
‘neio gym’
>>> word[ 0:7 ]
‘zanneio’
>>> word[ 8:len(word) ]
‘gymnasio’
>>> word[ 0:len(word) ]
‘zanneio gymnasio’
```

Ο τελεστής **:** είναι πολύ σημαντικός στην επεξεργασία λιστών στην Python, γιατί μπορούμε να τον χρησιμοποιήσουμε για να πάρουμε ένα αντίγραφο μιας λίστας όπως παρακάτω:

```
>>> fibonacci = [5,8,13,21,34]
>>> fib = fibonacci[ : ]        # δημιουργεί αντίγραφο
>>> a = fib                     # δείχνουν στην ίδια λίστα
>>> a.pop()                     # οι αλλαγές επηρεάζουν και την a και τη fib
34
>>> fib.pop()
21
>>> a[0]=a[1]=55
>>> print a
[55, 55, 13]
>>> print fib
[55, 55, 13]
>>> print fibonacci
```

```
# Η αρχική λίστα fibonacci δεν έχει αλλάξει
[5, 8, 13, 21, 34]
```

Προσοχή! Δεν μπορεί να γίνει το ίδιο με τις συμβολοσειρές. Δη- λαδή η εντολή word[:] δεν δημιουργεί αντίγραφο της συμβολοσει- ράς word. Αυτό μπορείτε να το διαπιστώσετε με χρήση της συνάρ- τησης id που δείχνει τη διεύθυνση στη μνήμη του κάθε αντικειμέ- νου.

Ο τελεστής διαμέρισης μπορεί να φανεί πολύ χρήσιμος σε κάποιες περιπτώσεις όπως για παράδειγμα στην δραστηριότητα 8 του κε- φαλαίου 5 του βιβλίου της Β' τάξης όπου ζητείται η υλοποίηση του αλγορίθμου κρυπτογράφησης του Καίσαρα. Αν κάθε γράμμα αν- τιστοιχίζεται με το γράμμα που βρίσκεται 3 θέσεις μπροστά τότε το νέο αλφάβητο μπορεί να προκύψει με τις παρακάτω εντολές:

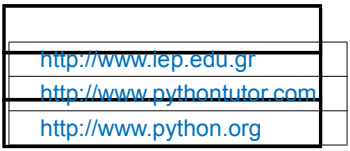
```
>>> alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
>>> cipherAlphabet = alphabet[3 : ] + alphabet[ : 3]
>>> print  cipherAlphabet
DEFGHIJKLMNOPQRSTUVWXYZABC
```

8.3 Στοίβα

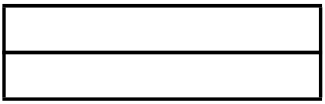
Εισαγωγική Δραστηριότητα

Όλοι χρησιμοποιούμε διάφορα προγράμματα για πλοήγηση στον παγκόσμιο ιστό, τα οποία είναι γνωστά ως φυλλομετρητές (browsers) και πολλές φορές, κατά την πλοήγηση, θέλουμε να επιστρέψουμε~~επιστρέψουμε~~ στην αμέσως προηγούμενη ιστοσελίδα στην οποία βρισκόμασταν. Γι'Γι' αυτό υπάρχει η λειτουργία **Πίσω** (Back) σε όλο- υς~~όλους~~ τους φυλλομετρητές, η οποία μας μεταφέρει στην προηγούμενη ιστοσελίδα. Για να επιτευχθεί~~επι- τευχθεί~~ αυτή η λειτουργία, το περιβάλλον διατηρεί ένα ιστορικό των ιστοσελίδων τις οποίες επισκεφθήκαμε. Η δομή δεδομένων που χρησιμοποιείται για την αποθήκευση~~αποθήκευ- ση~~ του ιστορικού, πρέπει να είναι τέτοια, ώστε οι ιστοσελίδες να ανακτών- ται~~ανακτώνται~~ με την αντίστροφη σειρά επίσκεψης. Δηλαδή στην πρώτη θέση θέλουμε να είναι η ιστοσελίδα~~ιστοσε- λίδα~~ που επισκεφτήκαμε πιο πρόσφατα και στην τελευταία η αρχική ιστοσελίδα από την οποία ξεκινήσαμε.

Ας υποθέσουμε για παράδειγμα ότι, ξεκινώντας από την ιστοσελί- δα ~~ιστοσελίδα~~ http://www.python.org μεταβαίνουμε στην ιστοσελίδα http://www.pythontutor.com και στη~~στην~~ συνέχεια στην ιστοσελίδα http://www.iep.edu.gr. . Το ιστορικό των επισκέψεων~~επισκέ- ψεων~~ μέχρι στιγμής περιέχει τις τρεις αυτές ιστοσελίδες όπως φαίνεται παρακάτω:



Η πρώτη ιστοσελίδα στην οποία έχουμε πρόσβαση είναι η τελευ- ταία~~τελευταία~~ που επισκεφτήκαμε~~επισκε- φτήκαμε~~ και βρίσκεται πάνω – πάνω στο ιστορικό. Αν μεταβούμε στην προηγούμενη~~προηγούμε νη~~ ιστοσελίδα, τότε το ιστορικό θα πάρει τη μορφή:

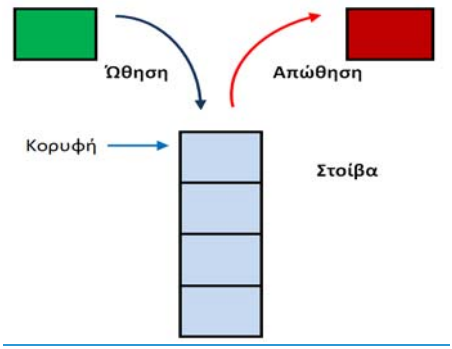


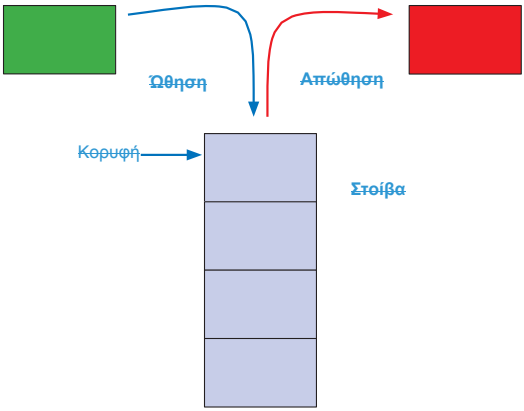
http://www.pythontutor.com
http://www.python.org

Αν στην συνέχεια μεταβούμε στην ιστοσελίδα http://www.minedu.gov.gr/, , τότε αυτή θα προστεθεί στην κορυφή του ιστορικού:

http://www.minedu.gov.gr/
http://www.pythontutor.com
~~http://www.python.org~~

Από τα παραπάνω μπορούμε να συμπεράνουμε ότι η δομή που χρησιμοποιείται από το ιστορικό, είναι μια λίστα στην οποία οι ει- σαγωγέςεισαγωγές και οι διαγραφές στοιχείωνστοιχεί- ων γίνονται από το πάνω άκρο μόνο. Στη λίστα αυτή το στοιχείο που προστέθηκε τελευταίο είναι και το πρώτο που θα εξαχθεί, έχουμε δηλαδή μια λειτουργία τύ- πουτύπου LIFO (**L**ast In **F**irst **O**ut), δηλαδή ο τελευταίος που εισέρχεται στη λίστα, είναι και ο πρώτος που θα εξαχθεί. Η συγκεκριμένη δομή δεδομένων ονομάζεται **Στοίβαστοίβα** και οι λειτουργίες εισαγωγής και εξαγωγής είναι γνωστές στη βιβλιογραφία ως **ώθηση** και **α- πώθησηαπώθηση**.





Εικόνα 8 -1. Οι βασικές λειτουργίες σε μια στοίβα είναι η ώθη- σηώθηση και η απώθηση

Ένα άλλο παράδειγμα που χρησιμοποιείται συχνά για να περιγ- ράψειπεριγράψει μια στοίβα, είναι αυτό του σωρού με πιάτα που περιμένουν να πλυθούν. Κάθε νέο πιάτο που έρχεται για πλύσιμο τοποθετείται πάνω στον σωρό, με αποτέλεσμα να είναι το πρώτοπρώ- το που θα πλυ- θείπλυθεί.

Η στοίβα αποτελεί μια από τις σημαντικότερες δομές δεδομένων της επιστήμης της Πληροφορικής και χρησιμοποιείται σε πολλά πεδία της, όπως είναι η θεωρία αλγορίθμωναλγο- ρίθμων, η ανάπτυξη μεταγ- λωττιστώνμεταγλωττιστών, η τεχνητή νοημοσύνη κ.ά.

Όταν η στοίβα είναι άδεια, είναι προφανές ότι δεν μπορεί να γίνει απώθηση. Άρα, όταν απωθούμε ένα στοιχείο από τη στοίβα, θα πρέπει προηγουμένως να έχουμε εξασφαλίσει ότι η στοίβα δεν είναι κενή. Για αυτό το λόγο, εκτός από την ώθηση και

την απώ- θησηαπώθηση, πρέπει να υλοποιήσουμε και τον έλεγχο, αν η στοίβα είναι κενή. Άρα οι βασικές λειτουργίες που πρέπει να υποστηρίζει η υ-λοποίησηυλοποίηση μιας στοίβας είναι:

- Δημιουργία μιας κενής στοίβας.
- Έλεγχος, αν η στοίβα είναι κενή.
- Ώθηση ενός στοιχείου στη στοίβα.
- Απώθηση ενός στοιχείου από τη στοίβα.

Η δομή της στοίβας μπορεί να υλοποιηθεί στην Python με μια λίστα~~λίστα~~ στην οποία οι εισαγωγές και οι εξαγωγές στοιχείων γίνονται μό-

νομόνο από το ένα άκρο. Παρακάτω δίνονται δύο υλοποιήσεις της στοίβαςστοίβας. Στην πρώτη περίπτωση, οι εισαγωγές/διαγραφέςδι- αγραφές στοιχείων γίνονται στο τέλοςπίσω μέρος της λίστας, ενώ στη δεύτερη, στην αρχή τηςαπό το εμπρός μέρος.

Υλοποίηση Στοίβας σε Python με δύο τρόπους

```
def push(stack, item) :
    stack.append( =
    stack + [item )]
def pop(stack) :
    return stack.pop( )
def isEmpty(stack) :
    return len(stack) == 0
def createStack( ) :
    return [ ]
```

```
def push(stack, item) :
    stack.insert(0, = [item)
  ] + stack def
pop(stack) :
    return stack.pop( 0 )
def isEmpty(stack) :
    return len(stack) == 0
def createStack( ) :
    return [ ]
```

Οι δύο υλοποιήσεις που δίνονται παραπάνω, διαφέρουν μόνο ως προς το σημείο της λίστας στο οποίο γίνονται οι εισαγω- γέςεισαγωγές/διαγραφές των στοιχείων.

Εφαρμογή Στοίβας.: Αντιστροφή αριθμών

Το παρακάτω πρόγραμμα δέχεται από το χρήστη αριθμούς μέχρι να δοθεί το 0 και τους εμφανίζει σε αντίστροφη σειρά από αυτή με την οποία δόθηκαν. Θέλουμε κάθε φορά να εμφανίσουμε πρώτο τον αριθμό που δόθηκε τελευταίος. Χρειαζόμαστε μια δομή δεδο-δεδομένων που να υποστηρίζει τη λειτουργία LIFO, όπως η στοίβα.
μένων που να υποστηρίζει τη λειτουργία LIFO, όπως η στοίβα.

```
stack = createStack()              # Δημιουργία της στοίβας stack
number = int( raw_input( ) )       # Διάβασε τον πρώτο αριθμό
while number != 0 :                # Όσο δεν δίνεται 0
    push(stack, number)            # Σπρώξε τον αριθμό στη στοίβα
    number = int( raw_input( ) )   # Διάβασε τον επόμενο αριθμό
```

```
while not isEmpty( stack ) :        # Μέχρι να αδειάσει η
    στοίβα number = pop(stack)      # βγάλε τον επόμενο
    αριθμό print number             # και εκτύπωσέ τον
```

Παρατηρήστε ότι στο παραπάνω πρόγραμμα δεν φαίνεται ποια υλοποίηση χρησιμοποιείται. Αν τροποποιήσουμε την υλοποίηση της ώθησης, δε θα χρειαστεί να αλλάξουμε τίποτα στο πρόγραμμα. Αυτό είναι γνωστό ως διαχωρισμός **διεπαφής (διασύνδεσης) – υλοποίησης (interface – implementation)**, αφού οι εφαρμογές που χρησιμοποιούν δομές δεδομένων όπως η στοίβα, είναι απολύτως ανεξάρτητες από την υλοποίηση.

Ουσιαστικά δε μας ενδιαφέρει πώς έχει υλοποιηθεί η στοίβα, αφού εμείς θέλουμε μόνο να χρησιμοποιήσουμε τις συναρτήσεις που έχουμε ορίσει παραπάνω.

Το σύνολο των επικεφαλίδων των συναρτήσεων το οποίο είναι διαθέσιμο στον προγραμματιστή που χρησιμοποιεί τη στοίβα, το ονομάζουμε διεπαφή ή διασύνδεση (interface) της δομής αυτής. Η διεπαφή μιας δομής ορίζει τι μπορεί να κάνει η δομή και όχι τον τρόπο με τον οποίο το κάνει. Το τελευταίο είναι ο ρόλος της υλοποίησης.

Διεπαφή της δομής δεδομένων Στοίβα
def push(stack, item)
def pop(stack)
def isEmpty(stack)
def createStack( )

8.4 Ουρά

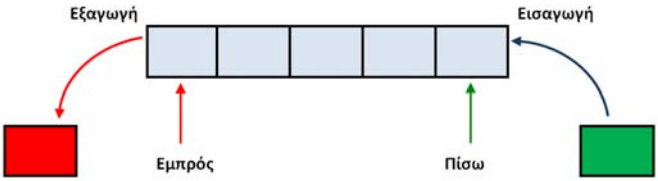
Μια δομή δεδομένων που χρησιμοποιείται για την μοντελοποίηση και προσομοίωση πραγματικών φαινομένων, είναι η δομή της ουράς. Τα φαινόμενα που μοντελοποιούνται αναφέρονται στην εξυπηρέτηση ανθρώπων, αντικειμένων ή προγραμμάτων. Τέτοια παραδείγματα ουρών είναι:

•

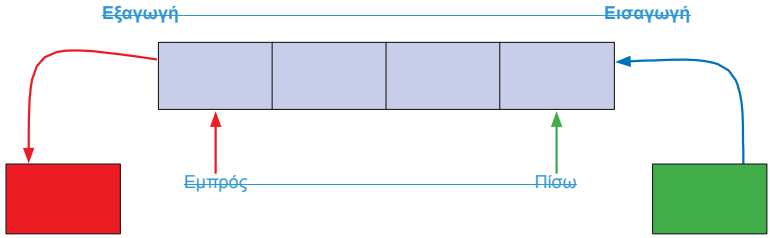
- Οι ουρές στις τράπεζες και τα σούπερ-μάρκετ.

- Η ουρά των προγραμμάτων που περιμένουν να εξυπηρετηθούν από τον επεξεργαστήεπε- ξεργαστή του υπολογιστή σας.
- Η ουρά των αιτήσεων προς το διακομιστή ιστού (web server) ενός δικτυακού τόπου.

Τα φαινόμενα αυτά μελετώνται από διάφορους κλάδους των Μαθηματικώνμαθηματικών και της Πληροφορικής, όπως είναι η Θεωρία Ουρών(θεωρία ουρών (Queueing theory) και η Επιχειρησιακή Έρευναέρευνα



ι ως LIFO (Last In
α ως FIFO (First In
η σειρά που έφτασε



Εικόνα 8-2

Δύο είναι οι βασικές λειτουργίες μιας ουράς:

- Εισαγωγή στοιχείου, η οποία γίνεται στο πίσω μέρος της ου- ράςουράς.
- Εξαγωγή στοιχείου, η οποία γίνεται από το εμπρός μέρος της ουράς.

ουράς.

Υλοποίηση Ουράς σε Python

```
def enqueue(queue, item) :
    queue = queue.append(-
        + [ item ])
def dequeue(queue) :
    return queue.pop( 0 )
def isEmpty(queue) :
    return
len(queuequere) == 0 def
createQueuecreateQuere
```

```
( ) :
    return [ ]
```

8.5 Πλειάδες

Οι **πλειάδες** είναι σύνθετες δομές της Python, οι οποίες χρησιμοποιούνται για την αναπαράσταση οντοτήτων του πραγματικού κόσμου. Χαρακτηρίζονται από κάποιες ιδιότητες, όπως για παράδειγμα όνομα, επώνυμο, βαθμό, μισθό κ.λπ. Οι πλειάδες είναι ακολουθίες όπως οι λίστες, όμως δεν μπορούν να τροποποιηθούν. Ανήκουν στην κατηγορία των *μη τροποποιήσιμων* (immutables) δομών της Python, όπως είναι οι συμβολοσειρές. Στις πλειάδες τα στοιχεία χωρίζονται με κόμματα, όπως στις λίστες, αλλά, αντί να περικλείονται από αγκύλες, περικλείονται σε παρενθέσεις, οι οποίες όμως δεν είναι υποχρεωτικές.

Η πλειάδα μοιάζει με μια λίστα, όπως φαίνεται στα ακόλουθα παραδείγματα:

```
>>> rec = (“Edsger”, “Dijkstra”, 1940, 2006)
>>> print rec[0], rec[1], rec[2], rec[3]
Edsger Dijkstra 1830 1879
>>> len( rec )
4
```

Μια γνωστή εφαρμογή τους είναι η αμοιβαία αλλαγή των τιμών δυο μεταβλητών, χωρίς τη χρήση βοηθητικής μεταβλητής, λειτουργία η οποία είναι γνωστή ως *αντιμετάθεση* (swap):

```
>>> a = 496
>>> b = 28
>>> a , b = b, a          # αντιμετάθεση των τιμών των μεταβλητών
a, b
>>> print “a =”, a, “b =”, b
a = 28 b = 496
```

8.6 Λεξικά

Το **λεξικό** είναι μια εξαιρετικά χρήσιμη ενσωματωμένη (built-in) δομή της Python. Φανταστείτε το σαν ένα τηλεφωνικό κατάλογο ο οποίος μας δίνει τη δυνατότητα να βρούμε πολύ γρήγορα τα στοι- χεία κάποιου, μόνο από το όνομά του. Για παράδειγμα, μπορούμε να γράψουμε version["python"] = 3.4. Όπως σε μια λίστα η θέση κάθε στοιχείου είναι μοναδική και δεν μπορεί να περιέχει ταυτόχ- ρονα δυο τιμές, έτσι και στο λεξικό, η λέξη-κλειδί που χρησιμοποι- ούμε μέσα στις αγκύλες έχει μοναδική τιμή και εμφανίζεται το πολύ μία φορά. Είναι το λεγόμενο **κλειδί**.

Έτσι, ένα λεξικό είναι ουσιαστικά ένα σύνολο ζευγών κλειδιών-τιμών, όπου κάθε κλειδί δεν εμφανίζεται δεύτερη φορά. Μπορούμε να ορίσουμε ένα λεξικό όπως παρακάτω:

```
>>> empty_dictionary = dict( )  # δημιουργία ενός άδειου λεξικού
>>> print empty_dictionary
   { }
{}
>>> dictionary = { 'Kilo':3, 'Mega':6, 'Giga':9, 'Tera':12 }
>>> print dictionary
{ 'Kilo': 3, 'Mega': 6, 'Giga': 9, 'Tera': 12 }
>>> dictionary ['Giga']      # έτσι βρίσκω την τιμή του Giga
9
>>> len(dictionary )        # το πλήθος των κλειδιών του λεξικού
4
```

Για να προσθέσουμε ένα ζεύγος κλειδί-τιμή στο λεξικό αρκεί να δώσουμε την αντίστοιχη εντολή ανάθεσης τιμής. Σε μια εντολή ανάθεσης τιμής :

Λεξικό[Κλειδί] = Τιμή

Αν το κλειδί υπάρχει ήδη στο λεξικό, η τιμή που είχε ενημερώνεται και στη θέση της μπαίνει η νέα τιμή. Τιμή, ενώ αν δεν υπάρχει, τότε δημιουργείται ένα νέο ζεύγος *Κλειδί:Τιμή*. Παρακάτω προσθέ- τουμε στο λεξικό το κλειδί *Peta* και με τιμή 15 και στη συνέχεια τροποποιούμε την τιμή του *Kilo* σε 1024.

Κλειδί:Τιμή. Παρακάτω προσθέτουμε στο λεξικό το κλειδί 'Peta' και με τιμή 15 και στην συνέχεια τροποποιούμε την τιμή του 'Kilo' σε 1024.

```
>>> dictionary ['Peta'] = 15
>>> print dictionary
{ 'Kilo': 3, 'Peta': 15, 'Mega': 6, 'Giga': 9, 'Tera': 12 }
>>> dictionary ['Kilo'] = 1024
>>> print dictionary
{ 'Kilo': 1024, 'Peta': 15, 'Mega': 6, 'Giga': 9, 'Tera': 12 }
```

Παρατηρήστε ότι τα στοιχεία του λεξικού δεν είναι διατεταγμένα και το νέο στοιχείο μπήκε σε μία τυχαία θέση. Στα λεξικά, όπως και στις λίστες και τις συμβολοσειρές, μπορούν να χρησιμοποιηθούν οι τελεστές **in** και **not in** για τον έλεγχο της ύπαρξης ενός κλειδιού στο λεξικό. Επίσης, για τη διαγραφή ενός κλειδιού από το λεξικό, χρησιμοποιείται ο τελεστής **del**.

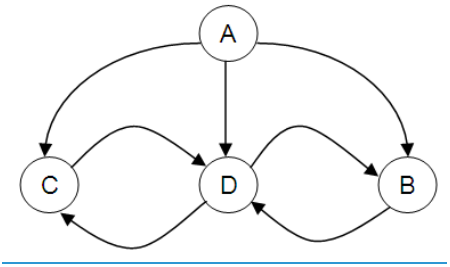
χρησιμοποιείται ο τελεστής **del**.

```
>>> del dictionary ['Kilo']
>>> del dictionary ['Mega']
>>> print dictionary
{ 'Peta': 15, 'Giga': 9, 'Tera': 12 }
```

```
>>> 'Kilo' in dictionary
False
>>> 'Peta' in dictionary
True
```

8.7 Εισαγωγή στους γράφους και τα δέντρα

Ένας **Γράφοςγράφος** είναι μια δομή που αποτελείται από ένα σύνολο *κορυφών* ή *κόμβων* και ένα σύνολο *ακμών* μεταξύ των κορυφών. Οι κορυφές θα μπορούσαν να είναι οι πόλεις μιας χώρας και οι ακμές οι δρόμοι που τις ενώνουν. Όταν οι δρόμοι είναι διπλής κυκλοφορίας, λέμε ότι ο γράφος είναι **μη κατευθυνόμενος**, ενώ, όταν κάποιοικά ποιοι από τους δρόμους είναι μονόδρομοι, λέμε ότι ο γράφοςγράφος είναι *κατευθυνόμενος*. Παρακάτω δίνεται ένας κατευθυνόμενοςκατευθυνόμενος γράφος με 4 κόμβους A, B, C, D.



Εικόνα 8-3

Για να μεταβούμε από την κορυφή C στη Β δεν υπάρχει απευθείας ακμή. Πρέπει πρώτα να περάσουμε από τη D. Η ακολουθία των ακμών από τις οποίες περνάμε για να φτάσουμε στον τελικό προ- ορισμόπροορισμό, λέγεται *μονοπάτι*. Το μονοπάτι από τη C στη Β συμβολίζε- ταισυμβολίζεται C →ο D →ο Β και έχει μήκος 2 αφού αποτελείται από 2 ακμές. Δηλαδή, το *μήκος ενός μονοπατιού* είναι το πλήθος των ακμών του μονοπατιού. Σε αυτό το βιβλίο, κάνουμε την υπόθεση ότι όλες οι ακμές έχουν το ίδιο μήκος ή κόστος.

Παρατηρήστε ότι δεν υπάρχει κανένα μονοπάτι από κάποια κορυ- φήκορυφή που να καταλήγεικατα λήγει στην Α. Ωστόσο, αν ακολουθήσουμε το μονοπάτιμονοπάτι Β →ο D →ο C, μπορούμε να επιστρέψουμε πίσω στη Β, αφούαφού υπάρχει δρόμος C →ο D →ο Β. Αυτό ονομάζεται *κύκλοςκύ- κλος* ή *κυκλικόκυκλικό μονοπάτι*, επειδή, αν το ακολουθήσουμε επιστρέφουμε στο σημείο από το οποίο ξεκινήσαμε.

Η δομή ενός γράφου συναντάται σε πολλά προβλήματα της επισ- τήμηςεπιστήμης της ΠληροφορικήςΠλη- ροφορικής, όπως για παράδειγμα στην αναπαράσ- τασηαναπαράσταση του παγκόσμιου ιστού, την κατάταξη των ιστοσελίδων (page ranking), την εύρεση ενός μονοπατιού ή του συντομότερου μονο- πατιούσυ- ντομότερου μονοπατιού σε ένα δίκτυο πόλεων και άλλα. Μια ειδική περίπτωση γράφωνγρά- φων είναι τα δέντρα.

Ένας γράφος στον οποίο υπάρχει μονοπάτι μεταξύ δυο οποιωνδήποτε κορυφώνκορυ- φών, και δεν περιέχει κυκλικές διαδ- ρομέςδιαδρομές, λέγεται **Δέντροδέντρο**.

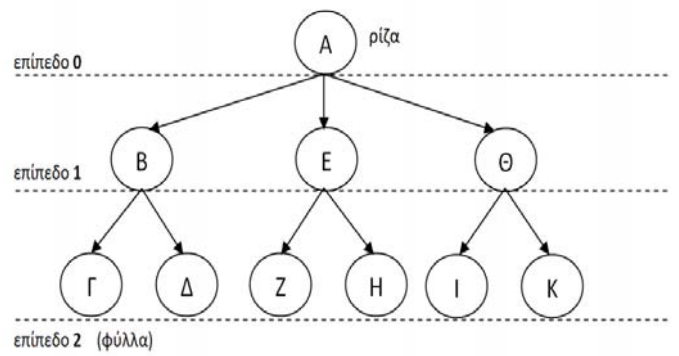
Το *δέντρο* συνιστά μια ιεραρχική δομή η οποία χρησιμοποιείται για τη μοντελοποίησημοντελοποί ηση διάφορων καταστάσεων, όπως για παράδειγμα ένα γενεαλογικό δέντρο ή η οργανωτική δομή μιας επιχείρησης. Αναφερόμαστε στα στοιχεία του δέντρου, όπως και του γράφου, ως *κόμβους*. Κάθε κόμβος συνδέεται με έναν ή περισσότερους κόμβους στους οποίους αναφερόμαστε, ως *παιδιά του* ή **απόγο- νούςαπόγονούς του**. Οι κόμβοικόμ βοι που δεν έχουν απογόνους και βρίσκονται στο τελευταίο επίπεδο λέγονται *φύλλα* του δέντρου. Ο αριθμός των παιδιών που έχει ένας κόμβος ονομάζεται *βαθμός* του κόμβου. ΟΟ μέγιστος βαθμός μεταξύ των κόμβων ενός δέντρου είναι ο *βαθμός του δέντρου*.

μέγιστος βαθμός μεταξύ των κόμβων ενός δέντρου είναι ο *βαθμός του δέντρου*.

Κάθε κόμβος έχει ακριβώς έναν πρόγονο, εκτός από τη **ρίζα**, που βρίσκεται συνήθωςσυ- νήθως στην κορυφή του δέντρου. Η απόσταση ενός κόμβου από τη ρίζα είναι το *επίπεδο* του κόμβου. Προφανώς, η ρίζα βρίσκεται στο μηδενικό (0) επίπεδο.

Το *ύψος* ενός δέντρου είναι η μέγιστη απόσταση κάποιου κόμβου από τη ρίζα. Όπως φαίνεται στο παρακάτω σχήμα, μέσω των επι- πέδωνεπιπέδων, διακρίνεται πιο εύκολα η ιεραρχική δομή του δέντρου. Το δέντρο του σχήματος έχει ύψος 2, αφού το τελευταίοτελευ- ταίο επίπεδο είναι το 2ο.

το $2^{ο}$.



Εικόνα 8-4. Ιεραρχική δομή του δέντρου

Δραστηριότητες - Ερωτήσεις - Ασκήσεις κεφαλαίου

8.8~~.8~~
Δραστηριότητες

Δραστηριότητα 1

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα εμφανίζει τα γράμματάγράμ~~-~~ματά της, ένα σε κάθε γραμμή.

Δραστηριότητα 2

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα εμφανίζει πόσα κεφαλαία αγγλικά γράμματα περιέχει η λέξη.

Δραστηριότητα 3

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα την εμφανίζει αντεστραμμένη, με τη χρήση μιας στοίβας.

Δραστηριότητα 4

Να γράψετε μια συνάρτηση *isSubstring(string, substring)* η οποία θα ελέγχει, αν η συμβολοσειρά substring περιέχεται στη συμβολο- σειράστην~~-~~ συμβολοσειρά string και αν ναι, θα επιστρέφειεπι~~-~~ στρέφει True. Στη συνέχεια να υλο- ποιήσετε μια δεύτερηχρησιμοποιήσετε αυτή τη συνάρτηση, η οποία θα επιστρέφειγια να βρείτε πόσες φορές εμφανίζεται μια συμβολοσειρά μέσα σε μια άλλη.

Δραστηριότητα 5

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο, μέχρι να δοθεί ο αριθμός 0. Κάθε φορά που θα διαβάζει έναν θετικό αριθμό, θα τον προσθέτει σε μια στοίβα. Όταν διαβάζει έναν αρνητικό αριθμό θα αφαιρεί τόσους αριθμούς από τη στοίβα, αν αυτόόσο είναι δυνατόν καιη τιμή του αριθμού. Ο αλγόριθμος θα τους εμφανίζει στην οθό- νητερματίζει όταν αδειάσει η στοίβα.

Δραστηριότητα 6

Να γράψετε πρόγραμμα στη γλώσσα Python το οποίο θα δέχεται ως είσοδο ένα κείμενο και θα εμφανίζει πόσες φορές εμφανίζεται κάθε γράμμα του αγγλικού αλφαβήτουαλφα~~-~~ βήτου σε αυτό. Να χρησιμοποιή- σετεχρησιμοποιήσετε ένα λεξικό.

Δραστηριότητα 7

Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίσ- ταλίστα από λέξεις και θα επιστρέφει τη λέξη με το μεγαλύτερο μήκος.

Δραστηριότητα 8

Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λέξη και θα επιστρέφει ~~επιστρέφει~~ το πλήθος των φωνηέντων που έχει. Στη συνέχεια ~~συνέχεια~~, να γράψετε μια δεύτερη συνάρτηση, η οποία θα δέχεται μια λίστα από λέξεις και θα επιστρέφει τη λέξη με τα περισσότερα φωνήεντα ~~φωνήεντα~~.

8.9 Ερωτήσεις

1. Να περιγράψετε τη δομή δεδομένων συμβολοσειρά ~~"Συμβολοσειρά"~~ (str) στην Python και να αναφέρετε τους τελεστές επεξεργασίας της.
2. Να περιγράψετε τη δομή δεδομένων ~~"~~Λίστα~~"~~ στην Python και να αναφέρετε τους τελεστές επεξεργασίας της.
3. Να αναφέρετε τις δομές δεδομένων οι οποίες επιτρέπουν τροποποίηση των δεδομένων τους (mutables).
4. Να αναφέρετε τις δομές δεδομένων οι οποίες δεν επιτρέπουν ~~επιτρέπουν~~ τροποποίηση των δεδομένων τους (immutables).
5. Πότε χρησιμοποιούμε τη δομή δεδομένων του Λεξικού ~~λεξικού~~;
6. Να αναφέρετε μια εφαρμογή των Πλειάδων ~~πλειάδων~~.
7. Ποιες είναι οι βασικές λειτουργίες σε μια Στοίβα ~~στοίβα~~ και ποιες σε μια Ουρά ~~ουρά~~;
8. Να αναφέρετε εφαρμογές της Στοίβας ~~στοίβας~~.
9. Να αναφέρετε εφαρμογές της Ουράς ~~ουράς~~.
10. Σε τι διαφέρει ένα Δέντρο ~~δέντρο~~ από ένα Γράφο; ~~γράφο;~~

Σύνοψη

Στο κεφάλαιο αυτό παρουσιάστηκαν οι βασικές δομές δεδομένων της Python που είναι ενσωματωμένες στη γλώσσα και χωρίζονται σε δυο κατηγορίες. Αυτές που επιτρέπουν ~~επι- τρέπουν~~ την τροποποίηση των περιεχομένων τους (immutables), όπως οι λίστες και τα λεξικά και αυτές που δεν το επιτρέπουν, όπως οι συμβολοσειρές και οι πλει- άδες ~~πλειάδες~~. Στη συνέχεια, με τη βοήθεια της δομής της λίστας υλοποιή- θηκαν ~~υλοποιήθηκαν~~ οι δομές δεδομένων ~~δεδομέ- νων~~ της στοίβας και της ουράς και παρου- σιάστηκαν ~~παρουσιάστηκαν~~ κάποιες εφαρμογές τους.

Τέλος παρουσιάστηκε η δομή δεδομένων του γράφου και του δέν- τρου ~~δέντρου~~.

9

Εφαρμογές σε γλώσσα προγραμματισμού με χρήση API~~API~~

9. Εφαρμογές σε γλώσσα προγραμματισμού με χρήση API

Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστούν ορισμένα θεωρητικά στοι- χεία~~στοιχεία~~ των διεπαφών και στη συνέχεια θα γίνει υλοποίηση μικρών προγραμμάτων~~Προγράμματων~~-εφαρμογών με την βιβλιοθήκη **Tkinter**. Η Tkinter είναι η πρότυπη βιβλιοθήκη της Python για την υλοποίηση γραφι- κών~~υλο- ποίηση γραφικών~~ διεπαφών (Graphical User Interface, GUI). Με τον τρόπο αυ- τό~~αυτό~~ θα έχουμε την ευκαιρία να δούμε πώς μπορούμε να κάνουμε τα προγράμματά μας πιο ελκυστικά και φιλικά προς το χρήστη, χρη- σιμοποιώντας~~χρησιμοποιώντας~~ μια βιβλιοθήκη με έτοιμα τμήματα κώδικα.

Η συνεισφορά της μεγάλης κοινότητας προγραμματιστών που υ- ποστηρίζει~~υποστηρίζει~~ τη γλώσσα~~γλώσ σα~~, με συλλογές έτοιμου κώδικα για πάρα πολλές εφαρμογές, αποτελεί ένα ισχυρό πλεονέκτημα της γλώσ- σας~~γλώσσας~~ προγραμματισμού Python. Οι περισσότερες από αυτές τις εφαρμογές ανήκουν στην κατηγορία του ελεύθερου λογισμικού και είναι γνωστές ως Διεπαφές Προγραμματισμού Εφαρμογών (Application Programming Interfaces- APIs) ή και ως βιβλιοθήκες λογισμικού.

Ωστόσο, ο βασικός σκοπός αυτού του κεφαλαίου δεν είναι να μά- θουμε~~μάθουμε~~ ένα συγκεκριμένο~~συγκε- κριμένο~~ API, ούτε φυσικά να "~~"~~αποστηθίσουμε κά- ποιες"~~κάποιες~~ βασικές συναρτήσεις των βιβλιοθηκών. Σκοπός του είναι να αντιληφθούμε ότι, κατά την ανάπτυξη μιας εφαρμογής~~εφαρ- μογής~~, είναι καλή πρακτική η αξιοποίηση~~αξιοποιήση~~ τυχόν διαθέσιμων τμημάτων έτοιμου κώδι- κα~~κώ- δικα~~ που ανήκουν σε κάποιο API και να πειραματιστούμε με τις δυ- νατότητες~~δυνατότητες~~ μιας τέτοιας βιβλιοθήκης.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- περιγράφουμε τις βασικές αρχές της επικοινωνίας ανθρώπου υπολογιστή
- χρησιμοποιούμε περιβάλλοντα, Application Program Interfaces (APIs) και βιβλιοθήκες για την ανάπτυξη και τροποποίηση εφαρμογών λογισμικού στη γλώσσα προγραμματισμού Python.

~~Κεφ.9: Εφ. σε γλώσσα προγραμματισμού με χρήση API~~

Λέξεις κλειδιά

Επικοινωνία ανθρώπου υπολογιστή, διεπαφή χρήστη, διεπαφές προγραμματισμού εφαρμογών API, βιβλιοθήκες λογισμικού.

Διδακτικές Ενότητες

9.1 Επικοινωνία ανθρώπου-υπολογιστή και διεπαφή χρήστη

Καθημερινά ερχόμαστε σε επαφή με πολλές συσκευές και εφαρ- μογέςεφαρμογές οι οποίες εκτελούν βασικές εργασίες, όπως για παράδειγμα τα μηχανήματα ATM (Automated Teller Machines) στις τράπεζες, τα tablet pc's, μια εφαρμογή αναζήτησης ενός βιβλίουβι- βλίου σε ένα Δια- δικτυακόΔιαδικτυακό βιβλιοπωλείο. Είναι ιδιαίτερα σημαντικό οι συσκευές και οι εφαρμογές τους να είναι κατάλληλα σχεδιασμένες, ώστε οι χρήστες να εκτελούν τις εργασίες που θέλουν εύκολα, αναγνωρί- ζοντεςαναγνωρίζοντας το επόμενο κάθε φορά βήμα, χωρίς λάθη και όσο το δυνα- τόνδυνατό πιο γρήγορα. Τα αντικείμενα, τα εικονίδια και τα μηνύματαμηνύμα- τα που εμφανίζονται στην οθόνη των συσκευών αυτών, οι επιλογές σε μια οθόνη αφής (soft-touch), η είσοδος που δίνουν οι χρήστες με συσ- κευέςσυσκευές όπως το ποντίκι, ο τρόπος ανατροφοδότησης του χρήστη μετά από κάποια ενέργειά του, αποτελούν αντικείμενο έρευνας του ερευνητικού πεδίου που ονομάζεται *αλληλεπίδραση ανθρώπου- αν- θρώπου- υπολογιστή (Human Computer Interaction, HCI)*.

Η **Επικοινωνία Ανθρώπου-Υπολογιστή** (ΕΑΥ) αποτελεί ένα πολλά υποσχόμενο και ταχύτατα αναπτυσσόμενο τομέα που ασ- χολείταιασχολείται με το σχεδιασμό, την υλοποίησηυλοποί- ηση και την αξιολόγηση διαδ- ραστικώνδιαδραστικών υπολογιστικών συστημάτων προορισμένων για ανθρώ- πινηανθρώπινη χρήση και τη μελέτη σημαντικών φαινομένων γύρω από αυτά (πηγή Association for Computer Machinery-ACM).

Ο βασικός στόχος της Επικοινωνίας Ανθρώπου-Υπολογιστή είναι η βελτίωση της ευχρηστίας του συστήματος (usability). Με τον όρο *ευχρηστία ενός συστήματος* και σύμφωνα με το διεθνές πρότυπο ISO 9241, περιγράφουμε την ικανότητα ενός συστήματοςσυ- στήματος να λει- τουργείλειτουργεί αποτελεσματικά και αποδοτικά, παρέχοντας υποκειμενική ικανοποίηση στους χρήστες του.

Η ευχρηστία του συστήματος επιτυγχάνεται όταν η επικοινωνία ενός χρήστη με ένα σύστημα υπολογιστή, γίνεται μέσω ενός περι- βάλλοντοςπεριβάλλοντος που χαρακτηρίζεται από:

- ευκολία εκμάθησης
- υψηλή απόδοση εκτέλεσης έργου (επίδοση)
- χαμηλή συχνότητα εμφάνισης λαθών χρήστη
- ευκολία συγκράτησης της γνώσης της χρήσης τουκαι
- υποκειμενική ικανοποίηση του χρήστη, δηλαδή το κατά πόσο ικανοποιημένοι είναι οι χρήστες από τη χρήση του συστήματος.

Η αλληλεπίδραση μεταξύ χρηστών και υπολογιστών γίνεται στο επίπεδο της διεπαφήςδιεπα- φής χρήστη (user interface), μέσω κατάλληλου λογισμικού και υλικού.

Η μελέτη ανάπτυξης της διεπαφής χρήστη επικεντρώνεται στην κατάλληλη δημιουργίαδημι- ουργία και προσαρμογή εκείνων των στοιχείων του υλικού και του λογισμικού του υπολογιστή, με τα οποία ο άνθρω-ποςάνθρωπος έρχεται σε άμεση επαφή και αποτελεί μέρος του ευρύτερου τομέα της Επικοινωνίας Ανθρώπου-Απολογητή για τη βελτίωση της ευχρηστίας ενός συστήματος (usability).

9.1.1 Παραδείγματα – Εφαρμογές

Η χρηστικότητα μιας διεπαφής σχετίζεται άμεσα με την πληροφο- ρίαπληροφορία που προσφέρει στο χρήστη, συνειδητά ή και υποσυνείδητα, για να εκτελέσει μια ενέργεια ή να έχει χρήσιμη ανατροφοδότηση. Ας παρατηρήσουμε, για παράδειγμα, την πληροφορία που μας δίνουν οι σχεδιαστές για να ανοίξουμε τις πόρτες που συναντάμε πολλές φορές σε καταστήματα. Πότε πρέπει να τραβήξουμε μια πόρτα για να την ανοίξουμε και πότε απλώς πρέπει να τη σπρώ- ξουμεσπρώξουμε; Το μήνυμα «ωθήσατε», που πολλές φορέςφο- ρές βρίσκεται πάνω στην πόρτα, παρότι μας πληροφορεί για την απαραίτητη ενέργειαενέρ- γεια, δεν μας οδηγεί άμεσα στο να κάνουμε αυτόματα την ενέργεια αυ- τήαυτή. Θα ήταν πιο εύκολο οι σχεδιαστές με τον κατάλληλο σχεδιασμό της πόρτας να παρείχαν στο χρήστη την ανάλογη πληροφορία, ώστε αυτόματα να εκτελεί αυτή την ενέργεια. ΑρκείΑρ- κεί στην πλευρά της πόρτας που απαιτείται να τη σπρώξουμε (ώθηση), να μην είχε προσαρμοστεί χερούλι παρά μόνο μια μικρή βιδωμένη μεταλλική πλάκα, ενώ στη πλευρά που απαιτείται να την τραβήξουμε να είχε προστεθεί μία χειρολαβή.

πλάκα, ενώ στη πλευρά που απαιτείται να την τραβήξουμε να είχε προστεθεί μία χειρολαβή.

~~Κεφ.9: Εφ. σε γλώσσα προγραμματισμού με χρήση API~~

Η αντίληψη κάποιας πληροφορίας από τη μεριά του χρήστη επη- ρεάζεται σημαντικά~~επηρεάζεται σημαντι κά~~ από τη *μεταφορά* που χρησιμοποιείται για την οπτική της αναπαράσταση και αποτελεί μια νοηματική σύνδεση της ενέργειας με ένα χαρακτηριστικό εικονίδιο. Ως παράδειγμα γνώριμης μεταφοράς σκεφθείτε τη χρήση του πράσινου και του κόκκινου~~κόκκι νου~~ εικονιδίου, που απεικονίζει ένα ακουστικό στο κινητό τη- λέφωνο~~τηλέφωνο~~ για να ξεκινήσουμε~~ξεκινή σουμε~~ ή να τερματίσουμε μια κλήση αντίστοι- χα~~αντίστοιχα~~. Άλλο παράδειγμα, είναι η χρήση του εικονιδίου της δισκέτας για να απεικονίσουμε τη διαδικασία αποθήκευσης μιας εργασίας μας στον επεξεργαστή κειμένου. Γενικά στο σχεδιασμό μιας δι- επαφής~~διεπαφής~~, οι μεταφορές χρησιμοποιούνται για την απεικόνιση μιας νέας πληροφορίας ή διεργασίας ~~διεργα- σίας~~ με κάποιο οπτικό αντικείμενο, που συνήθως είναι γνώριμο, ώστε εύκολα να καταλάβουμε το νό- ημά~~νόημά~~ της.

Για το σχεδιασμό μιας ΑΤΜ συσκευής και των εφαρμογών της σε μια τράπεζα, αναδεικνύονται ~~ανα- δεικνύονται~~ διάφορα σημαντικά ζητήματα ως προς το σχεδιασμό της διεπαφής με τους πελάτες της τράπεζας, που πρέπει να διερευνηθούν. Για παράδειγμα, ποιός είναι ο ελά- χιστος~~ελάχιστος~~ αριθμός ενεργειών που απαιτούνται για να κάνει κάποιος μια ανάληψη~~ανά- ληψη~~ σε ένα μηχάνημα ΑΤΜ; Τι πρέπει να προβλεφθεί για να μπορεί να χειριστεί ένα ΑΤΜ ένα ηλικιωμένο άτομο που δεν είναι εξοικειωμένο με τους υπολογιστές και πιθανά να έχει και προβλήματα όρασης χωρίς να κάνει λάθη; Τι πρέπει να προβλέ- πεται, ώστε να αποτρέπονται πιθανά λανθασμένες ενέργειες;

9.2 Γενικές αρχές σχεδίασης διεπαφής

Κατά τη διάρκεια του σχεδιασμού της διεπαφής, ο σχεδιαστής της λαμβάνει πάντοτε υπόψη του ότι σχεδιάζει ένα σύστημα για μια ευρύτερη ποικιλία ατόμων-χρηστών και δε περιορίζεται μόνο στο- υς~~στους~~ έμπειρους χρήστες. Κάτω από αυτό το πρίσμα η σχεδιαζόμενη διεπαφή πρέπει να προβλέπει τις ανθρώπινες ιδιαιτερότητες. Για παράδειγμα, η επιλογή των χρωμάτων, των ετικετών, της τοποθέ- τησης~~τοποθέτησης~~ των πλήκτρων~~πλή κτρων~~ και των λειτουργιών δεν θα πρέπει να αντι- βαίνουν~~αντιβαίνουν~~ στις συνήθειες και τα χαρακτηριστικά του χρήστη. Αν η

~~χαρακτηριστικά του χρήστη. Αν η~~ εφαρμογή απευθύνεται σε άτομα ηλικιωμένα ή άτομα με ειδικές ανάγκες, τότε πρέπει να ληφθούν υπόψη τα ιδιαίτερα χαρακτηρισ- τικά~~χαρακτηρι- στικά~~ τους. Για παράδειγμα, εφαρμογές για άτομα με προβλήματα όρασης, πρέπει να συμπεριλάβουν πολύ μεγαλύτερα εικονίδια, ηχητικά μηνύματα ανατροφοδότησης~~ανατροφοδότη σης~~ κ.ά.

Ο σκοπός της σχεδίασης μιας διεπαφής είναι η δημιουργία όσο το δυνατόν πιο εύχρηστων συστημάτων. Στο σχεδιασμό μιας διεπα- φής~~διεπαφής~~ πρέπει να ακολουθούνται ορισμένοι κανόνες, με διάφορους ερευνητές να έχουν παρουσιάσει~~προτείνει~~ δικά τους συστήματα~~συστή ματα~~ κανόνων που προτείνουν να τηρούνται. Ο Shneiderman (Shneiderman and Plaisant, 2005) για παράδειγμα για το σχεδιασμό μιας διεπαφής προτείνει διάφορους~~διάφο- ρους~~ κανόνες, όπως:

1. Ομοιομορφία και συνέπεια (consistency) στη λειτουργία της διεπαφής και αποφυγή απροσδόκητης συμπεριφοράς του συστήματος στα παρακάτω:
   1) Ορολογία.
   2) Μηνύματα.
   3) Μενού.
   4) Οθόνες βοήθειας.
   5) Γραμματοσειρές.
   6) Χρώμα.
   7) Μορφή.
2. Σύντομοι χειρισμοί (shortcut) για τη διευκόλυνση των έμπε- ιρων~~έμπειρων~~ χρηστών:
   1) Συντμήσεις.
   2) Ειδικά πλήκτρα.
   3) Μακροεντολές.
3. Συνεχής ανατροφοδότηση της κατάστασης (informative feedback) του συστήματος~~συστή- ματος~~. Για παράδειγμα το σύστημα πρέπει να απεικονίζει την πορεία εξέλιξης~~εξέλι- ξης~~ μιας τρέχουσας εργασίας. Για την εργασία αντιγραφή αρχείου, μια μπάρα απεικονίζει το ποσοστό από το αρχείο που έχει αντιγρα- φεί,~~αντιγραφεί~~ δίνοντας παράλληλα~~παράλ- ληλα~~ χρήσιμη πληροφορία στο χρήσ- τη~~χρήστη~~ για το χρόνο αναμονής.

4. *Διάλογοι* χρήστη - υπολογιστή, που πρέπει να ολοκληρώ- νονται~~ολοκληρώνονται~~ σε λίγα βήματαβή ~~ματα~~.
5. Πρόβλεψη για σφάλματα των χρηστών και του χειρισμού τουςχειρισμός ~~σφαλμάτων~~.
6. Δυνατότητα αναίρεσης μιας ή περισσότερων ενεργειών με ευκολία.
7. Έλεγχος της αλληλεπίδρασης, από την πλευρά του χρήσ- τηχρήστη και όχι του συστήματοςσυ- ~~στήματος~~.
8. Ελαχιστοποίηση του αριθμού των αντικειμένων που καλεί- ται~~καλείται~~ κάθε φορά να συγκρατήσει ένας χρήστης στη βραχύχ- ρονηβραχύχρονη μνήμη (κανόνας των 7±2 αντικειμένωναντικει ~~μένων~~ ως μέγιστη απαίτηση του συστήματος για το φόρτο της βραχύχρονης μνήμης του χρήστη).

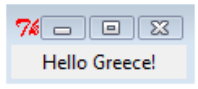
9.3 Η ~~Ή~~ βιβλιοθήκη Tkinter για ανάπτυξη γραφικών διεπαφών GUI στην Python

Στην ενότητα αυτή θα υλοποιήσουμε απλά παραδείγματα ανάπτυ- ξηςανάπτυξης διεπαφών. Μέσα από τη δημιουργία μηνυμάτων ανατροφοδό- τησηςανατροφοδότησης, κουμπιών για την εκτέλεσηεκτέ- ~~λεση~~ κάποιων διεργασιών και πεδίων εισόδων για την εισαγωγή δεδομένων, θα έχουμε την ευκαιρία να διερευνήσουμε πώς να κάνουμε τα προγράμματά μας πιο φιλικά προς το χρήστη, δίνοντας τη δυνατότητα αλληλεπίδρασης με το περιβάλλονπεριβάλ- ~~λον~~ των εφαρμογών μας. Για το σκοπό αυτό θα χρησιμο- ποιήσουμεχρησιμοποιήσουμε τη βιβλιοθήκη Tkinter, που μας προσφέρει πολλές δυ- νατότητεςδυνατότητες για να κατασκευάσουμε γραφικές διεπαφές.

Για να μπορέσουμε να αναπτύξουμε τα προγράμματά μας χρησι- μοποιώνταςχρησιμοποιώντας τη Tkinter, πρέπει να ακολουθήσουμε τη βασική προγραμματιστική λογική στην οποία αυτή η βιβλιοθήκη βασίζεται. Βασικό στοιχείο της είναι τα παράθυρα, μέσα στα οποία μπορούμε να τοποθετήσουμε και άλλα στοιχεία που μας παρέχει η βιβλιοθή- κη

βιβλιοθήκη όπως, μηνύματα, εικόνες, κουμπιά κ.ά. Τα στοιχεία αυτά αποτελούναποτελούν αντικείμενα, που τα ονομάζουμε *widgets*. Για τη δημιουργία και τη διαχείριση των αντικειμένων αυτών, χρησιμοποιούμε κατάλ- ληλακατάλληλα στιγμιότυπα κλάσεων που παρέχειπαρέ- χει η Tkinter.

Παράδειγμα 1



με το πρώτο παράδειγμα για τη δημιουργία ενός μηνύματος πισμού, όπως στην παρακάτω εικόνα.

Αρχικά θα δημιουργήσουμε ένα παράθυρο με τίτλο window_test, το οποίο όμως δεν θα εμφανιστεί αμέσως. Στο αντικείμενο αυτό θα προσθέσουμε στη συνέχεια ένα αντικείμενο *ετικέτας κειμένου* (Label) για να εμφανιστεί ένα μήνυμα χαιρετισμού. Για το σκοπό αυτό θα δημιουργήσουμε ένα στιγμιότυπο της κλάσης Label με την ανάλογη έκφραση.

Μελετήστε τον κώδικα που ακολουθεί, διαβάζοντας τα επεξηγηματικάεπεξηγηματικά σχόλια. Στη συνέχεια, πληκτρολογήστε τον κώδικα στο προγραμματιστικόπρογραμματιστικό περιβάλλον IDLE, για να δείτε το αποτέλεσμα και δοκιμάστε να εμφανίζονται τα δικά σας μηνύματα χαιρετισμούχαι ρετισμού.

```
from Tkinter import * #εισάγει το module Tkinter
window_test = Tk()
 #δημιουργεί ένα αρχικό γραφικό στοιχείο –ένα απλό παρά- θυροπαράθυρο με μπάρα τίτλουτίτ- λου, το οποίο δεν έχουμε εμφανίσει ακόμα. Πρέπει να δημιουργηθεί όμως πριν φτιαχτεί κάποιο άλλο γραφικό στοιχείο και είναι μοναδικό. Το όνομα window_test το χρησιμοποιούμε ως αναφορά στο παράθυρο που στη συνέ- χειασυνέχεια θα διαχειριστούμεδιαχειρι- στούμε για να τοποθετήσουμε τα υπόλοιπα γραφικά στοιχεία.
a = Label(window_test, text="Hello="Hello Greece!")!")
#δημιουργεί ένα γραφικό στοιχείο με όνομα a ως Label. Η πρώτη παράμετρος της Label είναι το όνομα του παραθύρου γονέα δηλ του window_test. Η δεύτερη
```

παράμετρος ορίζει το κείμενο που θα

εμφανίζεται. a.pack()

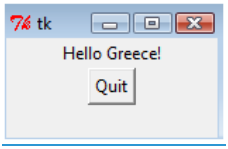
Η μέθοδος pack τοποθετεί το a γραφικό στοιχείο κειμένου στο παράθυρο και ελέγχουμε την εμφάνιση του παραθύρου.

window_test.mainloop()

#το script θα παραμείνει ενεργό μέχρι να κλείσει το παράθυ-

ροπαράθυρο. Η τελευταία εντολή θα εμφανίσει το παράθυρο

Στη συνέχεια θα τροποποιήσουμε κατάλληλα τον παραπάνω κώ- δικακώδικα, ώστε να αλλάξουμεαλ λάξουμε το μέγεθος του παραθύρου με τη μέθοδο wm_geometry() και να προσθέσουμε προσθέ- σουμε ένα κουμπί με όνομα buttonA, χρησιμοποιώντας το widget Button. Το κουμπί αυτό θα το χρησιμοποιήσουμε, ώστε να τερματίζει το πρόγραμμα, όταν το πατήσει ο χρήστης.



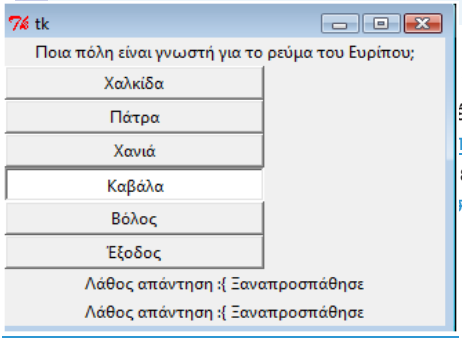

```
from Tkinter import * window_test = Tk()
window_test.wm_geometry("150x70("150x70
+10+20")20")
#Η μέθοδος wm_geometry μας επιτρέπει να αλλάξουμε το μέ- γεθοςμέγεθος του παραθύρουπαραθύ- ρου και να δηλώσουμε τις συντεταγμένες της πάνω αριστερά γωνίας
a = Label(window_test, text="Hello="Hello Greece!")!")
a.pack()
buttonA= Button(window_test, text= "Quit", com- mand "Quit", command=window_test.destroy)
#δημιουργεί ένα γραφικό στοιχείο κουμπί που αναφέρεται με το όνομα buttonA. Η πρώτη παράμετρος της Button είναι το όνομα του παραθύρου γονέα δηλ του window_test. Η δεύτερη παρά- μετροςπαράμετρος ορίζει το κείμενο που θα εμφανίζεται πάνω
```

στο κουμπί και η τρίτη παράμετρος ορίζει την ενέργεια που θα εκτελεστεί όταν πατηθεί το κουμπί.

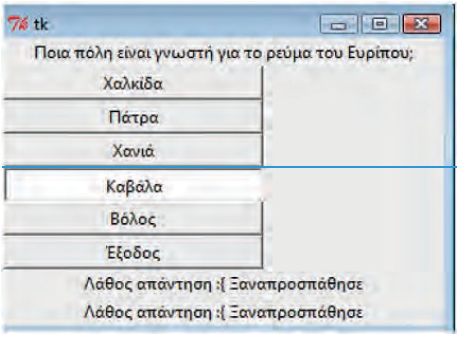
```
but
ton
A.
pa
ck(
)
```

Η μέθοδος pack τοποθετεί το γραφικό στοιχείο buttonA στο παράθυρο και ελέγχουμε την εμφάνιση του παραθύρου.

window_test.mainloop()



που χρησιμοποιείται για να εισάγουμε ένα μικρό quiz γνώσεων. Αν η επιλογή ται~~εμφανίζεται~~ ένα μήνυμα λάθους, ενώ εμφανίζεται ένα μήνυμα επιβράβευσης. ~~τουμε~~ ένα κουμπί για έξοδο από το quiz,



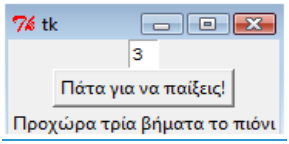

```
# -*- coding: utf-8 -*-
from Tkinter import *
root = Tk()
c = IntVar()
c.set(0) # αρχικοποίηση επιλογών
towns = [ ("Χαλκίδα",
   ("Χαλκίδα",1),
   ("Πάτρα",("Πά
   τρα",2),
   ("Χανιά",
```

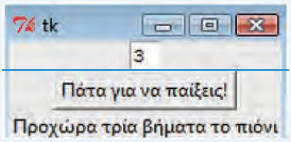
~~("~~
~~X~~
~~α~~
~~vı~~
~~ά'~~
~~',~~
3)
,
("
K
α
β
ά
λ
α
",
~~("~~
~~K~~
~~α~~
~~β~~
~~ά~~
~~λ~~
~~α'~~
~~',~~
4)
,-
~~("~~
~~B~~
~~ό~~
~~λ~~
~~ο~~
~~ς"~~
~~,5~~
~~);~~

```
        ("Βόλος",5),
        ("Έξοδος",6)
]
def ShowChoice():
# #συνάρτηση που ορίζει τις ενέργειες που θα εκτελεστούν
ανάλογα με την επιλογή που θα κάνει ο παίκτης του παιχνιδι-
ούπαιχνιδιού
    if c.get()==6:
        root.destroy()
  elif c.get()==1:
      Label (root, text= 'Μπράβο!! Σωστά απάντησες.
Κέρδι- σεςΚέρδισες 10 πόντους').pack()
  else:
      Label (root, text= 'Λάθος απάντηση,
Ξαναπροσπάθησε').pack()

Label(root, text="Ποια="Ποια πόλη είναι γνωστή για το ρεύμα του
Ευρίπου;",;'', justify =
      LEFT, padx =
      20).pack()
for txt, val in towns:
   Radiobutton(root,
               text=txt,
               indicatoron = 0,
               width = 20,
               padx = 20,
               variable=c,
               command= ShowChoice,
               value=val).pack(anchor=W)

 mainloop()
```

Παράδειγμα 3

Σας έχουν αναθέσει τη δημιουργία ενός παιχνιδιού γνώσεων, ό- πουόπου σε κάθε πάτημα ενός κουμπιού θα ορίζεται τυχαία ο αριθμός των βημάτων (1-6) που θα κάνει το πιόνι του παίκτη.






```
# -*- coding: utf-8 -*-
from Tkinter import *
from random import randint
window= Tk()
text= Text (window, width = 2, height =1)
 #ορίζει τη λευκή περιοχή που θα εμφανίζεται ο αριθμός των βημάτων που θα κάνει το πιόνι
text.pack()
def roll():
#συνάρτηση που ορίζει τις ενέργειες που τις οδηγίες που θα εμφανιστούν ανάλογα με τον αριθμό των βημάτων.
    text.delete(0.0, END)
    a=str (randint(1,6))
    text.insert(END,a)
      a=str (randint(1,6))
      text.insert(END,a) if a=="1":
    if a=="1":
        w=Label(window, text='Προχώρα ένα βήμα το πιόνι')
    elif a=="2":
        w=Label(window, text='Προχώρα δύο βήματα το πιόνι')
    elif a=="3":
        w=Label(window, text='Προχώρα τρία βήματα το πιόνι')
    elif a=="4":
        w=Label(window, text='Προχώρα 4τέσσερα βήματα το πιόνι')
    elif a=="5":
        w=Label(window, text='Προχώρα πέντε βήματα το πιόνι')
```

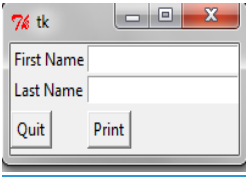
```
    else:
        w=Label(window, text= 'Προχώρα έξι βήματα το πιόνι')
        w.pack()
        return a
buttonA = Button (window, text = 'Πάτα για να παίξεις!',
com-command = roll)
```

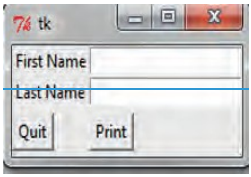
```
    mand = roll)
# Κεφ.9: Εφ. σε γλώσσα προγραμματισμού με χρήση API

 #δημιουργεί ένα γραφικό στοιχείο κουμπί που
αναφέρεται με το όνομα buttonA. Η πρώτη
παράμετρος της Button είναι το
όνομα του παραθύρου γονέα δηλ του window. Η
δεύτερη πα- ράμετροςπαράμετρος ορίζει το
κείμενο που θα εμφανίζεται πάνω στο κουμ-
πίκουμπί και η τρίτη παράμετρος ορίζει την
ενέργεια που θα εκτελεστεί όταν πατηθεί το κουμπί
δηλ. καλεί τη συνάρτηση roll () που δη-
μιουργήσαμεδημιουργήσαμε παραπάνω.
but
ton
A.
pa
ck(
)
# Η μέθοδος pack τοποθετεί το γραφικό στοιχείο
buttonA στο παράθυρο και ελέγχουμε την
εμφάνιση του παραθύρου. window.mainloop()
windo
w.main
loop()
```

Παράδειγμα 4.:

Εισαγωγή δεδομένων

Πολύ συχνά χρειάζεται σε μια εφαρμογή να διαβάσουμε κάποια δεδομένα από το πληκτρολόγιο, εμφανίζοντας στο χρήστη μια φόρμα εισαγωγής δεδομένων. Στη συνέχειασυ- νέχεια τα δεδομένα αυτά μπορούμε να τα χρησιμοποιήσουμε σε κάποιους υπολογισμούςυπολογι- σμούς ή απλά να τα εμφανίσουμε στην οθόνη του υπολογιστή.





Για την εισαγωγή δεδομένων χρησιμοποιούμε τα widgets Entry, τα οποία δημιουργούνδημιουρ γούν ένα πλαίσιο μέσα στο οποίο ο χρήστης μπορεί να εισάγει μια γραμμή κειμένου. Για να μπορέσουμε να έχουμε πρόσβαση και να διαβάσουμε τα περιεχόμενο που εισάγει ο χρήσ-χρήστης χρησιμοποιούμε τη μέθοδο get().

της χρησιμοποιούμε τη μέθοδο get().

Στο παράδειγμα που ακολουθεί χρησιμοποιούμε δύο πεδία Entry για να εισάγουμεεισά γουμε το όνομα (fname) και το επίθετο (lname) ενός μαθητή. Στη συνέχεια χρησιμοποιούμεχρη σιμοποιούμε δύο κουμπιά, το Quit για να τερματίσει το πρόγραμμα και το κουμπί Print για να

εμφανιστούν τα περιεχόμενα. Το κουμπί Print όταν πατηθεί, κα- λείκαλεί τη συνάρτηση show_eisodo() που έχουμε δημιουργήσει. Η συνάρτηση χρησιμοποιείχρησιμο- ποιεί τη μέθοδο get() για να διαβάσει το ό- νομαόνομα fname.get() και το επίθετο lname. get(0) ενός μαθητή και με τη χρήση της print εμφανίζονται στη συνέχεια στην

οθόνη του υπολογιστή. Για να διαγράψουμε τα στοιχεία από τη φόρμα, ώστε να εισάγουμε ξανά νέα στοιχεία χρησιμοποιούμε τη μέθοδο delete() για διαγραφή. Η μέθοδος grid() χρησιμοποιείται για να οργανώσει κατά γραμμές και στήλες την εμφάνιση των γραφικών αντικειμένων (Label, Entry, Button) που εμφανίζονται στο παράθυρο.

```
from Tkinter import *

def show_eisodo():
    print "First Name: %s\nLast Name: %s" % (fname.get(), lname.get())
    fname.delete(0,END)
    lname.delete(0,END)

window = Tk()
Label(window, text="First Name").grid(row=0)
Label(window, text="Last Name").grid(row=1)

fname = Entry(window)
lname = Entry(window)
fname.grid(row=0, column=1)
lname.grid(row=1, column=1)

Button(window, text='Quit', command=window.quit).grid(row=3, column=0, sticky=W, pady=4)
Button(window, text='Print', command=show_eisodo).grid(row=3, column=1, sticky=W, pady=4)
mainloop( )
```

Δραστηριότητες - Ερωτήσεις - Ασκήσεις κεφαλαίου

9.4 Δραστηριότητες

Δραστηριότητα 1

Τροποποιήστε κατάλληλα τον κώδικα του 1ου παραδείγματος για τη δημιουργία μηνυμάτωνμη- νυμάτων, ώστε να εμφανίσετε τα δικά σας μηνύ- ματαμηνύματα με χρήση του Label.

Δραστηριότητα 2

Τροποποιήστε κατάλληλα το 2ο παράδειγμα, που παρουσιάστηκε παραπάνω, ώστε να δημιουργήσετε το δικό σας quiz γνώσεων.

Δραστηριότητα 3

Δοκιμάστε στο 4ο παράδειγμα, που παρουσιάστηκε παραπάνω, να εμπλουτίσετε κατάλληλα τη φόρμα εισαγωγής στοιχείων, ώστε να εισάγονται και στη συνέχεια να εμφανίζονται περισσότερες πληροφορίες για το μαθητή, όπως το τμήμα και η τάξη του, η ηλι- κίαηλικία, το φύλο, ο M_O των βαθμών τριμήνου κ.ά.

Αν θέλετε να βρείτε περισσότερες πληροφορίες για τις δυνατότη- τεςδυνατότητες που μας παρέχειπαρέ- χει η βιβλιοθήκη Tkinter επισκεφτείτε τις προτει- νόμενεςπροτεινόμενες πηγές καθώς και τη βοήθεια του Tkinter που είναι διαθέ- σιμηδιαθέσιμη στην ιστοσελίδα της Python.

9.4.1 Εργαστηριακές ασκήσεις

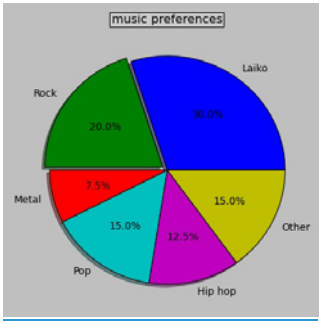
Άσκηση 1

Επισκεφτείτε το δικτυακό τόπο υποστήριξης της βιβλιοθήκης matplotlib ,http:// matplotlib.org/examples, που χρησιμοποιείται για την οπτικοποίηση δεδομένων. Δείτε μερικά έτοιμα παραδείγματα και συζητήστε στη τάξη για τις δυνατότητες που μας δίνει η βιβλιο- θήκηβιβλιοθήκη, προγραμματίζοντας σε γλώσσα Python και για την αξία που έχει η οπτικοποίηση των δεδομένων. Δώστε χαρακτηριστικά πα- ραδείγματα.παραδείγματα. Στη συνέχειασυ- νέχεια μελετήστε τα έτοιμα παραδείγματα οπτι- κοποίησηςοπτικοποίησης στατιστικών δεδομένων για τη δημιουργία ενός γραφή- ματοςγραφήματος πίτας (pie).

Επέκταση της άσκησης

Ακολουθήστε τις οδηγίες για την εγκατάσταση της βιβλιοθήκης αν χρειαστεί, ανάλογαανά ~~λογα~~ με το προγραμματιστικό περιβάλλον και το ΛΣ του υπολογιστή σας. Για παράδειγμα~~παρά δειγμα~~ σε ΛΣ Debian\Ubuntu αρκεί η: **sudo apt-get build-dep python-matplotlib** για να εγκαταστα- θείεγκατασταθεί η βιβλιοθήκη μαζί με όλες τις άλλες βιβλιοθήκες που συσχετίζε- ταισυσχετί ~~ζεται~~.

Στη συνέχεια τροποποιήστε το παράδειγμα οπτικοποίησης δεδο- μένωνδεδομένων ~~δεδομένων~~ με πίτα, ώστε να απεικονιστούν, όπως στην εικόνα, τα δε- δομέναδεδομένα ~~δεδομένα~~ του παρακάτω προβλήματος.



Στην εικόνα φαίνεται η

~~music preferences~~

~~Rock~~

~~laiko~~

~~Metal~~ ~~20,0%~~

~~7,5%~~ ~~30,0%~~

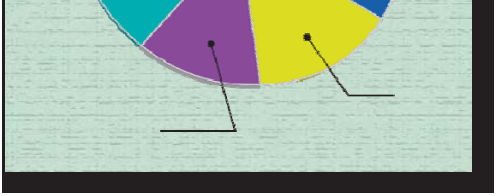
~~15,0%~~

~~Pop~~

~~12,5%~~ ~~15,0%~~

~~Other~~

~~Hip hop~~



Εικόνα: Η οπτικοποίηση των δεδομένων του παρακά- τω ~~παρακάτω~~ προβλήματος σε πίτα, με αξιοποίηση της βιβλιοθήκης matplotlib και χρήση της pie() και με αυτόματη απεικόνιση των ποσοστών.

Πρόβλημα~~**Το πρόβλημα**~~ (από το βιβλίο Μαθηματικών Β' Γυμνασίου).

Μια δισκογραφική εταιρεία προσπαθεί να επεκτείνει τις πωλήσεις της σε εφήβους. Προτού να επενδύσει σε είδη μουσικής που προτιμούν~~προτιμούν~~ οι μαθητές, αποφασίζει να κάνει μια έρευνα ανάμεσα σε ~~200 μαθητές που επέλεξε τυχαία απ' όλη την Ελλάδα. Ο υπεύθυνος, που έκανε την έρευνα, παρουσίασε στο διευθυντή της εταιρείας τα παρακάτω δεδομένα:~~

200 μαθητές που επέλεξε τυχαία απ' όλη την Ελλάδα. Ο υπεύθυνος, που έκανε την έρευνα, παρουσίασε στο διευθυντή της εταιρείας τα παρακάτω δεδομένα:

60 μαθητές προτι-	Λαϊκό
40 μαθητές προτιμούν το	Rock
15 μαθητές προτιμούν το	Metal

30 μαθητές προτιμούν το	Pop
25 μαθητές προτιμούν το	Hip hop
30 μαθητές δήλωσαν	άλλο

9.5 Σύντομα Θέματα για συζήτηση στην τάξη

Διαβάστε τα παρακάτω θέματα και αναπτύξτε στη τάξη τις προτάσειςπροτάσεις σας. ΤεκμηριώστεΤεκμηρι- ώστε ανάλογα την άποψή σας.

Θέμα 1ο

Σε ένα «έξυπνο» τηλέφωνο πόσες κινήσεις χρειάζονται για να κάνουμεκάνουμε μία κλήση από το ευρετήριο; Σε σχέση με ένα «παραδοσιακό» κινητό τηλέφωνο είναι πιο σύνθετησύν- θετη ή πιο απλή διαδικασία; Τι θα προτείνατε για να βελτιώσετε τη διαδικασία αυτή;

Θέμα 2ο

Ποια εικονίδια πιστεύετε ότι είναι τα πιο κατάλληλα να χρησιμοποιήσουμεχρησιμοποιήσουμε σε μια εφαρμογή για να δηλώσουμε στο χρήστη ότι με αυτά μπορεί αντίστοιχα: να αλλάξει τη γλώσσα γραφής από το πληκτρολόγιο, να εκτυπώσει ένα κείμενο, να ζητήσει βοήθεια για τη χρήση του, να αναζητήσει ένα στοιχείο, να διαγράψει ένα στοιχείοστοιχείο;

Θ
έ
μ
α

3
3
ο

Στις υπηρεσίες μιας τράπεζας εξυπηρέτησηςγια την εξυπηρέτηση των πελατών της, προστίθεται ως νέα υπηρεσία η δυνατότητα πληρωμής λογαριασμών τηλεφώνων μέσω των ΑΤΜ με αυτόματα χρέωση του λογαριασμού του ενδια- φερόμενουενδιαφερόμενου πελάτη. ΛαμβάνονταςΑαμβά- νοντας υπόψη σας τους βασικούς κα- νόνεςκανόνες του Shneiderman, που παρουσιάστηκαν στην ενότητα για το σχεδιασμό μιας διεπαφής, να αναπτυχθεί στη τάξη συζήτηση με βάση τα παρακάτω ερωτήματα:

- Ποια είναι τα βασικά στοιχεία προσοχής των σχεδιαστών~~που πρέπει να προσέξουν οι σχεδιαστές~~ της εφαρμογής για τη νέα αυτή υπηρεσία της Τράπεζας, ώστε να μη γίνουν λάθη κατά τη διαδικασία πληρωμής;
- Πώς μπορούν να αποτραπούν λάθη κατά την είσοδο της ταυτότητας του λογαριασμού~~λο- γαριασμού~~ τηλεφώνου, που θα~~πρόκειται να~~ πληρωθεί;
- Πώς ο πελάτης θα γνωρίζει ότι τελικά η διαδικασία πληρωμής ολοκληρώθηκε με επιτυχία;

9.6 Ερωτήσεις - Ασκήσεις

~~9.6.1~~ Ερωτήσεις

1. Τι είναι η επικοινωνία ανθρώπου υπολογιστή;
2. Τι είναι η διεπαφή χρήστη;
3. Τι είναι η ευχρηστία συστήματος και ποια βασικά χαρακτηριστικά~~χαρακτηριστικά~~ έχει ένα σύστημα με αυξημένη ευχρηστία;
4. Ποιοι είναι οι βασικοί κανόνες για το σχεδιασμό μιας διε- παφής~~διεπαφής~~;
5. Τι είναι οι διεπαφές προγραμματισμού εφαρμογών ~~–~~ ~~–~~Application Programming Interfaces (APIs);
6. Σε τι μας χρησιμεύει η βιβλιοθήκη Tkinter της ~~γλώσσας~~ Python;
7. Πώς μπορούμε να εμφανίσουμε ένα απλό μήνυμα ανατ- ροφοδότησης~~ανατροφοδότησης~~ προς το χρήστη με χρήση της Tkinter στην γλώσσα Python;
8. Σε τι μας χρησιμεύουν τα RadioButtons;

Σύνοψη κεφαλαίου

Στην ενότητα αυτή αρχικά προσεγγίσαμε τις βασικές αρχές σχεδί- ασης~~σχεδίασης~~ διεπαφής και υλοποιήσαμε~~. Στη συνέχεια είχαμε την ευκαιρία να υλοποιήσουμε~~ απλά παραδείγματα ανάπτυξης διεπαφών, χρησιμοποιώντας τη βιβλιοθήκη Tkinter της γλώσσας Python. Βιβλιοθήκη~~Βιβλιοθή- κη~~ που μας προσφέρει πολλές δυνατότητες για να κατασκευάσουμε γραφικές διεπαφές~~δι- παφές~~. Μέσα από τη δημιουργία μηνυμάτων ανατροφοδότησης, κουμπιών για την εκτέλεση κά- ποιων~~κάποιων~~ διεργασιών και πεδίων εισόδων για την εισαγωγή δεδομέ- νων~~δεδομένων~~, διερευνήσαμε πώς να κάνουμε τα προγράμματά μας πιο φι- λικά~~φιλικά~~ προς το χρήστη δίνοντας τη δυνατότητα αλληλεπίδρασης με το περιβάλλον των εφαρμογών μας.

10

Βάσεις Δεδομένων

10. Βάσεις δεδομένων

Εισαγωγή

Στο κεφάλαιο αυτό προσεγγίζονται θέματα που αφορούν στη δημιουργίαδημιουργία και διαχείρισηδιαχείριση Βάσεων Δεδομένων μέσα από εντολές της γλώσσας Python.

Ένας συνηθισμένος τρόπος αποθήκευσης δεδομένων στον προγραμματισμόπρογραμματισμό είναι αυτός που χρησιμοποιεί αρχεία. Τον τρόπο αυτό τον γνωρίσαμε τόσο στη Β'Β' τάξη όσο και σε προηγούμενα κεφάλα- ιακεφάλαια αυτού του βιβλίου. Ο τρόπος αυτός είναι ικανοποιητικόςικα νοποιητικός μόνο, όταν τα δεδομένα είναι σχετικά περιορισμένα σε μέγεθος, απλά και χρησιμοποιούνται από λίγους χρήστες. Όταν τα δεδομένα είναι πολύπλοκα και υπάρχουν πολλοί χρήστες, τότε η αποθήκευσή τους σε ένα μόνο αρχείο ή σε πολλά αρχεία παρουσιάζει πολλά προβλήματα, τα οποία λύνουν οι Βάσεις Δεδομένων.

Μια **Βάση Δεδομένων**-ΒΔ (Data Base) αποτελείείναι μια συλλογή δε- δομένωνδεδομένων με υψηλό βαθμό οργάνωσης. Αναπόσπαστο τμήμα μιας βάσης δεδομένων αποτελεί ένα πακέτοπα κέτο προγραμμάτων λογισμι- κούλογισμικού που ονομάζεται *Σύστημα Διαχείρισης Βάσης Δεδομένων**Δεδο μένων* – ΣΔΒΔ (Data Base Management System), το οποίο επιτρέπει τη δημιουργία της ΒΔ και τη διαχείριση των δεδομένων που περιέχει, κρύβοντας την πολυπλοκότητάπολυπλοκό τητά της από το χρήστη.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- περιγράφουμε τους βασικούς τύπους δεδομένων που χρησιμοποιεί η βιβλιοθήκηβιβλιο- θήκη της Python, sqlite3, για τη δημιουργία και διαχείριση ΒΔ
- περιγράφουμε το Μοντέλο Δεδομένων και τα επίπεδά του
- χρησιμοποιούμε τις βασικές εντολές για τη δημιουργία μιας ΒΔ
- χρησιμοποιούμε τις βασικές εντολές για τη δημιουργία ενός πίνακα σε μια ΒΔ
- χρησιμοποιούμε τις βασικές εντολές για την εισαγωγή, τροποποίηση και διαγραφήδια- γραφή εγγραφών σε ένα πίνακα

- χρησιμοποιούμε παραλλαγές της εντολής SELECT για την αναζήτηση στοιχείωνστοι- χείων σε μια ΒΔ.
- διαγράφουμε έναν πίνακα.

~~*Κεφ.10: Βάσεις Δεδομένων*~~

Λέξεις κλειδιά

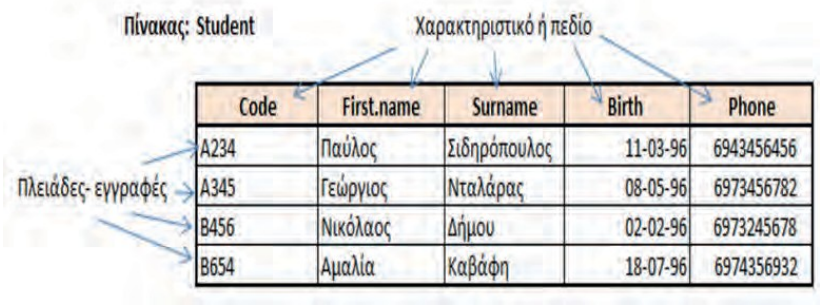
Βάση δεδομένων, μοντέλο δεδομένων, σχεσιακό μοντέλο δεδομέ- νων~~δεδομένων~~, sqlite3, εισαγωγή~~εισα- γωγή~~, τροποποίηση και διαγραφή δεδομένων, πίνακας.

Διδακτικές Ενότητες

10.1 Αναφορά στο Μοντέλο Δεδομένων

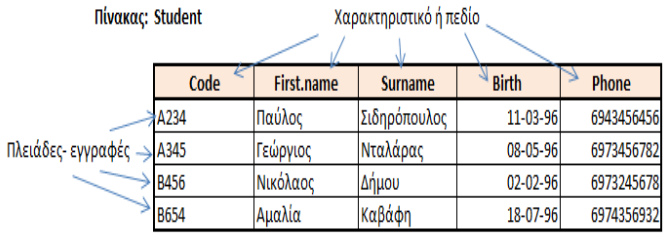
Κάθε ΒΔ ακολουθεί ένα σύστημα οργάνωσης και συσχέτισης των δεδομένων της, που ονομάζεται *μοντέλο δεδομένων*. Συνεπώς ένα μοντέλο δεδομένων καθορίζει τον τρόπο που οργανώνονται τα δεδομένα, καθώς επίσης και τον τρόπο με τον οποίο αυτά σχετί- ζονται~~σχετίζονται~~ μεταξύ τους. Ένα βασικό συστατικό ενός μοντέλου δεδομέ- νων είναι ότι παρέχει ακρίβεια αναπαράστασης, δηλαδή τα στοιχε- ία~~στοιχεία~~ και οι συσχετίσεις~~συσχετί- σεις~~ τους μπορούν να ερμηνευτούν με ένα και μο- ναδικό~~μοναδικό~~ τρόπο. Επίσης το μοντέλο δεδομένων παρέχει το μέσο με το οποίο η ομάδα ανάπτυξης μιας ΒΔ, που ίσως αποτελείται από άτομα με διαφορετικές γνώσεις και ικανότητες, όπως για παράδε- ιγμα~~παρά- δειγμα~~ από λογιστές, πρόσωπα της Διεύθυνσης, προγραμματιστές, να έχουν τη δυνατότητα να επικοινωνούν μεταξύ τους.

Στο παρόν κεφάλαιο θα ασχοληθούμε με τις **Σχεσιακές Βάσεις Δεδομένων** (Relational Data Base), οι οποίες βασίζονται στο αν- τίστοιχο~~αντίστοιχο~~ σχεσιακό μοντέλο δεδομένων. Σύμφωνα με αυτό, τα δε- δομένα~~δεδομένα~~ οργανώνονται σε *πίνακες* (tables ή relations) οι οποίοι σχετίζονται μεταξύ τους με *συσχετίσεις* (relationships). Ένας πίνα- κας~~πίνακας~~ αποτελείται από γραμμές και στήλες, όπου τοποθετούμε τα στοιχεία σε οριζόντια και κάθετη μορφή. Κάθε γραμμή περιέχει τις πληροφορίες για ένα στοιχείο~~στοι- χείο~~ του πίνακα και αποκαλείται ~~*πλειάδα* (tuple) ή~~ *εγγραφή* (record). Κάθε στήλη χαρακτηρίζει κάποια ιδιότητα του πίνακα και αποκαλείται *χαρακτηριστικό* (attribute) ή *πεδίο* (field) κάθε εγγρα- φής~~εγγραφής~~.



Code	First.name	Surname	Birth	Phone
A234	Παύλος	Σιδηρόπουλος	11-03-96	6943456456
A345	Γεώργιος	Νταλάρας	08-05-96	6973456782
B456	Νικόλαος	Δήμου	02-02-96	6973245678
B654	Αμαλία	Καβάφη	18-07-96	6974356932

Παράδειγμα



Code	First.name	Surname	Birth	Phone
A234	Παύλος	Σιδηρόπουλος	11-03-96	6943456456
A345	Γεώργιος	Νταλάρας	08-05-96	6973456782
B456	Νικόλαος	Δήμου	02-02-96	6973245678
B654	Αμαλία	Καβάφη	18-07-96	6974356932

Κάθε μοντέλο δεδομένων έχει μια **αρχιτεκτονική τριών** επιπέ- δωνεπιπέδων. Τα επίπεδα αυτά είναι:

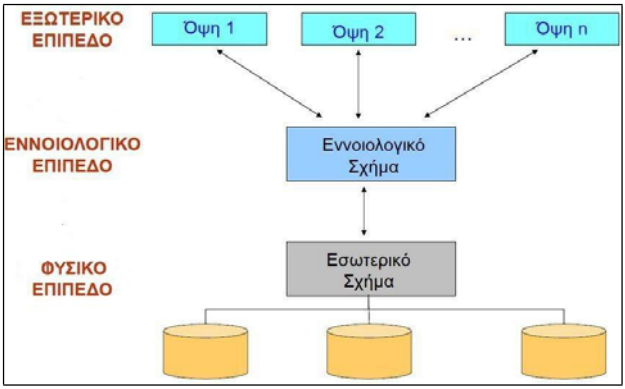
- Το Εξωτερικό επίπεδο (external level), η όψη της ΒΔ από τους χρήστες.
- Το Εννοιολογικό επίπεδο (conceptual level), η λογική όψη της ΒΔ.
- Το Φυσικό επίπεδο (internal level), η φυσική οργάνωση και αποθήκευση των δεδομένων.~~)~~

Το **εξωτερικό** επίπεδο περιλαμβάνει όλες τις όψεις (views) της βάσης δεδομένων που έχουν οι διάφοροι χρήστες. Το εξωτερικό επίπεδο είναι ουσιαστικά ο τρόπος με τον οποίο βλέπει ο κάθε χρήστης τα περιεχόμενα της βάσης. Έτσι, για παράδειγμα, άλλες πληροφορίες βλέπει το λογιστήριο και άλλες η διαχείριση υλικών.

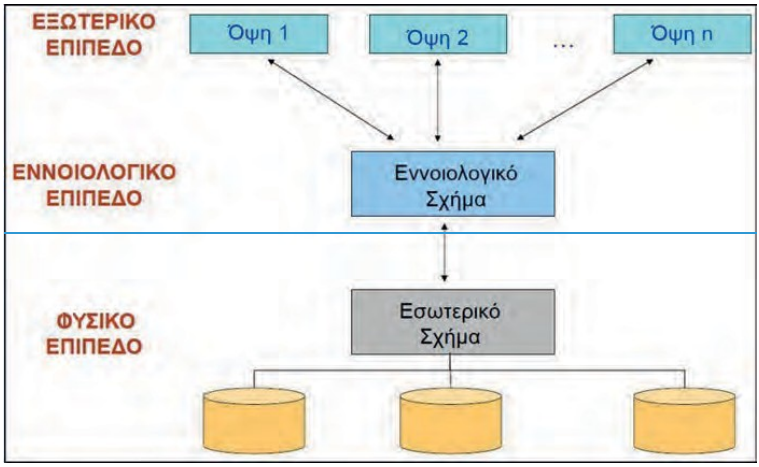
Το **εννοιολογικό** επίπεδο περιλαμβάνει τη συνολική *λογική* οργά- νωσηοργάνωση των δεδομένωνδε- δομένων της βάσης δεδομένων. Σε αυτό το επίπεδο περιγράφονται αναλυτικά τα δεδομένα καθώς και ο τρόπος με τον οποίο αυτά συσχετίζονται μεταξύ τους. Στο εννοιολογικό επίπεδο, του σχεσιακού μοντέλου το οποίο θα χρησιμοποιήσουμε, περιγ- ράφονταιπε- ριγράφονται όλοι οι πίνακες της βάσης δεδομένων καθώς και ο τρό-ποςτρόπος με τον οποίο συσχετίζονται μεταξύ

τους.

Το **φυσικό** επίπεδο περιλαμβάνει τη φυσική αποθήκευση των δεδομένωνδεδομένων σε αρχεία πάνω σε φυσικά μέσα αποθήκευσης, όπως για παράδειγμα σε σκληρούς δίσκους.



~~*Κεφ.10: Βάσεις Δεδομένων*~~



Εικόνα 10-1. Αρχιτεκτονική τριών επιπέδων στις ΒΔ

10.2 Εισαγωγή στη διαχείριση Βάσεων Δεδομένων με προγραμματισμό

Έχει αναφερθεί ότι τα δεδομένα μιας Βάσης Δεδομένων αποθηκεύονται~~αποθηκεύονται~~ με κάποια μορφή αρχείων, σε φυσικά μέσα αποθήκευσης. Αυτό επιτρέπει σε γλώσσες *τρίτης γενιάς* να μπορούν να διαχειριστούν~~διαχειριστούν~~ τα στοιχεία μιας σχεσιακής ΒΔ. Στην πράξη βέβαια, η διαχείριση~~διαχείριση~~ της ΒΔ γίνεται μέσα από το ΣΔΒΔ.

Τη δεκαετία του 1980 έγινε προσπάθεια να δημιουργηθούν ειδικές γλώσσες προγραμματισμού~~προ γραμματισμού~~ οι οποίες να συνεργάζονται με βάσεις

δεδομένων και να είναι εύκολο να τις χρησιμοποιήσουν και απλοί χρήστες, χωρίς ειδικές γνώσεις προγραμματισμού.~~προγραμματισμού.~~ Οι γλώσσες αυτές ονομάστηκαν γλώσσες τέταρτης γενιάς (4GL). Στο χειρισμό σχεσιακών βάσεων δεδομένων επικράτησε η SQL (Structure Query Language-Δομημένη~~Δομημένη~~ γλώσσα ερωταποκρίσεων).

10.2.1 Η~~Η~~ γλώσσα SQL

Η γλώσσα **SQL** ξεκίνησε από τα εργαστήρια της IBM και τυποποιήθηκε~~τυποποιήθηκε~~ το 1987 (ANSI SQL). Οι διάφοροι κατασκευαστές ακολουθούν~~ακολουθούν~~ το πρότυπο αυτό, αλλά παράλληλα~~παράλληλα~~ επεκτείνουν τις δυνατό- τητες~~δυνατότητες~~ της γλώσσας με την εισαγωγή νέων εντολών~~εντολών~~ και συναρτή- σεων.~~συναρτήσεων.~~ Η γλώσσα αυτή είναι προσανατολισμένη στη διαχείριση

σχεσιακών βάσεων δεδομένων και όχι στον κλασικό προγραμμα-

τισμό.προγραμματισμό. Έτσι, δε διαθέτειδι- αθέτει δομές επιλογής τύπου ΑΝ … ΤΟΤΕ (if … then …) ή δομές επανάληψης. ΠαρέχειΠα- ρέχει όμως τη δυνατότητα πρόσβασης στα στοιχεία της Βάσης Δεδομένων, μπορεί να εκτε- λέσειεκτελέσει αριθμητικές και λογικές πράξεις, να χειριστεί συμβολοσειρές (strings), καθώς και να παράγει αναφορές (εκθέσεις) σε οποιαδή- ποτεοποιαδήποτε μορφή.

Η γλώσσα SQL περιέχει δύο μέρη:

- Τη **Γλώσσα Ορισμού Δεδομένων** (Data Definition Language -DDL), που χρησιμοποιείται για τη δημιουργία της δομής της ΒΔ.
- Τη **Γλώσσα Χειρισμού Δεδομένων** (Data Manipulation Language - DML), που χρησιμοποιείται για την εισαγωγή, διαγραφή, ενημέρωση, αναζήτηση και ταξινόμηση δεδομένων της βάσης δεδομένων.

Η SQL είναι εύκολη στην εκμάθηση και μπορεί να χρησιμοποιηθεί και από απλούς χρήστες μετά από εξάσκηση.

Για παράδειγμα η εντολή:

```
SELECT Name, Phone
FROM customers
WHERE City = "Λάρισα"
```

θα εμφανίσει το όνομα και το τηλέφωνο των πελατών που περιέ- χονταιπεριέχονται στον πίνακα customers και είναι από τη Λάρισα.

10.2.2 Η Η βιβλιοθήκη SQLite της Python

Η SQLite είναι ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων συμβατό με το ISO SQL με χαρακτηριστικά όπως:

- Δε χρειάζεται καμία ρύθμιση.
- Αποθηκεύει μια Βάση Δεδομένων σε ένα μόνο αρχείο.
- Είναι εξαιρετικά γρήγορη, για λογικό όγκο εγγραφών.

Η SQLite είναι γραμμένη σε C και είναι το ευρύτερα διαδεδομένο ελεύθερο σύστημα ΣΔΒΔ. Ο κώδικας της SQLite είναι ελεύθερος για χρήση για οποιονδήποτε σκοπό (public domain), εμπορικό ή ιδιωτικό και υποστηρίζεται από μια μεγάλη κοινότητα.

Η SQLite υποστηρίζει πέντε τύπους δεδομένων:

- NULL - Αντιπροσωπεύει το "τίποτα" (όχι το μηδέν).
- INTEGER - Ακέραιος.

- REAL - Πραγματικός (δεκαδικοί αριθμοί).
- TEXT - Κείμενο.
- BLOB - χρησιμοποιείται για την αποθήκευση δυαδικών. αρχείων κάθε τύπου
  (π.χ. φωτογραφίες).

Η SQLite μπορεί να χρησιμοποιηθεί ως αυτόνομο προϊόν, αλλά και μέσα από τη γλώσσα Python, όπως θα γνωρίσουμε στο πλαίσιο του κεφαλαίου αυτού. Η ενσωμάτωση των λειτουργιών της SQLite σε ένα πρόγραμμα Python γίνεται εισάγοντας στην αρχή του τη δήλωση:

```
import sqlite3
```

10.3 Δημιουργία ή σύνδεση με μια Βάση Δεδομένων στην Python

Η δημιουργία μιας νέας κενής βάσης δεδομένων σε ένα πρόγραμμα Python με χρήση της SQLite είναι πολύ απλή:

Αρχικά εισάγουμε τη βιβλιοθήκη SQLite

```
import sqlite3
```

Η δημιουργία ή σύνδεση με μια βάση δεδομένων επιτυγχάνεται με την εντολή:

```
conn = sqlite3.connect('example.db')
```

Η παραπάνω εντολή δημιουργεί ένα αντικείμενο *χειριστή* που θα συνδεθεί με το αρχείο που περιέχει τη βάση δεδομένων example.db. Αν δεν υπάρχει τέτοιο αρχείο, δηλαδή (δεν υπάρχει η αναφερόμενη ΒΔ,), τότε θα δημιουργηθεί μια νέα βάση δεδομένων με αυτό το όνομα. Η βάση θα δημιουργηθεί στο φάκελο που υπάρχει και η Python, ενώ αν επιθυμούμε να αποθηκευτεί σε κάποιο διαφορετικό φάκελο θα πρέπει να το δηλώσουμε μαζί με το όνομα της βάσης, όπως για παράδειγμα:

```
conn = sqlite3.connect('c:/SQL/example.db')
```

Αν θέλουμε να δημιουργήσουμε προσωρινά στην κύρια μνήμη τη βάση δεδομένων, τότε θα γράψουμε:

```
conn =sqlite3.connect('c:/SQL/example.db:memory')
```

Τέλος, στο όνομα του αρχείου της ΒΔ μπορούμε να δώσουμε όποια επέκταση θέλουμε (ή χωρίς επέκταση), αλλά καλό είναι να χρησιμοποιούμε είτε το .db είτε το
.sql.

Μετά τη δημιουργία της *σύνδεσης* με τη ΒΔ, χρειαζόμαστε τη δημιουργία ενός αντικειμένου τύπου cursor μέσω του οποίου θα αλληλεπιδρούμε με τη ΒΔ, όπως για παράδειγμα, αν θέλουμε να προσθέσουμε εγγραφές:

```
curs = conn.cursor()
```

Μετά και τη δημιουργία *του αντικειμένου cursor*, μπορούμε να εκτελέσουμε μια εντολή SQL μέσω της εντολής .execute(). Τις εντολές SQL δεν έχει σημασία αν τις γράψουμε με πεζά η κεφαλαία γράμματα, αλλά στο κεφάλαιο αυτό οι εντολές γράφονται με ΚΕΦΑΛΑΙΑ γράμματα για να ξεχωρίζουν.

Στο τέλος της εργασίας μας, θα πρέπει να θυμηθούμε να *κλείσουμε* τη σύνδεση καθώς και το αντικείμενο cursor με τις εντολές:

```
curs.close()
conn.close()
```

10.4 Εισαγωγή, ενημέρωση και διαγραφή δεδομένων

Η βάση δεδομένων example.db που έχουμε δημιουργήσει, με τις εντολές της προηγούμενης παραγράφου, είναι κενή και δεν περιέχει δεδομένα. Για να εισαγάγουμε ορισμένα δεδομένα, ας βασιστούμε στο παράδειγμα μιας ΒΔ που χρησιμοποιείται από μια εταιρία μεταχειρισμένων αυτοκινήτων. Στη ΒΔ αυτή θέλουμε να δημιουργήσουμε έναν πίνακα που να περιέχει τα στοιχεία των αυτοκινήτων που διαθέτει η εταιρία.

Η δομή του πίνακα είναι:

Πεδίο	**Περιγραφή**	**Τύπος**
id	Κωδικός εγγραφής (αύξων αριθμός)	Ακέραιος
firma	Μάρκα (Εργ. κατασκευής) –μοντέλο	Κείμενο
cyear	Έτος κατασκευής	Κείμενο
fuel	Καύσιμο	Κείμενο
cc	Κυβισμός αυτοκινήτου	Ακέραιος
km	Χιλιόμετρα που έχει διανύσει	Ακέραιος
price	Τιμή	Πραγματικός

Η εντολή που θα χρησιμοποιήσουμε για τη δημιουργία του παρα- πάνωπαραπάνω πίνακα είναι:

```
# create a table
curs.execute("""CREATE TABLE cars
             (id INTEGER PRIMARY KEY not null,
              firma  TEXT,
              cyear   TEXT,
              fuel  TEXT,
                cyear TEXT, fuel
                   TEXT,
               cc  INTEGER,
              km   INTEGER,
              price   REAL)
              """);
                """);
```

Η εντολή αυτή θα δημιουργήσει μέσα στη ΒΔ example.db τον κενό πίνακα cars. Το πρώτο πεδίο αποτελεί το *πρωτεύον κλειδί* του πίνακα, το οποίο ορίστηκε ως ακέραιοςακέ- ραιος αριθμός, αύξων αριθμός εγγραφής, πεδίο που θα μπορούσε να είχε δηλωθεί ως `INTEGER PRIMARY KEY AUTOINCREMENT`

Το ***πρωτεύον κλειδί*** (primary key) ενός πίνακα είναι ένα πεδίο ή ένας συνδυασμόςσυνδυα- σμός πεδίων, που καθορίζει μονοσήμαντα κάθε εγ- γραφήεγγραφή του.

Η επόμενη λογική ενέργεια είναι να προσθέσουμε στοιχεία στον πίνακα που μόλις δημιουργήσαμε. Η εισαγωγή μιας εγγραφής επι-επιτυγχάνεται με την εντολή **INSERT**:

τυγχάνεται με την εντολή **INSERT**:

```
# Insert a row of data
curs.execute("INSERT INTO cars VALUES (1, 'Audi TT', '10/2010',
'Benzin',2000, 24000, 23500)"),
```

Αν θέλουμε να εισάγουμε τιμές μόνο σε ορισμένα από τα πεδία, τότε θα πρέπει δίπλα στο όνομα του πίνακα να καθορίσουμε ποια πεδία θα είναι αυτά:

curs.execute("INSERT INTO cars (firma, fuel, price) VALUES ('Audi A4', 'Diesel', 19000)")
('Audi A4', 'Diesel', 19000)")

Οι εισαγωγές που έχουμε πραγματοποιήσει, βρίσκονται ακόμα στη μνήμη του υπολογιστήυπο- λογιστή και δεν έχουν καταχωρηθεί στο αρχείο (πίνακα) της βάσης δεδομένων. Για να καταχωρηθούν, θα πρέπει να εκτε-

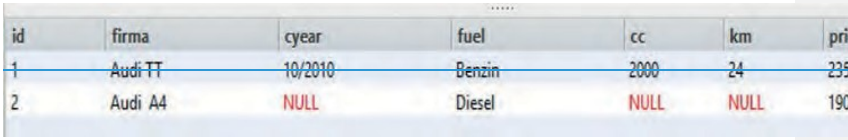
λέσουμεεκτελέσουμε την παρακάτω εντολή:

```
# save data to database
conn.commit()
```

Η εικόνα του πίνακα cars μετά τις δύο εισαγωγές είναι η ακόλου- θηακόλουθη:

Πίνακας 12-1. Ο πίνακας cars μετά από τις εισαγωγές

id	firma	cyear	fuel	cc	km	price
1	Audi TT	10/2010	Benzin	2000	24	23500.0
2	Audi A4	NULL	Diesel	NULL	NULL	19000.0



id	firma	cyear	fuel	cc	km	pri
1	Audi TT	10/2010	Benzin	2000	24	235
2	Audi A4	NULL	Diesel	NULL	NULL	190

Παρατηρούμε ότι για τις τιμές που δεν έχουν οριστεί έχει εισαχθεί η τιμή NULL, εκτός από το πεδίο id. Ο τρόπος που έχει οριστεί το πεδίο id, επιτρέπει στη ΒΔ να συμπληρώνει με αύξοντες αριθμούς το πεδίο αυτό.

Μπορούμε να εισάγουμε πολλές εγγραφές μαζί, αν χρησιμοποιή- σουμεχρησιμοποιήσουμε μια λίστα σε συνδυασμό με την εντολή- μέθοδο `executemany()`:

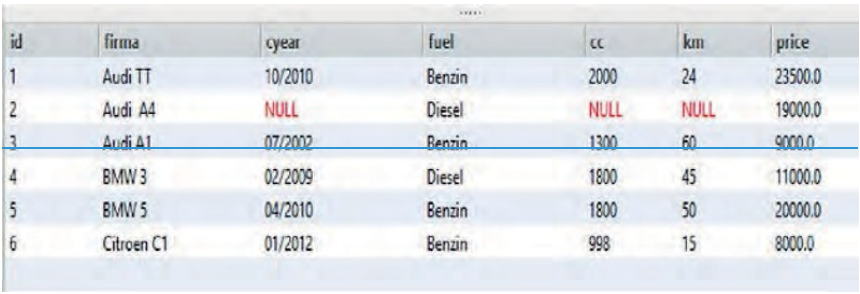
```
#       insert
multiple
records
autos = [(3, 'Audi A1', '07/2002', 'Benzin', 1300,
          60000, 9000), (4, 'BMW 3', '02/2009',
          'Diesel', 1800, 45000, 11000), (5, 'BMW 5',
          '04/2010', 'Benzin', 1800, 50000, 20000),-
          (6, 'Citroen C1', '01/2012', 'Benzin', 998,
          15000, 8000)]
```

```
(6, 'Citroen C1', '01/2012', 'Benzin', 998, 15000, 8000)]
curs.executemany("""INSERT INTO cars VALUES (?,?,?,?,?,?,?)""", autos)
# save data to database
conn.commit()
```

Παρατηρήστε τη χρήση επτά ερωτηματικών “?” για τη διαδοχική αντικατάσταση των τιμών.

Πίνακας 12-2. Ο πίνακας cars μετά από τις πολλαπλές εισα- γωγέςεισαγωγές

id	firma	cyear	fuel	cc	km	price
1	Audi TT	10/2010	Benzin	2000	24	23500.0
2	Audi A4	NULL	Diesel	NULL	NULL	19000.0
3	Audi A1	07/2002	Benzin	1300	60	9000.0
4	BMW 3	02/2009	Diesel	1800	45	11000.0
5	BMW 5	04/2010	Benzin	1800	50	20000.0
6	Citroen C1	01/2012	Benzin	998	15	8000.0



id	firma	cyear	fuel	cc	km	price
1	Audi TT	10/2010	Benzin	2000	24	23500.0
2	Audi A4	NULL	Diesel	NULL	NULL	19000.0
3	Audi A1	07/2002	Benzin	1300	60	9000.0
4	BMW 3	02/2009	Diesel	1800	45	11000.0
5	BMW 5	04/2010	Benzin	1800	50	20000.0
6	Citroen C1	01/2012	Benzin	998	15	8000.0

Τέλος, θα πρέπει να κλείσουμε τις συνδέσεις:

curs.close()

conn.close()

10.4.1 Ενημέρωση δεδομένων

Τα δεδομένα που περιέχει μια Βάση Δεδομένων συνεχώς αλλάζο- υναλλάζουν και για το λόγο αυτό, αν θέλουμε να αναφερθούμε σε μία συγ- κεκριμένησυγκεκριμένη μορφή της, μιλάμε για στιγμιότυποστιγ μιότυπο της Βάσης Δεδομέ- νωνΔεδομένων. Η εντολή sql η οποία επιτρέπει την τροποποίηση των δεδο- μένωνδεδομένων ενός πίνακα είναι η **UPDATE**.

Για παράδειγμα, θέλουμε στην εγγραφή με id =2 του πίνακα cars να εισαγάγουμε το μήνα και το έτος κατασκευής του αυτοκινήτου Audi A4 που είναι 05/2000. Το πρόγραμμαπρό- γραμμα που θα χρησιμοποιή- σουμεχρησιμοποιήσουμε είναι:

```
# update
import sqlite3
conn = sqlite3.connect('example.db')
curs = conn.cursor()
```

```
curs.execute("""UPDATE cars
             SET cyear = '05/2000'
             WHERE id = 2
             """)
conn.commit()
```

id	firma	cyear	fuel	cc	km	price
1	Audi TT	10/2010	Benzin	2000	24	23500.0
2	Audi A4	05/2000	Diesel	NULL	NULL	19000.0
3	Audi A1	07/2002	Benzin	1200	60	9000.0

Αν θέλαμε να μειώσουμε όλες τις τιμές των αυτοκινήτων κατά 10%, η αντίστοιχη εντολή θα ήταν:

```
curs.execute("""UPDATE cars
                SET price = price * 0,9
                """)
                """)
```

Διαγραφή δεδομένων

Η διαγραφή δεδομένων είναι μια σχετικά εύκολη διαδικασία και για το λόγο αυτό χρειάζεται προσοχή, ώστε να μη διαγραφούν χρήσι- μες πληροφορίες, κατά λάθος. Η διαγραφή δεδομένων από έναν πίνακα μέσω της sql, γίνεται με την εντολή **DELETE FROM**.

Παράδειγμα

Θέλουμε να διαγράψουμε από τον πίνακα cars όλα τα αυτοκίνητα που χρησιμοποιούν ως καύσιμο το πετρέλαιο (fuel = Diesel). Το αντίστοιχο πρόγραμμα είναι:

```
# delete rows
import sqlite3
conn = sqlite3.connect('example.db')
curs = conn.cursor()
curs.execute(""" DELETE FROM cars
                 WHERE fuel = 'Diesel'
                 """)""")
conn.commit()
```

Πίνακας 12-~~43~~. Ο πίνακας cars μετά από τις διαγραφές

id	firma	cyear	fuel	cc	km	price
1	Audi TT	10/2010	Benzin	2000	24	23500.0
3	Audi A1	07/2002	Benzin	1300	60	9000.0
5	BMW 5	04/2010	Benzin	1800	50	20000.0
6	Citroen C1	01/2012	Benzin	998	15	8000.0

id	firma	cyear	fuel	cc	km	price
1	Audi TT	10/2010	Benzin	2000	24	23500.0
3	Audi A1	07/2002	Benzin	1300	60	9000.0
5	BMW 5	04/2010	Benzin	1800	50	20000.0
6	Citroen C1	01/2012	Benzin	998	15	8000.0

10.4.2 Διαγραφή πίνακα

Εκτός από τη διαγραφή συγκεκριμένων δεδομένων, μπορούμε να διαγράψουμε ολόκληρο τον πίνακα με την εντολή **DROP TABLE** της sql, όπως φαίνεται στο ακόλουθο~~ακό λουθο~~ πρόγραμμα:

```
# Delete table
import sqlite3
conn = sqlite3.connect('example.db')
curs = conn.cursor()

curs.execute("DROP table cars")
conn.commit()
```

10.5 Αναζήτηση και ταξινόμηση δεδομένων

Η κυριότερη ίσως λειτουργία μιας ΒΔ, είναι η αναζήτηση δεδομέ- νων~~δεδομένων~~ που περιέχει. Για την αναζήτηση χρησιμοποιείται η εντολή SELECT της sql, η οποία έχει διάφορες μορφές, ανάλογα με το είδος της αναζήτησης. Τις μορφές αυτές θα γνωρίσουμε στη συνέ-~~συνέχεια.~~
χεια.

```
import sqlite3

conn = sqlite3.connect('example.db')
curs = conn.cursor()

# A listing of all the records in the table
curs.execute("SELECT * FROM cars")
print curs.fetchall()
```

```
# A listing of all benzin cars in the table
curs.execute("SELECT * FROM cars WHERE fuel =
'Benzin'")
print curs.fetchall()

print "A listing of all Audi cars"
for row in curs.execute("SELECT * FROM cars
            WHERE firma LIKE 'Audi%'");
    print row
conn.commit()
```

Η πρώτη αναζήτηση χρησιμοποιεί το χαρακτήρα μπαλαντέρ "*", “*”, για να επιστρέψειεπιστρέ- ψει όλα τα στοιχεία των εγγραφών από τον πίνακα cars. Οι εγγραφές εμφανίζονται με την εντολή print. Η εντολή αυτή συνδυάζεται με την fetchall(), που επιστρέφει όλες τις εγγραφές τις οποίες έχει ανακτήσει η προηγούμενη SELECT. Αντίστοιχη της fetchall() είναι η fetchone(), η οποία επιστρέφει μόνο μία εγγραφή. Αν θέλουμε την εμφάνιση μόνο συγκεκριμένων πεδίων, θα πρέπει να αναφέρουμε τα ονόματά τους ακριβώς. Για παράδειγμα: curs.execute("SELECT firma, cyear, price FROM cars")

να αναφέρουμε τα ονόματά τους ακριβώς. Για παράδειγμα:

```
curs.execute("SELECT firma, cyear, price FROM cars")
```

Η δεύτερη αναζήτηση χρησιμοποιεί την επιλογή WHERE για να περιορίσει το εύροςεύ- ρος της αναζήτησης. Στον περιορισμό μπορούν να χρησιμοποιηθούν και οι λογικοί τελεστές AND και OR. Για παράδε-Για παράδειγμα: curs.execute("SELECT * FROM cars WHERE fuel = 'Benzin' AND price > 10000 ")

ιγμα:

```
curs.execute("SELECT * FROM cars WHERE fuel = 'Benzin' AND price
> 10000 ")
```

Η τρίτη αναζήτηση εισάγει ένα πολύ ενδιαφέρον όρισμα στην επι- λογήεπιλογή WHERE, το όρισμα **LIKE**, το οποίο, σε συνδυασμό με χα- ρακτήρεςχαρακτήρες μπαλαντέρ, χρησιμοποιείται για την αναζήτηση τμήματος αλφαριθμητικού. Στη συγκεκριμένη περίπτωση επιλέγονταιεπιλέ- γονται οι εγ- γραφέςεγγραφές στις οποίες το πεδίο firma αρχίζει με τους χαρακτήρες "Audi". Υπάρχουν πολλοί τρόποι χρήσης του ορίσματος LIKE, αλ- λάαλλά δεν θα αναφερθούν εδώ, γιατί ξεφεύγουν από το σκοπό του κεφαλαίου αυτού.

Αναζήτηση με ταξινόμηση

Σε μια αναζήτηση, η εμφάνιση των εγγραφών γίνεται με τη σειρά που αυτές είναι αποθηκευμένες στον πίνακα. Αυτός ο τρόπος εμ- φάνισης εμφάνισης συχνά δεν μας ικανοποιεί, εφόσον θέλουμε τα αποτελέσ- ματα αποτελέσματα να είναι ταξινομημένα. Στην περίπτωση αυτή, η εντολή SELECT συμπληρώνεται με το όρισμα **ORDER BY**.

Για παράδειγμα, στην πρώτη αναζήτηση όλων των στοιχείων του πίνακα cars επιθυμούμε επι- θυμούμε τα αποτελέσματα να είναι ταξινομημένα κατά αύξουσα σειρά σε σχέση με την τιμή των αυτοκινήτων, δηλα- δηλαδή τα φτηνότερα πρώτα. Η εντολή είναι:

δή τα φτηνότερα πρώτα. Η εντολή είναι:

```
curs.execute("SELECT * FROM cars ORDER BY price ASC")
```

Αξίζει να σημειωθεί ότι η αύξουσα ταξινόμηση "ASC" δεδεν χρειάζεται να δηλωθεί, σε αντίθεση με την φθίνουσα "DESC" η οποία πάντο- τεπάντοτε δηλώνεται.

Η ταξινόμηση μπορεί να αναφέρεται σε δύο ή περισσότερα πεδία. Για παράδειγμα, στη δεύτερη αναζήτηση, των βενζινοκίνητων αυ- αυτοκινήτων, η εντολή θα μπορούσε να ήταν:

τοκινήτων, η εντολή θα μπορούσε να ήταν:

```
curs.execute("""SELECT * FROM cars
                WHERE fuel = 'Benzin'
             ORDER BY firma  ASC,  Price  DESC""")
```

Σημείωση: Τα στοιχεία της ΒΔ που δημιουργήσαμε, μπορούμε να τα δούμε με χρήσηχρή- ση άλλων προγραμμάτων. Μια εύκολη και δωρεάν λύση είναι το πρόσθετο *SQLite manager* που μπορούμε να εισα- γάγουμεεισαγάγουμε στο φυλλομετρητή ιστού Firefox. Ένα άλλο πρόγραμμα που μπορούμε να χρησιμοποιήσουμε, είναι το SQLite inspector. Οι εικόνες που έχουν χρησιμοποιηθεί στο κεφάλαιο, είναι από το πρόγραμμα SQLiteManager της SQLABS, το οποίο είναι εμπορικό προϊόν, αλλά με περιορισμένεςπεριορισμέ- νες δυνατότητες, διατίθεται ελεύθερα.

Τα τριπλά εισαγωγικά " " " και τα "" είναι ισοδύναμα και χρησιμο- ποιούνταιχρησιμοποιούνται, όταν μια παράσταση καταλαμβάνει περισσότερες από

μία γραμμές.

Δραστηριότητες - Ερωτήσεις - Ασκήσεις κεφαλαίου

10.6 Δραστηριότητες

Δραστηριότητα 1. Επαναληπτική δραστηριότητα κεφαλαίου

Μελετήστε και εκτελέστε το παρακάτω πρόγραμμα, όπου δημιουργείται η Βάση Δεδομένων example2.db και ο πίνακας users, ο οποίος περιέχει το όνομα, το τηλέφωνο, το mail και το συνθηματικό των χρηστών ενός συστήματος.

```
#
# Εισαγωγή του module sqlite3
import sqlite3
#
# Δημιουργία ή άνοιγμα της βάσης 'example2.db'
conn = sqlite3.connect('example2.db')
## Δημιουργία του αντικειμένου cursor object
cursor = conn.cursor()
# Έλεγχος αν ο πίνακας δεν υπάρχει και δημιουργία του
cursor.execute("'CREATE TABLE IF NOT EXISTS
    users(id INTEGER PRIMARY KEY,
    name TEXT,
    phone TEXT,
    email TEXT unique,
    password TEXT)"')
# Καταχώρηση αλλαγών στη βάση
conn.commit()
## Κλείσιμο της σύνδεσης conn
conn.close()

db = sqlite3.connect('example2.db')
cursor = db.cursor()
```

```
name1 =
'Panagiotis''Panagioris'
phone1 = '2105666858'
email1 = 'panos@example.com'
```

```
password1 = '12345'

name2 = 'Ioannis'
phone2 = '2105657241'
email2 = 'john@example.com'
password2 = 'abcdef'
# Εισαγωγή χρήστη 1 με χρήση τεχνικής υποκατάστασης με ερωτηματικά
cursor.execute('''INSERT INTO users(name, phone, email, password)VALUES(?,?,?,?)''', (name1, phone1, email1, password1))
print('First user inserted')

#
# Εισαγωγή χρήστη 2
cursor.execute('''INSERT INTO users(name, phone, email, password)VALUES(?,?,?,?)''', (name2, phone2, email2, password2))
print('Second user inserted')

db.commit()
```

Παρατήρηση: Στη δημιουργία του πίνακα, το πεδίο mail δηλώθηκε unique, δήλωση που επιβάλει το κάθε mail που εισάγεται να είναι διαφορετικό.

10.7 Ερωτήσεις

1. Πώς εισάγονται εντολές της sqlite3 μέσα σε κώδικα Python;
2. Με ποια εντολή μπορούμε να δημιουργήσουμε μια κενή βάση δεδομένων με όνομα trade.db;
3. Με ποιες εντολές κώδικα μπορούμε να δημιουργήσουμε έναν πίνακα customers με στοιχεία το AFM, Name, City και Phone στη ΒΔ trade.db;

Οι επόμενες ερωτήσεις - - ασκήσεις αναφέρονται στη ΒΔ trade.db και ειδικότερα στον πίνακα customers.

4. Με ποια εντολή μπορούμε να εισάγουμε στον πίνακα customers ~~customers~~ τον πελάτη με στοιχεία 012231212, Dimou Manolis, Athens, 212456789120;

5. Να εισαχθεί επιπλέον ο πελάτης 0123456789, Gatos Nikos, Athens, 212923876543. Επίσης να

~~6. Να~~ τροποποιηθεί η πόλη (city) του πελάτη Gatos σε Larisa.

6. ~~7.~~ Να διαγραφούν όλοι οι πελάτες που ως πόλη (city) έχουν την τιμή Athens.

10.8 Εργαστηριακές ασκήσεις

1. Δημιουργήστε μια βάση δεδομένων με όνομα school.db
2. Εισάγετε ένα πίνακα με όνομα students, ο οποίος θα περιέχει~~περιέχει~~ στοιχεία των μαθητών της τάξης σας, επιλέγοντας εσε- ίς τιμές για το . ~~Το~~ πλήθος και τον τύπο των στοιχείων ~~που θα περι- έχονται, θα το αποφασίσετε εσείς~~.
3. Εισάγετε τα στοιχεία ενός μαθητή και στη συνέχεια εισάγετε~~εισάγετε~~, όλα μαζί, τα στοιχεία 9 άλλων μαθητών.
4. Τροποποιήστε κάποιο από τα στοιχεία ενός μαθητή, όπως για παράδειγμα τη διεύθυνση ~~Διεύθυνση~~ ή το τηλέφωνό του.
5. Διαγράψτε ένα μαθητή.
6. Εμφανίστε όλους τους μαθητές ταξινομημένους ως προς το επίθετό τους.
   Εμφανίστε ξανά όλους τους μαθητές ταξινομημένους~~ταξινομημένους~~ ως προς το όνομά τους.

Σ ύ ν ο

ψ
η

Στο κεφάλαιο αυτό γνωρίσαμε τη χρήση της βιβλιοθήκης sqlite3, που επιτρέπει ~~η οποία επιτρέπει~~ τη χρήση εντολών της SQL ~~μέσα~~ σε προγράμματα Python. Παρουσιάστηκε το Σχεσιακό Μοντέλο Δεδομένων που

σχεσιακό μοντέλο δεδομένων που χρησιμοποιείται ευρέως στις βάσεις δεδομένων και οι βασικές εν- τολέςεντολές της sqlite3 για τη δημιουργία μιας ΒΔ. Στη συνέχεια παρου- σιάστηκε η δημιουργία ενός ή περισσοτέρων πινάκων μέσα στη ΒΔ, καθώς και οι εντολές που είναι απαραίτητες εντολές για τη διαχείριση μιας ΒΔ. Εντολές, όπωςβάσης δεδομένων. Οι εντολές αυτές είναι η INSERT, για την εισαγωγή δεδομένων σε πίνα- καπίνακα, η εντολή UPDATE, για την τροποποίηση στοιχείων ενός πίνακα και η εντολή DELETE, για τη διαγραφή των στοιχείων ενός πίνακα. Παρουσι- άστηκαν οι κυριότερες ίσως λειτουργίες σε μια ΒΔ, όπως η αναζή- τηση και η εμφάνιση στοιχείων της ΒΔ, λειτουργίες που βασίζονται σε παραλλαγές της εντολής SELECT. Τέλος, θα πρέπει να σημει- ωθεί ότι λύσεις στα παραδείγματα, εξυπηρετούν εκπαιδευτικούς σκοπούς και δεν είναι οι βέλτιστες..

των στοιχείων ενός πίνακα. Παρουσιάστηκαν οι κυριότερες ίσως λειτουργίες σε μια ΒΔ, όπως η αναζήτηση και εμφάνιση στοιχείων της ΒΔ, λειτουργίες που βασίζονται σε παραλλαγές της εντολής SELECT.

Τέλος, θα πρέπει να σημειωθεί ότι τα παραδείγματα των Βάσεων Δεδομένων που χρησιμοποιήθηκαν στο κεφάλαιο, εξυπηρετούν εκπαιδευτικούς σκοπούς και δεν είναι οι βέλτιστες λύσεις.

11

Αντικειμενοστρεφής προγραμματισμός ~~προγραμματισμός~~

11. Αντικειμενοστρεφής Προγραμματισμός

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται οι έννοιες που αφορούν τον αντικειμενοστρεφή προγραμματισμό.

Στη Β΄ τάξη ΕΠΑ.Λ. καθώς και στην αρχή του παρόντος διδακτικούδιδακτικού υλικού, είχε γίνει μια σύντομη αναφορά στον αντικειμενοστρε- φήαντικειμενοστρεφή προγραμματισμό με παραδείγματα. Στο κεφάλαιο αυτό θα γίνει εμβάθυνση στις σχετικές έννοιες και εξάσκηση στη δημιουργίαδημι- ουργία κλάσεων και αντικειμένων με χρήση της Python.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- αναγνωρίζουμε και να κατονομάζουμε χαρακτηριστικά του αντικειμενοστρε- φούς προγραμματισμού
- περιγράφουμε τις διαφορές του αντικειμενοστρεφούς προγραμματισμού
- περιγράφουμε τις διαφορές του αντικειμενοστρεφούς προγραμματισμού σε σχέση με το τον δομημένο προγραμματισμό
- δημιουργούμε απλά αντικείμενα και κλάσεις
- διακρίνουμε την έννοια της κλάσης από εκείνη του αντικειμένου
- αναγνωρίζουμε το ρόλο της κληρονομικότητας και του πολυμορφισμού στη δημιουργία επαναχρησιμοποιήσιμου κώδικα.

Λέξεις κλειδιά

Αντικειμενοστρεφής προγραμματισμός, κλάση, αντικείμενο, μέθο- δος, κληρονομικότηταμέθοδος, κληρονομικό τητα, πολυμορφισμός, προγραμματισμός οδηγο- ύμενοςοδηγούμενος από γεγονότα.

11.1 Αντικείμενα και Κλάσεις

Πριν την εμφάνιση του αντικειμενοστρεφούς προγραμματισμού, τα προγράμματα δομούνταν πάνω στην ιδέα της προτεραιότητας των ενεργειών έναντι των εννοιών/ δεδομένων/οντοτήτων. Έτσι, περιγ- ράφαμεπεριγράφαμε διαδικασίες και συναρτήσεις που επεξεργάζοντανεπε- ξεργάζονταν και τροποποιούσαν δεδομένα, τις οποίες θεωρούσαμε δομικά στοιχεία των προγραμμάτων μας. Οι εφαρμογές ακολουθούσαν το Διαδι- καστικό προγραμματισμόΔιαδικαστικό προγραμ ματισμό (procedural programming) και αποτελο-

ύνταναποτελούνταν από ακολουθίες εντολών, επαναλήψεις, δομές ελέγχου και υποπρογράμματα, προσέγγιση ιδιαίτερα λειτουργικήλειτουρ- γική στις περισ-σότερεςπερισσότερες των περιπτώσεων. Όμως, όταν κανείς ασχολιόταν με βάσεις δεδομένων ή με την Τεχνητή Νοημοσύνη, γινόταν φανερό ότι εξυπηρετούσε

καλύτερα μια προσέγγιση βασισμένη στην προτεραιότητα προτεραιότητα των εννοιών-οντοτήτων έναντι των ενεργειών ενεργειών/διεργασιών. Επίσης, όσο οι εφαρμογές με βάση το διαδικαστικό προγραμματισμό γίνονταν μεγαλύτερες και πιο περίπλοκες, η παρακολούθηση, η διόρθωση και η συντήρησή τους γινόταν δυσκολότερη δυσκολότερη.

Τι είναι λοιπόν ο αντικειμενοστρεφής Αντικειμενοστρεφής προγραμματισμός και πού εξυπηρετεί;

Ο Αντικειμενοστρεφής προγραμματισμός αλλάζει την εστίαση του προγραμματισμού από τις διαδικασίες στις έννοιες. Σε αυτές αναθέτει αναθέτει χαρακτηριστικά, τα οποία ονομάζουμε ονομά ζουμε ιδιότητες (attributes), και τα οποία επεξεργάζεται μέσω ειδικών συναρτήσεων που ονο- μάζουμε ονομάζουμε *μεθόδους*. Οι ιδιότητες δεν είναι άλλες από τα σχετικά με το αντικείμενο αντι- κείμενο δεδομένα, και οι μέθοδοι αποτελούν περιγραφές ενεργειών που αφορούν συγκεκριμένα σε αυτές τις ιδιότητες. Με τα θέματα αυτά θα ασχοληθούμε περισσότερο περισσό τερο αργότερα.

Έχει ενδιαφέρον να υπενθυμίσουμε ότι η Python, που είδαμε ότι πολύ καλά μπορεί να εξυπηρετήσει ένα διαδικαστικού τύπου προγραμματισμό, είναι στη βάση της μια αντικειμενοστρεφής γλώσσα. Έτσι, οι βασικές βιβλιοθήκες της έχουν δομηθεί με αντικειμενοστρεφή αντικειμενοστρεφή λογική και όλα στην Python αποτελούν στιγμιότυπα μιας κλάσης ή ενός τύπου δεδομένων. Μάλιστα και οι τύποι δεδο- μένων δεδομένων λειτουργούν στην ουσία σαν κλάσεις.

Εισαγωγή στις κλάσεις και τα αντικείμενα

Τι είναι ένα αντικείμενο στον αντικειμενοστρεφή προγραμματισμό; Πολύ απλά είναι ένα αντικείμενο της πραγματικής ζωής που το περιγράφουμε με χρήση κώδικα. Για παράδειγμα ένα αντικείμενο, που γνωρίζουμε όλοι από την πραγματική ζωή, είναι ένα αυτοκίνητο αυτοκίνητο. Πώς όμως θα περιγράφαμε το αυτοκίνητο που έχουμε στην οικογένειά οικογέ νειά μας ή ανήκει σε κάποιον άλλον, αν μας το ζητούσαν; Θα αναφέραμε προφανώς τα χαρακτηριστικά που διαθέτει, όπως για παράδειγμα τη μάρκα του, το μοντέλο, το έτος κατασκευής

του, ~~το χρώμα του, τα χιλιόμετρα που έχει κάνει, το ποσό κόστησε.~~

το χρώμα του, τα χιλιόμετρα που έχει κάνει, το πόσο κοστίζει.

Όλα αυτά τα χαρακτηριστικά αναφέρονται σε συγκεκριμένο αυτοκίνητο, ενώ υπάρχουν και χαρακτηριστικά, όπως ο αριθμός πλαισίου, που το χαρακτηρίζουν με μοναδικό τρόπο. Στον αντικειμενοστρεφή προγραμματισμό, τα ***χαρακτηριστικά*** (attributes) ενός αντικειμένουαντικειμένου ονομάζονται και ***ιδιότητες*** (properties).

Μια επόμενη ερώτηση θα μπορούσε να είναι, τι μπορούμε να κάνουμεκάνουμε με ένα αυτοκίνητοαυ- τοκίνητο;

Μπορούμε να:

- ~~Να το~~ ξεκινήσουμε~~.~~
- ~~Να~~ το επιταχύνουμε
- ~~Να~~ φρενάρουμε (να το επιβραδύνουμε) κ.λπ~~κλπ~~.

Στον Αντικειμενοστρεφή προγραμματισμό, αυτές οι ενέργειες ονομάζονταιονομάζονται **μέθοδοι (methods)**. Οι μέθοδοι επιτρέπουν στα αντικεί- μενααντικείμενα να κάνουν διάφορες ενέργειες, μέσω των οποίων, μπορούμε να ελέγχουμε τις ιδιότητες του αντικειμένου, όπως για παράδειγμα να ελέγχουμε την ταχύτητα του αυτοκινήτου.

Δημιουργία αντικειμένων

Τώρα που έχουμε διευκρινίσει τις ιδιότητες, τις μεθόδους και τρό- πουςτρόπους για να ρυθμίσουμερυθμί- σουμε ή να αλλάξουμε τις ιδιότητες μέσω μεθό- δωνμεθόδων, το ερώτημα είναι: πώς

δημιουργούμεδημιουρ- γούμε τελικά ένα αντικείμενο στον αντικειμενοστρεφή προγραμματισμό;

Η απάντηση είναι ότι ένα αντικείμενο δημιουργείται από μια ειδική μέθοδο που στην αντικειμενοστρεφή ορολογία ονομάζεται **κατασ- κευαστής κατασκευαστής** (constructor). Στην πραγματικήπραγ- ματική ζωή, ο κατασκευαστής θα ήταν η κατασκευαστική γραμμή του εργοστασίου που θα πα- ρήγαγεπαρήγαγε το αυτοκίνητο. Σε αυτήναυτή τη διαδικασία κατασκευής ορισμέ- ναορισμένα από τα χαρακτηριστικά του αυτοκινήτου θα είχαν συγκεκριμέ- νεςσυγκεκριμένες τιμές. Για παράδειγμα: η μάρκα θα ήταν TheCar, το μοντέλο A20, τα χιλιόμετρα μηδέν(0) κ.λπκλπ. Η αντίστοιχη διαδικασία στον προγραμματισμό, δηλαδή η απόδοση αρχικών τιμών σε ιδιότητεςιδιότη- τες ενός αντικειμένου που δημιουργείται, ονομάζεται *αρχικοποίηση* (initialization). Όπως είμαστε σίγουροι ότι η συγκεκριμένη κατασ- κευαστική γραμμή θα παράγει πάντα αυτοκίνητα TheCar, τύπου A20 με μηδέν χιλιόμετρα. Έτσι και στον προγραμματισμό, όταν ένας κατασκευαστής παράγει ένα αντικείμενο, γνωρίζουμε ότι οκατασκευαστική γραμμή θα παράγει πά-

ρισμένες

Κεφ.11: Αντικειμενοστρεφής προγραμματισμός

ντα αυτοκίνητα TheCar, τύπου A20 με μηδέν χιλιόμετρα. Έτσι και στον προγραμματισμό, όταν ένας κατασκευαστής παράγει ένα αντικείμενο, γνωρίζουμε ότι ορισμένες από τις ιδιότητές του θα έχουν συγκεκριμένες τιμές. Στα αυτοκίνητα γνωρίζουμε ότι μπορούμε να παραγγείλουμε ένα συγ- κεκριμένοσυγκεκριμένο μοντέλο, το οποίο στη στάνταρ έκδοσηέκ- δοση έχει συγκεκριμέ- νασυγκεκριμένα χαρακτηριστικά, μπορούμε όμως παράλληλα να επιλέξουμεεπιλέξου- με ορισμένα από τα χαρακτηριστικά του, όπως για παράδειγμα τις ζάντες του. Άρα, ορισμένες από τις ιδιότητες ενός αντικείμενου μπορούν να καθοριστούν αργότερα από τη βασική κατασκευή του.

Φυσικά, αφού υπάρχει μια ειδική μέθοδος η οποία κατασκευάζει αντικείμενα, η οποία παρεμπιπτόντως δεσμεύει μνήμη στον υπο- λογιστήυπολογιστή, θα πρέπει να υπάρχει και μια μέθοδος η οποία να κατασ- τρέφεικαταστρέφει τα αντικείμενα αυτά και να απελευθερώνει την αντίστοιχη μνήμη. Η μέθοδος αυτή ονομάζεται **αποδομητής** (destructor).

Μέχρι τώρα, με τον όρο αυτοκίνητο σκεφτόμαστε συνήθως ένα επιβατικό, κλειστό αυτοκίνητο (sedan). Η εταιρία όμως κατασκευά- ζεικατασκευάζει εκτός από τα επιβατικά, και φορτηγάφορ- τηγά και λεωφορεία. Τι γίνεται, αν θέλουμε να περιγράψουμε και αυτά; Μια λύση βέβαια θα ήταν, να περιγράψουμε κάθε είδος ως ξεχωριστό αντικείμενο. Θα δια- πιστώναμεδιαπι- στώναμε όμως, ότι όλα αυτά τα αντικείμενα έχουν κοινές ιδιότη- τεςιδιότητες, για παράδειγμα όλα έχουν τροχούς και όλα έχουν κοινές με- θόδουςμεθόδους, ότι όλα επιταχύνουν και φρενάρουνφρενά- ρουν. Δε θα ήταν λογικό και αποδοτικό να κάνουμε την ίδια εργασία για κάθε ένα αντικείμε- νοαντικείμενο χωριστά.

Μια λύση που χρησιμοποιεί ο αντικειμενοστρεφής προγραμματισμόςΑντικειμενοστρεφής προγραμματισμός είναι η δημιουργίαδημι- ουργία ενός *προτύπου* με όνομα *όχημα*, το οποίο έχει όλες τις ιδιότητες και τις μεθόδουςμεθόδους που είναι κοινές μεταξύ των επιβατικών των φορτηγών και των λεωφορείων. Αυτά τα πρότυπα που συγκεντρώνουν τα κοινά στοιχεία άλλων αντικειμένων, έχουν ένα ειδικό όνομα και ονομάζονται **κλάσεις** (classes). Οι κλάσεις μας επιτρέπουν να γνωρίζουμε πράγματα για αντικείμενα που α- κόμαακόμα δεν έχουν δημιουργηθεί.

~~Προγραμματισμός Υπολογιστών~~

Πολλές προσεγγίσεις του αντικειμενοστρεφούς προγραμματισμού αρχίζουν από τις κλάσεις τις οποίες προσομοιάζουν με το σχέδιο ενός αυτοκινήτου ή το αρχιτεκτονικό σχέδιο ενός σπιτιού. Εφαρμόζοντας~~Εφαρμόζοντας~~ αυτό το σχέδιο, μπορούμε να δημιουργήσουμε αντικεί- μενα~~αντικείμενα~~, για παράδειγμα αυτοκίνητα ή σπίτια, που έχουν κοινά στοι-

χείαστοιχεία, αλλά διαφέρουν και σε πολλά. Στις κλάσεις ενσωματώνονται και οι κατασκευαστές των αντικειμένων.

Για να κατανοήσουμε καλύτερα τη διαφορά μεταξύ κλάσεων και αντικειμένων, θα χρησιμοποιήσουμε το αγαπημένο παράδειγμα στους προγραμματιστές, της τάρταςτάρ- τας. Σε αυτό η κλάση τάρτα είναι μια συνταγή για τάρτες. Αν την έχεις, μπορείς να φτιάξεις τάρτες, όμως, όσο έχεις μόνο την συνταγή, δεν έχεις τάρτα για να τη φας. Το αντικείμενο τάρτα είναι μια «πραγματική» τάρτα, μια υλοποίηση της συνταγής με συγκεκριμένα χαρακτηριστικά, ενώ μπορούμε να έχουμε πολλά διαφορετικά αντικείμενααντικεί- μενα τάρτες κατασκευασμένα με την ίδια συνταγή.

Αν ξαναγυρίσουμε στο παράδειγμά μας, έχουμε μια κλάση που ονομάζεται όχημα και περιέχει τις κοινές ιδιότητες και μεθόδους των τριών τύπων οχημάτων. Αυτή η κλάση, ονομάζεται *κλάση γο- νέαςγονέας* (parent class) και οι τρεις τύποι των οχημάτων αποτελούν τις υπο-κλάσεις (subclasses). Είναι σημαντικό να προσέξουμε ότι οι κοινέςκοι νές ιδιότητες και μέθοδοι ορίζονται μόνο στην κλάση γονέα. Οι υποκλάσεις (κλάσεις παιδιά) αυτόματα, έχουν τις ίδιες ιδιότητες και μεθόδους, χωρίς να είναι απαραίτητο να τις ορίσουμε ξανά. Αυτή, η μεταβίβαση ιδιοτήτων και μεθόδων ονομάζεται **κληρονομικότη- τακλη- ρονομικότητα** (inheritance). Σε κάθε υποκλάση ορίζονται οι ιδιότητες και οι μέθοδοι που δεν καλύπτονται από την πατρική κλάση.

Επίσης παρατηρούμε ότι τα αντικείμενα που ανήκουν στις τρεις υποκλάσεις κάνουν όλα το ίδιο, μεταφέρουν άτομα και είδη, αλλά με ένα διαφορετικό τρόπο, φαινόμενο που ονομάζεται **πολυμορ- φισμόςπολυμορφισμός** (polymorphism).

Ορισμός κλάσης στην Python

Ο ορισμός μιας κλάσης στην Python είναι παρόμοιος με τον αυτόν μιας συνάρτησης, που γνωρίσαμε στα προηγούμενα κεφάλαια. Μόνο που, αντί για τη λέξη def, χρησιμοποιούμε τη *λέξη κλειδί* **class**. Για παράδειγμα, αν θέλουμε να ορίσουμε την κλάση όχημα με ορισμένες ιδιότητες και μεθόδους, από αυτές που αναφέραμε παραπάνω, η σύνταξη είναι:

```
class Vehicle:
    def __init__( self, color, price, wheels, speed):
        self.color=color
        self.price=price
        self.wheels=wheels
        self.speed=speed
    def accelerate(self, amount) :
        self.speed += amount
        return self.speed
    def decelerate(self, amount) :
        self.speed -= amount
        return self.speed
```

Παρατηρούμε ότι οι ιδιότητες της κλάσης εισάγονται σε μια ιδιαίτερη συνάρτηση την __init__ ως παράμετροι. Η συγκεκριμένη έχει μια ιδιαιτερότητα στο όνομα, αφού αρχίζει και τελειώνει με διπλές κάτω παύλες "__". Για τη μεταβλητή self που εμφανίζεται στην κλάση, θα μιλήσουμε αργότερα. Τέλος, οι μέθοδοι της κλάσης, έχουν τη μορφή συναρτήσεων, τοποθετούνται όμως, μέσα στο σώμα της κλάσης.

Ας δούμε πιο αναλυτικά τι σημαίνουν τα παραπάνω.

11.2 Στιγμιότυπα (αυτόματη αρχικοποίηση αντικειμένων)

Στην προηγούμενη παράγραφο ορίσαμε την κλάση Vehicle η οποία περιγράφει την έννοια του οχήματος. Η ύπαρξη της κλάσης δε σημαίνει και την αυτόματη ύπαρξη ενός αντικειμένου αυτής της κλάσης. Σημαίνει μόνο ότι υπάρχει απλώς η περιγραφή με την οποία μπορούμε να δημιουργήσουμε τέτοιο/α αντικείμενο/α.

Το αρχικό τμήμα κώδικα της κλάσης:

```
def __init__( self, color, price, wheels, speed):
        self.color=color
        self.price=price
        self.wheels=wheels
        self.speed=speed
```

είναι η μέθοδος που αποτελεί τον κατασκευαστή των αντικειμένων και καλείται επίσης *μέθοδος αρχικοποίησης τιμών*. Παρατηρούμε ότι δίνει τιμές στις ιδιότητες του αντικειμένου που ορίζονται στις παραμέτρους της ___init___. Κανονικά, μπορούμε να ορίσουμε εμείς τα ονόματα των μεθόδων, αλλά με το όνομα __init__, η γλώσσα αντιλαμβάνεται ότι είναι μια μέθοδος κατασκευαστής και έτσι καλείται αυτόματα με κάθε νέα δημιουργία αντικειμένου, τύπου Vehicle.

Για να δημιουργήσουμε ένα τέτοιο αντικείμενο πρέπει να καλέσουμε την κλάση Vehicle, με ορίσματα ή τιμές για κάθε μία από τις ιδιότητες του αντικειμένου που αντιστοιχούν στις παραμέτρους της μεθόδου ___init___. Στο αντικείμενο που δημιουργείται δίνουμε ένα όνομα, όπως για παράδειγμα:

mybeetle = Vehicle('yellow', 2000.00, 4, 80).

Έχουμε τώρα δημιουργήσει ένα όχημα κίτρινο, με τιμή 2000 ευρώ, τέσσερις τροχούς και ταχύτητα 80 χιλιόμετρα την ώρα. Το όνομά του είναι mybeetle. Το mybeetle είναι ένα αντικείμενο, ανήκει στην κλάση όχημα (Vehicle) και έχει συγκεκριμένες τιμές για τις ιδιότητες του οχήματος, αυτές που ορίσαμε με την κλήση της. Ένα τέτοιο αντικείμενο ονομάζεται **στιγμιότυπο** της κλάσης.

Δεν καλέσαμε ονομαστικά τη μέθοδο __init__, απλώς καλέσαμε την κλάση σχεδόν ως συνάρτηση, με τις τιμές που αφορούν το αντικείμενο που θέλουμε να δημιουργήσουμε για τις ιδιότητες της κλάσης, όπως αυτές εμφανίζονται ως μεταβλητές της μεθόδου.

~~*Κεφ.11: Αντικειμενοστρεφής προγραμματισμός*~~

~~Η~~ χρήση της παραμέτρουδεσμευμένης ~~λέξης~~ self

Η~~Η δεσμευμένη λέξη~~ **self** εμφανίζεται ως πρώτη παράμετρος σε όλες τις μεθόδους της κλάσης (δείτε στην προηγούμενη παράγραφο). Αυτή η παρά- μετρος~~παράμετρος~~ επιτρέπει στη μέθοδο να αναφέρεται στο ίδιο το αντικείμε- νο~~αντικείμενο~~ και όχι σε ολόκληρη την κλάση, πράγμα που τουλάχιστον ισχύ- ει~~ισχύει~~ στη γενική περίπτωση. Τοποθετώντας στη self το όνομα του αντικειμένου, διασφαλίζουμε ότι μια μέθοδος που θα κληθεί μέσω ενός αντικειμένου, θα επιδράσει μόνο σε αυτό.

Συγκεκριμένα, με μια κλήση, όπως η mybeetle.accelerate(10), ε- νεργοποιούμε~~ενεργοποιούμε~~ τις οδηγίες της μέθοδου accelerate για το αντικείμε- νο ~~αντικείμενο~~ mybeetle. Σε αυτή την περίπτωση ~~περίπτω- ση~~ η **def** accelerate(self, amount) αντιλαμβάνεται ως self το mybeetle.

Άρα, όταν γράφουμε **def** accelerate(self, amount) και μετά δίνουμε οδηγίες, αυτό που εννοούμε είναι “ορίστε πώς θα εφαρμόσεις την accelerate, σε όποιο αντικείμενο~~αντικείμε- νο~~ της κλάσης Vehicle την καλέσει”. Και υλοποιούμε με αυτόν τον τρόπο την αύξηση ταχύτητας στο συγκεκριμένο αντικείμενο, στην περίπτωσή μας, στο mybeetle μο- υ. **Προσοχή!!!** Η λέξη self δεν αποτελεί δεσμευμένη λέξη της py- thon αλλά μια σύμβαση μεταξύ των προγραμματιστών της γλώσ- σας~~μου~~.

Η~~Ή~~ ιδιαιτερότητα της μεθόδου __init__

Ας επιστρέψουμε τώρα στο λόγο που δεν καλούμε άμεσα την ίδια την μέθοδο, αλλά, αυτή καλείται έμμεσα μέσα από κλήση της κλά- σης~~κλάσης~~. Ο λόγος γι'~~γι'~~ αυτό είναι ότι στο πρόγραμμά μας σπάνια, γι' αυ- τό~~γι' αυτό~~, θα θελήσουμε να αναφερθούμε σε οποιαδήποτε ιδιότητα ή μέ- θοδο~~μέθοδο~~ της κλάσης γενικά. Έτσι, όταν αναφερόμαστε στην κλάση, πέρα από τον ορισμό της, θέλουμε να είμαστε βέβαιοι ότι έχουμε δημιουργήσει ένα αντικείμενο~~αντι- κείμενο~~ στο οποίο θα εφαρμόσουμε μεθό- δους~~μεθόδους~~ της κλάσης πάνω στις ιδιότητές του. Οπότε, όταν καλούμε την κλάση, έχουμε φροντίσει να καλείται η μέθοδος-συνάρτηση~~συνάρτη- ση~~ αρχικοποίησης __init__ η οποία κατασκευάζει αντικείμενα, λόγος για τον οποίο η μέθοδος __init ονομάζεται κατασκευαστής.

Φυσικά, με βάση τα παραπάνω, η χρήση της self στην __init__ έχει λιγότερο νόημα από ότι σε άλλες μεθόδους. Στις υπόλοιπες αναφέρεται στο αντικείμενο μέσα από το οποίο καλείται, ενώ εδώ αναφέρεται σε ένα αντικείμενο που κατασκευάζεται τώρα. Όμως,~~Όμως, αυτή η μικρή~~

~~αβαρία από πλευράς λογικής κάνει τη ζωή μας πιο εύκολη.~~

αυτή η μικρή αβαρία από πλευράς λογικής κάνει τη ζωή μας πιο εύκολη.

Σημαντικό είναι όταν ορίζουμε κατασκευαστές, να θυμόμαστε ότι κάθε ιδιότητα που χρησιμοποιείται από άλλες μεθόδους, πρέπει να αρχικοποιείται, πριν χρησιμοποιηθεί. Ο πιο εύκολος τρόπος για την αποφυγή λαθών είναι, να ορίζονται όλες μέσα στην __init__, καθώς δεν υπάρχει τρόπος, να πούμε στο πρόγραμμα να τρέξει δύο μεθόδους για αρχικοποίηση. Ας δούμε ένα παράδειγμα προβληματικά ορισμένης κλάσης:

```
class Vehicle1:
    def __init__( self, color, price, wheels):
        self.color=color
        self.price=price
        self.wheels=wheels
    def set_speed(self, speed = 100)
        self.speed=speed
    def accelerate(self, amount) :
        self.speed += amount
        return self.speed
    def decelerate(self, amount) :
        self.speed -= amount
        return self.speed
```

```
self.speed -= amount
return self.speed
```

Στην κλάση Vehicle1 αρχικοποιούνται στην ____init__ όλες οι ιδιότητες, εκτός από την ταχύτητα (speed) που αρχικοποιείται στην set_speed. Το προβληματικό εδώ είναι ότι οι ακόλουθες δύο γραμμές κώδικα οδηγούν σε πρόβλημα, αφού γίνεται απόπειρα να μεταβληθεί η ταχύτητα που δεν έχει αρχικοποιηθεί:

```
betty=Vehicle1('yellow', 2000.00, 4)
betty.accelerate(10)
```

Επομένως, η αρχικοποίηση ιδιοτήτων έξω από την ___init__ οδηγεί εύκολα σε λογικά λάθη. Φυσικά, αυτό δεν μας εμποδίζει να ορίσουμε με τέτοιο τρόπο την κλάση

200

μας, απλά απαιτεί ιδιαίτερη προ- σοχήπροσοχή στην αρχικοποίηση. Ας δούμε πώς χειριζόμαστεχειριζόμα- στε μια τέτοια περίπτωση, στο επόμενο παράδειγμα.

Αρχικοποίηση ιδιοτήτων αντικειμένων

Όπως έχουμε ήδη αναφέρει, τις περισσότερες φορές θέλουμε οι ιδιότητες διαφορετικώνδιαφορε- τικών αντικειμένων να έχουν διαφορετικές τιμές. Ας δούμε ένα πιο ολοκληρωμένο παράδειγμα.

Θα δημιουργήσουμε μια κλάση με όνομα dog που θα έχει τις ιδιό- τητεςιδιότητες breed (ράτσαρά- τσα), size (μέγεθος), color (χρώμα) και τις μεθόδο- υςμεθόδους eat (τρώω) και bark (γαυγίζω). Στη συνέχεια θα δημιουργήσου- μεδημιουργήσουμε δύο στιγμιότυπα της κλάσης, αρχικοποιώντας τις ιδιότητές τους~~ και θα καλέσουμε τις μεθόδους τους:~~

και θα καλέσουμε τις μεθόδους τους:

```
class Dog:
    def __init__(self, breed, size, color):
        self.breed=breed
        self.size = size
        self.color = color
        print "breed:","breed:",self.breed,"size:","size:",self.size,"color:","color:",self.color
    def eat(self,food):
        self.food = food
        print "I am eating ", food
    def bark(self):
        print "I am barking "

Max = Dog("terrier", "medium", "brown")
Rocky = Dog( "labrador", "large", "light brown")
Max.eat("bone")
Rocky.eat("meat")
```

Όπως εξηγήσαμε και νωρίτερα η δήλωση self.breed = breed απο- θηκεύειαποθηκεύει την τιμή της παραμέτρου **breed**, ως τιμή της αντίστοιχης ιδιότητας του αντικειμένου που κάλεσεκά- λεσε την __init__. Η κατασκευή των αντικειμένων μας γίνεται εδώ στις δύο πρώτες γραμμές μετά τον ορισμό της κλάσης. Η αρχικοποίησή τους όμως τελειώνει εκεί που τελειώνει ο κώδικας που γράψαμε! Δεν πρέπει να παρεμβάλ- λουμεπαρεμβάλλουμε κάποια εντολή που να αφορά ένα από τα δύο αυτά στιγμιό- τυπαστιγμιότυπα ανάμεσα στις τέσσερις

τελευταίες γραμμές του παραδείγμα- τόςπαραδείγματός μας.
Για να μην δημιουργούνται τέτοια προβλήματα η σωστή πρακτική είναι να δίνουμε αρχική τιμή σε όλες τις ιδιότητες των αντικειμένων μιας κλάσης μέσα στην __init__. Η αρχικοποίηση των ιδιοτήτων των αντικειμένων έξω από την __init__ αποτελεί κακή πρακτική και θα πρέπει να αποφεύγεται.

Δραστηριότητα εμπέδωσης παραγράφου

Δίνεται η παρακάτω κλάση:

```
class Car:
    def __init__(self, make):
        self.make=make
        self.speed = 60
    def speed_up(self,speed):
        self.speed = speed
        print "I am driving at a speed", self.speed, "km/h"
    def turn(self):
        print "I am turning … "
```

1. Ποιος είναι ο κατασκευαστής (constructor) της κλάσης;
2. Να καταγράψετε τις ιδιότητες της κλάσης, τις μεθόδους της, καθώς και τις ενέργειες που πραγματοποιούν.
3. Να προσθέσετε τις ιδιότητες color και year που αντιπρο- σωπεύουν ~~αντιπροσωπεύουν~~ το χρώμα και το έτος κυκλοφορίας του αυτοκι- νήτου~~αυτοκινήτου~~ αντίστοιχα και να αρχικοποιούνται~~αρχικοποι- ούνται~~ στον κατασκευ- αστή~~κατασκευαστή~~.
4. Να αλλάξετε τη μέθοδο turn, έτσι ώστε να δέχεται ως πα- ράμετρο~~παράμετρο~~ μια συμβολοσειρά~~συμβο- λοσειρά~~ που ορίζει αν το αυτοκίνητο θα στρίψει αριστερά ή δεξιά.
5. Να δημιουργήσετε τα παρακάτω στιγμιότυπα της κλάσης:
   I. Αντικείμενο με όνομα convertible και μάρκα “bmw”, χρώμα “μαύρο” και έτος κυκλοφορίας “2013”.
   II. Αντικείμενο με όνομα sedan και μάρκα “toyota”, χρώμα “κόκκινο” και έτος κυκλοφορίας “2009”.
6. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο convertible να στρίψει~~στρί- ψει~~ δεξιά.
7. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο sedan να τρέχει με 90 χιλ/ώρα.

11.3 Ιδιότητες και Μέθοδοι

Όπως αναφέραμε ήδη σε προηγούμενη παράγραφο, τα αντικείμε- ναντικείμενα είναι συγκεκριμένεςσυγκε- κριμένες οντότητες που χαρακτηρίζονται από ιδιότη- τεςιδιότητες και μπορούν να εκτελούν συγκεκριμένες ενέργειες.

Ας θυμηθούμε μερικά παραδείγματα:

Ένας **σκύλος** έχει τις ιδιότητες **φύλο**, **ράτσα**, **χρώμα** κ.λπ.κλπ και μπο- ρείμπορεί να εκτελεί διάφορες ενέργειες, όπως να **τρέξει**, να **γαυγίσει**, να **φάει**.

Ένα **αυτοκίνητο** έχει τις ιδιότητες **χρώμα**, **μάρκα**, **μοντέλο** κ.λπ.κλπ και μπορεί να εκτελείεκτε- λεί διάφορες ενέργειες όπως να **τρέχει**, να *σταματά-* ~~**σταματάει**, να στρίβει.~~

ει, να στρίβει.

Οι ενέργειες που μπορεί να κάνει ένα αντικείμενο ονομάζονται μέ- θοδοι~~μέθοδοι~~ και δεν είναι τίποτε άλλο παρά συναρτήσεις που ανήκουν σε ένα αντικείμενο.

Ορισμός ιδιοτήτων (attributes)

Στην Python, μια ιδιότητα μπορεί να δημιουργηθεί μέσα στην κλά- σηκλάση γράφοντας την πρόταση: *όνομα ιδιότητας ίσον με την επιθυμη- τήεπιθυμητή τιμή που θέλουμε να έχει*.

Παράδειγμα

```
class C:                          #δημιουργούμε την κλάση C
    classattr = "attr on class"
```

Όμως μια ιδιότητα μπορεί να δημιουργηθεί και από το στιγμιότυπο (αντικείμενο) της κλάσης γράφοντας το όνομα του αντικειμένου, τελεία, το όνομα της ιδιότητας, ίσον με την τιμή που θέλουμε να παίρνει.

Παράδειγμα

```
class C:                          #δημιουργούμε την κλάση C
    classattr = "attr on class"
...
>>> cobj = C() #δημιουργούμε το αντικείμενο cobj
>>> cobj.instattr = "attr on instance"     # ορίζουμε μια νέα ιδιότητα
                                           # για το αντικείμενο και
                                           # δίνουμε τιμή
```

Πρόσβαση σε ιδιότητες αντικειμένων

Προκειμένου να αποκτήσουμε πρόσβαση στις ιδιότητες των αντικειμένων~~αντικειμένων~~ που δημιουργήσαμε~~δημι- ουργήσαμε~~, χρησιμοποιούμε το λεγόμενο *dot notation*. Δηλαδή:

όνομα_αντικειμένου.ιδιότητα

Παράδειγμα

```
class C:                          #δημιουργούμε την κλάση C
      classattr = ""attr on class""
...
>>> cobj = C()                    #δημιουργούμε το αντικείμενο cobj
>>> cobj.instattr = "attr on instance"
>>>
>>> cobj.instattr
'attr on instance'
>>> cobj.classattr
'attr on class'
```

Η έκφραση **cobj.classattr** σημαίνει " Πήγαινε στο αντικείμενο στο οποίο αναφέρεται η μεταβλητή cobj και πάρε την τιμή της ιδιότητας **classattr**".

Ορισμός μεθόδων

Οι μέθοδοι είναι σημασιολογικά ίδιες με τις συναρτήσεις, αλλά υπάρχουν~~υπάρχουν~~ δύο *συντακτικές* διαφορές:~~συ- ντακτικές~~ διαφορές:

- Οι μέθοδοι ορίζονται μέσα στον ορισμό μιας κλάσης, προκειμένου να γίνει σαφής η σχέση μεταξύ της κλάσης και της μεθόδου.
- Η σύνταξη για την κλήση μιας μεθόδου είναι διαφορετική από τη σύνταξη για την κλήση μιας συνάρτησης.

Ως **υπογραφή** ορίζεται το όνομα, τα ορίσματα και η τιμή επιστρο- φής~~επιστροφής~~ της μεθόδου. Οι μέθοδοι ορίζονται μέσα στην κλάση, χρησιμοποιώντας την ίδια σύνταξη που χρησιμοποιούμε~~χρη- σιμοποιούμε~~, για να ορίσουμε μια συνάρτηση (βλ. κεφάλαιο 7). Αυτό με τη διαφορά ότι πάντα η πρώτη παρά- μετρος~~παράμετρος~~ πρέπει να είναι το στιγμιότυπο (αντικείμενο) της κλάσης του οποίου η μέθοδος καλείται. Επειδή το αντικείμενο που καλεί δεν είναι προκαταβολικά γνωστό, χρησιμοποιούμε στη θέση του μια παράμετρο την οποία

ονομάζουμε πάντα με το όνομα **self**. Με αυτόν τον τρόπο, η Python γνωρίζει σε ποιο αντικείμενο ανήκει η μέθοδος που καλείται και ποιου αντικειμένου τις ιδιότητες αλλάζει.

Παράδειγμα

```
class C:
        classattr = "attr on class"
        def my_print(self):
                print "my attribute is ", self.classattr
```

Πρόσβαση σε μεθόδους

Προκειμένου να αποκτήσουμε πρόσβαση στις μεθόδους των αντι- κειμένωναντικειμένων που δημιουργήσαμεδη- μιουργήσαμε, χρησιμοποιούμε το λεγόμενο *dot notation.* Δηλαδή:

όνομα_αντικειμένου.μέθοδος()

Είναι φανερό ότι οι μέθοδοι αυτές ορίζονται στην κλάση, αλλά όταν καλούνται αφορούναφο- ρούν πάντα ένα αντικείμενο ή στιγμιότυπο.

Παράδειγμα

Δημιουργούμε ένα αντικείμενο της κλάσης που ορίσαμε στο προη- γούμενο παράδειγμαπροηγούμενο παρά- δειγμα:

```
>>> cobj = C ()
```

Στη συνέχεια, καλούμε τη μέθοδο my_print του αντικειμένου ως εξής:

```
>>> cobj.my_print()
"“my attribute is attr on
class"”
```

Μέθοδοι που δεν αφορούν στιγμιότυπα

Μια ιδιαίτερη περίπτωση μεθόδου είναι οι λεγόμενες **στατικές μέ- θοδοιμέθοδοι**. Στο παράδειγμαπαρά- δειγμα της κλάσης σκύλος (Dog) είχαμε τη μέθο- δομέθοδο bark().

```
class Dog:
```

```
...
def bark(self):
    print "I am barking "
```

Ανεξάρτητα με το ποιο στιγμιότυπο καλεί τη συνάρτηση, το αποτέλεσμα θα είναι το ίδιο έτσι όπως την έχουμε ορίσει, μια δήλωση του εκάστοτε σκύλου ότι γαβγίζει. Σε αυτή την περίπτωση η παράμετρος self είναι περιττή. Για να το κάνουμε αυτό σαφές έχουμε ένα ειδικό διακριτικό, το @staticmethod, που προηγείται της συνάρτησης. Έτσι ο ορισμός γίνεται:

ένα ειδικό διακριτικό, το @staticmethod, που προηγείται της συ- νάρτησης. Έτσι ο ορισμός γίνεται:

```
class Dog:
    …
    @staticmethod
    def bark( ):
        print "I am barking "
```

Μια μέθοδος με το διακριτικό @staticmethod είναι στην πράξη μια κλασική συνάρτησησυνάρ- τηση την οποία όμως έχουμε τοποθετήσει μέσα σε μια κλάση. Σε περίπτωση μιας υποκλάσης που κληρονομεί την κλάση (δες επόμενη παράγραφο), ο ορισμός δεν επηρεάζεται και μένει ακριβώς ο ίδιος που ορίστηκε στην κλάση. Βέβαια, εφόσον χρησιμοποιήσουμε το διακριτικό @staticmethod, δε χρησιμοποι- ούμε το αντικείμενοδεν χρησιμοποιούμε το αντικείμε- νο ως παράμετρο.

Μπορούμε επίσης, να χρησιμοποιήσουμε μια μέθοδο κλάσης, μέ- σωμέσω του διακριτικού
@classmethod. Όσο δεν ασχολούμαστε με υποκλάσεις και κληρονομικότητα, μια μέθοδος κλάσης λειτουργεί όπως μια στατική μέθοδος. Όμως μια *μέθοδος κλάσης* είναι κλη- ρονομήσιμηκληρονομήσιμη και άρα, αν κληθεί από μια υποκλάση, θα αφορά αυ- τήαυτή. Για τον ορισμό μιας μεθόδου κλάσης περνάμε ως πρώτη πα- ράμετροπαράμετρο την κλάση, αντί του αντικειμένου, με τη δεσμευμένη λέξη cls που εννοεί την κλάση που την καλεί.

Περισσότερο νόημα έχει η χρήση μεθόδων κλάσης όταν χρησιμο- ποιούμε κληρονομικότηταχρησιμοποιούμε κληρονο- μικότητα (δες επόμενη παράγραφο), με μια συ- νηθισμένησυνηθισμένη χρήση να είναι η παρακολούθησηπαρακο- λούθηση του αριθμού των στιγμιότυπων της κλάσης.

Παράδειγμα

class Dog:

```
    no_inst = 0
    def __init__(self, breed, size, color):
        ...
            Dog.no_inst = Dog.no_inst + 1
    ...
    @classmethod
    def get_no_of_dogs(cls):
            return cls.no_inst
```

~~11.4~~ Χαρακτηριστικά: Κληρονομικότητα, Πολυμορφισμός

11.34.1 Κληρονομικότητα (Inheritance)

Το πιο σημαντικό χαρακτηριστικό του αντικειμενοστρεφούς προγραμματισμούΑντικειμενοστρεφούς Προγραμματισμού είναι η *κληρονομικότητα*.Κληρονομικότητα (Inheritance).

Κληρονομικότητα (Inheritance), ονομάζεται η ιδιότητα των κλάσεωνκλάσεων να μεταβιβάζουν τις ιδιότητέςιδιότη τές τους σε νέες κλάσεις, που λέ- γονταιλέγονται κληρονόμοι ή υποκλάσεις. Οι *υποκλάσεις* μπορούν να επαναχρησιμοποιήσουνεπαναχρησιμοποιήσουν τις μεταβιβάσιμες μεθόδους και ιδιότητες της κλάσης από την οποία κληρονομούν, αλλά και να προσθέσουν δικές τους.

Χρησιμοποιώντας λίγο πιο επίσημη ορολογία, λέμε ότι η κλάση που ορίζεται πρώτη είναι η βασική κλάση (base class) και η κλάση που κληρονομεί τη *βασική κλάση* είναι η *παράγωγη κλάση* (derived class). Εναλλακτικά, οι βασικές κλάσεις ονομάζονταιονομάζο- νται και *υπερκλάσεις*, ενώ οι παράγωγες κλάσεις ονομάζονται *υποκλάσεις*.

Στιγμιότυπα των υποκλάσεων μπορούν να χρησιμοποιηθούν όπουόπου απαιτούνται στιγμιότυπα των υπερκλάσεων, εφόσον η υποκ-

λάσηυποκλάση είναι, κατά κάποιον τρόπο, μια πιο εξειδικευμένη εκδοχή της υπερκλάσης, αλλά το αντίστροφο δεν ισχύει.

Παράδειγμα κληρονομικότητας, είναι μια υπερκλάση Vehicle (=Όχημα) και οι δύο πιο εξειδικευμένες υποκλάσεις της Car (=Αυτοκίνητο) και Bicycle (=Ποδήλατο), οι οποίες λέμε ότι *κληρο- νομούν*~~"κληρονομούν"~~ από αυτήν.

Από μια υποκλάση μπορούν να προκύψουν νέες υποκλάσεις που κληρονομούν από αυτήν, με αποτέλεσμα μια ιεραρχία κλάσεων που συνδέονται μεταξύ τους, ""~~"~~ανά γενιά",~~",~~ με σχέσεις κληρονομικό- τητας.~~κληρονομικότητας.~~ Για παράδειγμα, δυο υποκλάσεις της κλάσης Car (Αυτοκί- νητο~~Αυτοκίνητο~~), είναι η Convertible (ανοικτό) και η Sedan (Κλειστού τύπου).

Γενικότερα, *η Κληρονομικότητα χρησιμοποιείται όταν έχουμε μια ιεραρχία αντικειμένων*~~*αντικειμέ- νων*~~ *τα οποία πηγαίνουν από το γενικότερο στο ειδικότερο*. Χαρακτηριστικό παράδειγμα, ~~παρά- δειγμα,~~ είναι το προηγούμενο ~~που αναφέραμε (Όχημα Þ Αυτοκίνητο Þ Κλειστού τύπου).~~

που αναφέραμε (Όχημα ⇒ Αυτοκίνητο ⇒ Κλειστού τύπου).

Αν μελετήσουμε τις σχέσεις μεταξύ των διαδοχικών κλάσεων, θα δούμε ότι η "κάθε υποκλάση **είναι** η υπερκλάση της επόμενης". Δηλαδή:

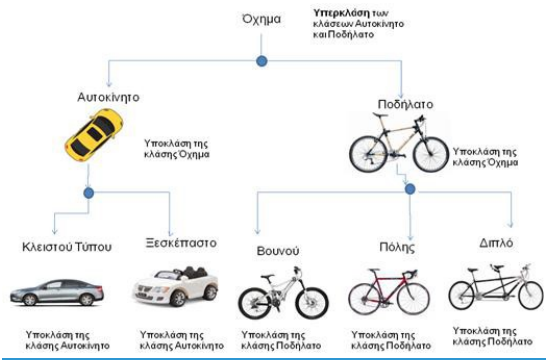
- ~~●~~Ένα αυτοκίνητο κλειστού τύπου είναι αυτοκίνητο.
- ~~●~~Ένα αυτοκίνητο είναι όχημα.

Η σχέση αυτή ισχύει όχι μόνο για την αμέσως ανώτερη υπερκλά- ση~~υπερκλάση~~, αλλά για κάθε υπερκλάση. Δηλαδή:

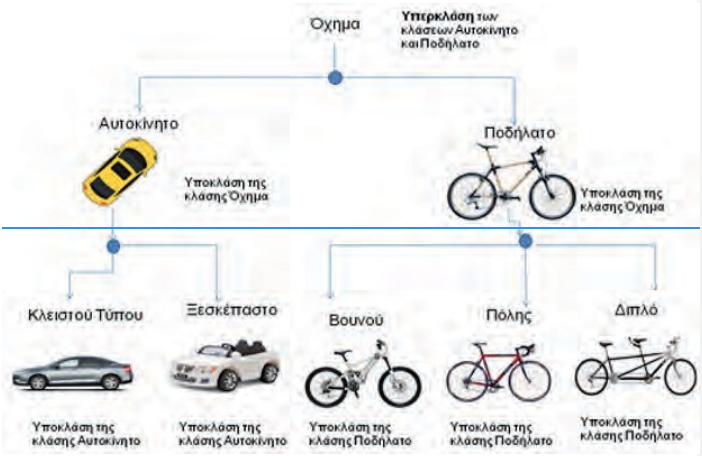
- ~~●~~Ένα αυτοκίνητο κλειστού τύπου είναι αυτοκίνητο.
- ~~●~~Ένα αυτοκίνητο κλειστού τύπου είναι και όχημα.

Κάθε φορά λοιπόν που έχουμε μια παρόμοια σχέση μεταξύ αντικε- ιμένων~~αντικειμένων~~, δηλαδή μπορούμε να πούμε ότι *κάτι είναι κάτι άλλο* (is-a, relationship), τότε μπορούμε/πρέπει~~πρέ- πει~~ να χρησιμοποιήσουμε κληρο- νομικότητα~~κληρονομικότητα~~.

Με αυτόν τον τρόπο δημιουργείται μια ιεραρχία, που για το παρά- δειγμα~~παράδειγμα~~ του οχήματος~~οχήμα- τος~~, θα μπορούσε να είναι αυτή που δίνεται στο παρακάτω σχήμα.


Όχημα
Υπερκλάση των κλάσεων Αυτοκίνητο και Ποδήλατο
Αυτοκίνητο
Υποκλάση της κλάσης Όχημα
Ποδήλατο
Υποκλάση της κλάσης Όχημα
Κλειστού Τύπου
Ξεσκέπαστο
Βουνού
Πόλης
Δίπλό
Υποκλάση της κλάσης Αυτοκίνητο
Υποκλάση της κλάσης Αυτοκίνητο
Υποκλάση της κλάσης Ποδήλατο
Υποκλάση της κλάσης Ποδήλατο
Υποκλάση της κλάσης Ποδήλατο

~~Κεφ.11: Αντικειμενοστρεφής προγραμματισμός~~



Τα πλεονεκτήματα από τη χρήση της κληρονομικότητας φαίνονται, αν παρατηρήσουμε~~παρατηρή σουμε~~ τις υποκλάσεις μιας κλάσης. Στο παράδειγμα που αναφέραμε παραπάνω, οι υποκλάσεις αυτοκίνητο και *ποδή- λατο*~~ποδήλατο~~ έχουν κάποια κοινά χαρακτηριστικά. Ορίζουμε αρχικά τα κοι- νά~~κοινά~~ χαρακτηριστικά αυτών των κλάσεων στην υπερκλάση Όχημα, τα κληρονομούμε και τα εξειδικεύουμε καταλλήλως στις υποκλάσε- ις~~υποκλάσεις~~.

Με τον ίδιο τρόπο, αν στο πρόγραμμά μας χρειαζόμαστε να ανα- παραστήσουμε~~αναπαραστήσουμε~~ *Ψάρια, Έντομα και Θηλαστικά*, τότε μπορούμε τα κοινά χαρακτηριστικά τους να τα ορίσουμε στην υπερκλάση Ζώο, από την οποία θα τα κληρονομούν. Στη συνέχεια βέβαια, η κάθε μία από τις υποκλάσεις *Ψάρι, Έντομο* και *Θηλαστικό* θα έχει το δικό της ιεραρχικό δέντρο, στο οποίο θα εξειδικεύεται καταλλήλως.

Με αυτόν τον τρόπο επιτυγχάνεται η επαναχρησιμοποίηση κώδικα που εξοικονομεί χρόνο, χρήμα και αυξάνει την αξιοπιστία του προγράμματος.

Ορισμός υποκλάσης στην Python

Για να ορίσουμε μια υποκλάση στην Python γράφουμε:

class ΌνομαΥποκλάσης(ΌνομαΥπερκλάσης):

όπου ΌνομαΥποκλάσης είναι το όνομα της υποκλάσης που θέλουμε~~θέλουμε~~ να ορίσουμε και ΌνομαΥπερκλάσης, είναι το όνομα της υπερ- κλάσης~~υπερκλάσης~~ από την οποία κληρονομεί.

Παράδειγμα 1

```
class BaseClass(object):
        def my_print(self):
                print "Hello ""Hello "


class InheritingClass(BaseClass):
        pass
```

Στο παραπάνω παράδειγμα ορίσαμε την υπερκλάση BaseClass, η οποία βλέπουμε ότι κληρονομεί από την έτοιμη (built in) κλάση object της Python. Στη συνέχεια, ορίσαμεορί- σαμε την υποκλάση InheritingClass, η οποία κληρονομεί από την κλάση BaseClass.

Αν θέλουμε να δημιουργήσουμε ένα στιγμιότυπο της κλάσης InheritingClass και να καλέσουμε τη μέθοδο my_print (), αυτό υ- λοποιείταιυλοποιείται ως εξής:

```
>>>x = InheritingClass ()
>>>x. my_print()
Hello
```

Βλέπουμε λοιπόν, ότι η υποκλάση μπορεί να χρησιμοποιεί όλες τις ιδιότητες και μεθόδους της υπερκλάσης.

Γιατί όμως είναι απαραίτητο η κλάση BaseClass να κληρονομεί από την object; Διότι με αυτόν τον τρόπο επιτρέπουμε στην υποκ- λάσηυποκλάση InheritingClass να καλεί μεθόδουςμε θόδους, όπως για παράδειγμα την ___init___ της κλάσης BaseClass. Αυτό θα γίνει καλύτερα κατα- νοητόκατανοητό με το παρακάτω παράδειγμα.:

Παράδειγμα 2

Ορίζουμε την παρακάτω κλάση:

```
class Person(object):
        def __init__(self, name):
                self._name= name
```

Στη συνέχεια ορίζουμε την υποκλάση Student ως εξής:

```
class
Student
(Person):
    def __init__(self, name,classatt):
        super (Student, self).__init__(name)
        self.classatt = classatt
```

Μέσα στον κατασκευαστή ___init___ της κλάσης Student χρησιμοποιούμε~~χρησιμοποιούμε~~ τη δήλωση:

super (Student, self).___init___(name), για να καλέσουμε τον κατασκευαστή~~κατασκευαστή~~ της υπερκλάσης~~υπερ κλάσης~~ Person. Αυτό μπορούμε να το κάνουμε ακριβώς, επειδή η κλάση Person κληρονομεί από την κλάση object.

Γι' αυτό είναι καλή πρακτική, όταν ορίζουμε μια κλάση που δεν είναι υποκλάση κάποιας~~κά ποιας~~ άλλης, να κληρονομεί πάντα από την κλάση object.

Δραστηριότητα εμπέδωσης ενότητας

- Ποιες είναι οι ιδιότητες της κλάσης Student;
- Δημιουργήστε στιγμιότυπο της κλάσης Student με όνομα "Γιάννης" και τάξη "Β".
- Πώς θα εμφανίσετε τις ιδιότητες του στιγμιότυπου που δημιουργήσατε στο προηγούμενο ερώτημα;

- Να ορίσετε υποκλάση Teacher της κλάσης Person, η οποία εκτός από το όνομαόνο- μα, θα έχει επιπλέον τις ιδιότητες κλάδος (field) και χρόνια υπηρεσίας (years).
- Δημιουργήστε τα παρακάτω στιγμιότυπα της κλάσης Teacher:
  - Στιγμιότυπο με όνομα “Αναστασίου”, ”,κλάδο “ΠΕ02” και χρόνια υπηρεσίας “11”.
  - Στιγμιότυπο με όνομα “Παπαχρήστου”, ”,κλάδο “ΠΕ03” και χρόνια υπηρεσίας “16”.

11.34.2 Πολυμορφισμός (polymorphism)

Στον αντικειμενοστρεφή προγραμματισμό, ο πολυμορφισμός αναφέρεται~~αναφέρεται~~ στη δυνατότηταδυ- ~~νατότητα~~ των αντικειμενοστρεφών μεταγλωττισ- τών~~μεταγλωττιστών~~/διερμηνευτών να αποφασίζουν δυναμικά ποια, από της ομώ- νυμες~~ομώνυμες~~ μεθόδους, είναι κατάλληλη να κληθεί ανάλογα με το αντικεί- μενο~~αντικείμενο~~ που την καλεί ή τον τύπο και τον αριθμό των παραμέτρων της.

11.~~4.~~3.3 Ενθυλάκωση και απόκρυψη~~Απόκρυψη~~ δεδομένων

Ενθυλάκωση (encapsulation) καλείται η ιδιότητα που προσφέρο- υν~~προσφέρουν~~ οι κλάσεις, να
«κρύβουν» τα ιδιωτικά δεδομένα τους από το υπόλοιπο πρόγραμμα και να εξασφαλίζουνεξασφα- λίζουν ότι, μόνο μέσω των δημόσιων μεθόδων τους, θα μπορούν αυτά να προσπελαστούν~~προσπε- λαστούν~~.

Μπορούμε να πούμε ότι η ενθυλάκωση έχει δύο χαρακτηριστικά:

1. **Συνδυάζει δεδομένα και μεθόδους σε ένα θύλακα (capsule).**

   Η κλάση ορίζεται ως ένας τύπος δεδομένων και περιλαμ-~~περιλαμβάνει:~~ βάνει:

   - Μεταβλητές ή πεδία τα οποία αντιπροσωπεύουν διάφορες ιδιότητες.
   - Μεθόδους οι οποίες χρησιμοποιούνται για να κατασκευάσουν στιγμιότυπα της κλάσης (κατασκευαστές - constructors) και μεθόδους οι οποίες χρησιμοποιούνται~~χρησι- μοποιούνται~~ για να αναπαραστήσουν τις λειτουργίες της κλάσης.

2. **Ελέγχει την πρόσβαση στα δεδομένα (information hiding).**

Η κλάση, όπως την ορίζουμε, εμφανίζει στους χρήστες και στις άλλες κλάσεις μια διεπαφή (interface), η οποία παρουσιάζει τις λειτουργίες της κλάσης. Με αυτό τον τρόπο, κρύβονται πληροφο- ρίες~~πληροφορίες~~ χαμηλού επιπέδου για τον τρόπο υλοποίησης των λειτουργι- ών~~λειτουργιών~~.

Έτσι, μπορούμε να χρησιμοποιούμε τα αντικείμενα, χωρίς να χρε- ιάζεται~~χρειάζεται~~ να καταλαβαίνουμε~~καταλα- βαίνουμε~~ τον τρόπο που έχουν υλοποιηθεί.

~~Κεφ.11: Αντικειμενοστρεφής προγραμματισμός~~

Παράδειγμα

Έχουμε έναν οδηγό αυτοκινήτου. Κάθε οδηγός μπορεί να οδηγεί το αυτοκίνητο, χωρίς να χρειάζεται να ξέρει πώς λειτουργεί η μη- χανήμηχανή του. Είναι αρκετό να γνωρίζει πώς ξεκινάει το αυτοκίνητο, πώς να βάζει βενζίνη, πώς να φρενάρει, πώς να επιταχύνειεπι- ταχύνει και πώς να σταματάει τη μηχανή. Αυτές οι βασικές λειτουργίες αποτε- λούναποτελούν τη *διεπαφή* (interface) του αυτοκινήτου. Μπορούμε να σκεφ- τούμεσκεφτούμε τη διεπαφή ως το σύνολο των ενεργειών που έχουμε τη δυ- νατότηταδυνατότητα να κάνουμε στο αυτοκίνητο, χωρίς να χρειάζεται να γνω- ρίσουμεγνωρίσουμε πώς αυτά υλοποιούνται.

Κρύβοντας την πολυπλοκότητα υλοποίησης του αυτοκινήτου από το χρήστη επιτρέπουμεεπι- τρέπουμε σε οποιονδήποτε, ανεξάρτητα από το αν είναι μηχανικός αυτοκινήτων ή όχι, να το οδηγήσει. Κατά τον ίδιο τρόπο, αποκρύπτοντας την υλοποίηση των λειτουργιών ενός αντι- κειμένουαντικειμένου από το χρήστη, επιτρέπουμε σε οποιονδήποτε να το χρησιμοποιήσει ανεξαρτήτως από το αν γνωρίζει ή όχι τον τρόπο υλοποίησής του.

Ένα άλλο παράδειγμα ενθυλάκωσης, είναι η χρήση του τελεστή in~~ στην Python. Θεωρήστε το παρακάτω τμήμα προγράμματος σε Python:~~

στην Python. Θεωρήστε το παρακάτω τμήμα προγράμματος σε Python:

```
if c in "συμβολοσειρά":
        print "βρέθηκε"
else:
        print "δε βρέθηκε"
```

Ο παραπάνω κώδικας θα εμφανίσει το μήνυμα " βρέθηκε", αν ο χαρακτήρας c βρεθείβρε- θεί μέσα στη συμβολοσειρά "συμβολοσειρά", αλλιώς θα εμφανίσει το μήνυμα "δε βρέθηκε". Όμως, με ποιο τρό-ποτρόπο η Python υλοποιεί τη λειτουργία του τελεστή in; Ποιον αλγόριθ- μοαλγόριθμο αναζήτησης χρησιμοποιεί; Αυτό εμείς δε χρειάζεται να το ξέρο- υμεξέρουμε, γιατί η ακριβής υλοποίηση της λειτουργικότητας του τελεστή αυτού είναι κρυμμένη για τους χρήστες του τελεστή. Αυτή η απόκ- ρυψηαπόκρυψη πληροφοριών είναι βασική στον αντικειμενοστρεφή προγ- ραμματισμόπρογραμματισμό και στη γλώσσα Python.

Πλεονεκτήματα

Η χρήση της ενθυλάκωσης έχει δύο βασικά πλεονεκτήματα:

1. Επιτρέπει Έλεγχο.
   - Η χρήση του αντικειμένου γίνεται μόνο μέσα από τις μεθόδους του.
   - Δεν επιτρέπεται η άμεση αλλαγή των δεδομένων του αντικειμένου.

   Με αυτόν τον τρόπο δεν επιτρέπεται σε άλλα αντικείμενα να αλλάξουν άμεσα τα δεδομένα ενός άλλου αντικειμένου.
2. Επιτρέπει αλλαγές
   - Η χρήση του αντικειμένου δεν αλλάζει, αν αλλάξουν τα δεδομένα.
   - Πρέπει όμως, να παραμείνουν τα ίδια πρότυπα μεθόδων.

Για να επανέλθουμε στο παράδειγμα του αυτοκινήτου που χρησιμοποιήσαμε παραπάνω~~χρησιμοποιήσαμε πα- ραπάνω~~, ο οδηγός μπορεί να οδηγήσει το αυτοκί- νητο~~αυτοκίνητο~~ είτε είναι βενζινοκίνητο, είτε πετρελαιοκίνητο, είτε, αν κινείται με αέριο, χωρίς κανένα πρόβλημα. Αυτό συμβαίνει~~συμβαί- νει~~ διότι τα αυτοκί- νητα~~αυτοκίνητα~~ αυτά έχουν διαφορετικό εσωτερικό μηχανισμό λειτουργίας, κρυμμένο για τον οδηγό. Αυτό σημαίνει ότι, ακόμα κι αν ο μηχανισμός~~μηχανισμός~~ λειτουργίας του αυτοκινήτου αλλάξει, το αυτοκίνητο λειτο- υργεί~~λειτουργεί~~ με τον ίδιο τρόπο από την πλευρά~~πλευ- ρά~~ του οδηγού.

Τροποποιητές πρόσβασης (Access Modifiers)

Η Ενθυλάκωση στον αντικειμενοστρεφή προγραμματισμό επιτυγχάνεται~~επιτυγχάνεται~~ με τους

τροποποιητές πρόσβασης.

Οι βασικοί τροποποιητές πρόσβασης είναι οι εξής:

- **public:** Η πρόσβαση στην κλάση, δεδομένα, μεθόδους είναι ανοικτή σε όλους
- **private:** Η πρόσβαση στα δεδομένα και τις μεθόδους είναι περιορισμένη μόνο στην κλάση
- **protected:** η πρόσβαση είναι ανοικτή σε κλάσεις που κληρονομούν από την ίδια κλάση.

~~Κεφ.11: Αντικειμενοστρεφής προγραμματισμός~~

Διαφορετικές γλώσσες προγραμματισμού μπορεί να προσφέρουν και άλλους τροποποιητέςτρο- ποποιητές πρόσβασης πέρα από τους βασικούς. Για παράδειγμα η Java έχει 4 , η C++ 3 και η C# 5.

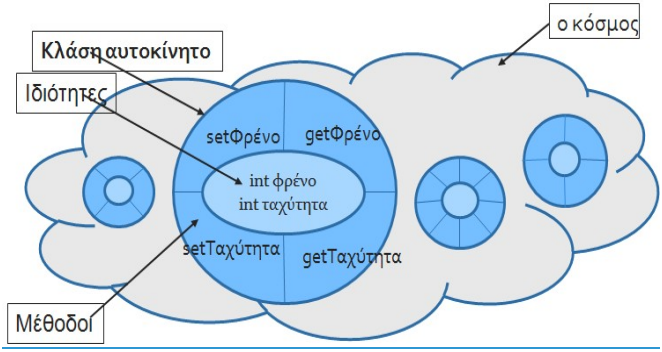
Μια κλάση ελέγχει την πρόσβαση στα διάφορα μέλη της (δεδομέναδεδομένα, μεθόδους) με τους τροποποιητές πρόσβασης.

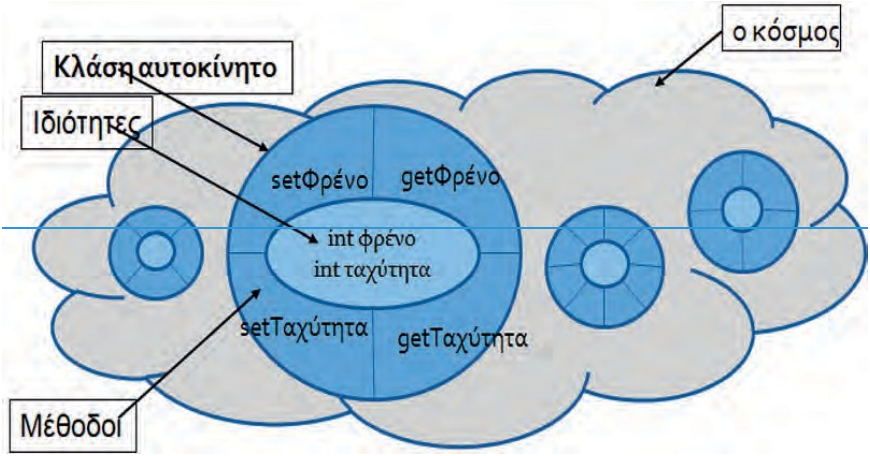
Οι βασικοί κανόνες σύμφωνα με τους οποίους χρησιμοποιούνται οι παραπάνω τροποποιητέςτρο- ποποιητές πρόσβασης σε μια κλάση, είναι οι εξής:

- Τα μέλη μιας κλάσης, όπως οι υψηλού επιπέδου μέθοδοι, ορίζονται public για να εκθέσουν τις λειτουργίες τους σε χρήστες/κλάσεις.
- Τα μέλη μιας κλάσης που παρέχουν λεπτομέρειες υλοποίησης, ορίζονται private για να κρύψουν πληροφορίες από τους χρήστες/κλάσεις.

Πώς όμως αποκτούμε πρόσβαση στις private πληροφορίες μιας κλάσης; Συνήθως χρησιμοποιούμε ειδικές μεθόδους που τις ονομάζουμεονομάζουμε get() και set(). Οι μέθοδοι αυτές επιστρέφουν τις τιμές των private ιδιοτήτων και τους δίνουν τιμή αντίστοιχα.

Αν πάρουμε πάλι το παράδειγμα του αυτοκινήτου, μπορούμε να πούμε ότι οι ιδιότητες ιδιότη- τες του αυτοκινήτου, όπως για παράδειγμα φρένο ή ταχύτητα, ορίζονται ως private πληροφορίες και ότι χρησιμοποιούνταιχρησιμοποιούνται ειδικές μέθοδοι για την πρόσβαση στις πληροφορί- εςπλη- ροφορίες αυτές, όπως φαίνεται και από το παρακάτω σχήμα.


ο κόσμος
Κλάση αυτοκίνητο
Ιδιότητες
setΦρένο
getΦρένο
int φρένο
int ταχύτητα
setΤαχύτητα
getΤαχύτητα
Μέθοδοι


ο κόσμος
Κλάση αυτοκίνητο
Ιδιότητες
setΦρένο
getΦρένο
int φρένο
int ταχύτητα
setΤαχύτητα
getΤαχύτητα
Μέθοδοι

Οι περισσότερες αντικειμενοστρεφείς γλώσσες, όπως Java ή C++, χρησιμοποιούν τους παραπάνω τροποποιητές πρόσβασης άμεσα για να ορίσουν περιορισμούς πρόσβασης στις ιδιότητες και στις μεθόδους τους.

Στην Python όμως, αυτό μπορεί να οριστεί μόνο με σύμβαση. Για παράδειγμα τα ονόματα όλων των private μεθόδων και ιδιοτήτων ξεκινούν με διπλή κάτω παύλα.

11.45 Οδήγηση από Γεγονότα - Μηνύματα

Έχουμε ήδη αναφέρει τη σημασία που έχουν τα γεγονότα στον αντικειμενοστρεφή προγραμματισμό. Είναι ο τρόπος με τον οποίο καθορίζεται η συμπεριφορά των αντικειμένων: Μια συμπεριφορά καθορίζεται από ένα γεγονός, εξωτερικό του αντικειμένουαντι- κειμένου, αλλά επίσης από ένα χειριστή του γεγονότος, δηλαδή από ένα κομμάτι κώδικα -στην Python συνάρτηση- μέσα στο αντικείμενο που υλο-ποιείυλοποιεί την ανταπόκρισηανταπό- κριση του αντικειμένου στο γεγονός.

Υπάρχει μια πληθώρα πιθανών *γεννητριών* γεγονότων. Ξεκινά από το ίδιο το σύστημασύ- στημα, τα μέρη του και ιδιαίτερα τις μονάδες ει- σόδουεισόδου εξόδου, από εφαρμογές που τρέχουν πάνω σε αυτό και επικοινωνούν με τη δική μας, από άλλα αντικείμενα της εφαρμο- γήςεφαρμογής μας. Ένα γεγονός προκαλεί ένα μήνυμα, ένα σύνολο συνα- φώνσυναφών με το γεγονός δεδομένων, που έχει σκοπό να προκαλέσει κάποια προγραμματισμένη αντίδραση. Φυσικά ένα μήνυμα - ειδικά όταν ανταλλάσσεται μεταξύ εφαρμογών ή scripts ή ακόμα και αντι-κειμένωναντικειμένων- μπορεί να είναι το ίδιο ένα γεγονός.

Τα πιο γνωστά μας γεγονότα είναι αυτά που έχουν σχέση με το ποντίκι και το πληκτρολόγιο.πλη- κτρολόγιο. Για παράδειγμα, το κλικ στο αριστερό κουμπί του ποντικιού, το πάτημα του πλήκτρου return, το πάτημα του πλήκτρου «G», αποτελούν, τουλάχιστον εν δυνάμειδυ- νάμει, γεγονότα. Για να έχουν ενδιαφέρον για την εφαρμογή μας βέβαια, θα πρέπει να υπάρχει κάποια προγραμματισμένη ανταπόκριση σε αυτά.

Ένα κρίσιμο ζήτημα είναι πώς εξασφαλίζουμε ότι το σωστό αντικε- ίμενοαντικείμενο θα λάβει το μήνυμα που σχετίζεται με ένα γεγονός που το αφορά. Υπάρχουν διάφοροι τρόποι διαχείρισης αυτού του θέμα- τοςθέματος, όπως:

Κεφ.11: Αντικειμενοστρεφής προγραμματισμός

Ένας τρόπος διαχείρισης των μηνυμάτων είναι να έχουμε χειρισ- τέςχειριστές (κατάλληλες μεθόδους) σε διάφορα αντικείμενα, χωρίς κεντρι- κόκεντρικό έλεγχο, ή τουλάχιστον χωρίς εμφανή κεντρικό έλεγχο, με τα αντικείμενα να αποφασίζουν ποια μηνύματα τα αφορούναφο- ρούν. Σε μια παραλλαγή της προηγούμενης περίπτωσης μπορεί να έχουμε ένα μοντέλο πελάτη/εξυπηρετητή, όπου ο πελάτης ζητά από τον εξυ- πηρετητήεξυπηρετητή κάποια ενέργεια και αυτός ανταποκρίνεται, αν κρίνει ότι τον αφορά. Ένα καλό παράδειγμα είναι η περιήγηση στο Διαδίκ- τυοΔιαδίκτυο, όπου κάθε φορά που αλλάζουμε σελίδα ζητάμε μια εξυπηρέ- τησηεξυπηρέτηση από έναν διαδικτυακό εξυπηρετητή και αυτός ανταποκρίνεται στέλνοντάςστέλνο- ντάς μας τη σελίδα που ζητήσαμε, εφόσον κρίνει ότι έχουμε δικαίωμα πρόσβασης σε αυτή.

Ένας άλλος τρόπος διαχείρισης είναι, να έχουμε κεντρικό έλεγχο, ο οποίος δρομολογείδρομο λογεί τα μηνύματα στους κατάλληλους χειριστές. Οι χειριστές αυτοί σε έναν καλό αντικειμενοστρεφή σχεδιασμό θα ανήκουν, βέβαια, σε διάφορα αντικείμενα. Η δρομολόγησηδρο- μολόγηση γίνεται μέσω ενός *διανεμητή* (dispatcher), στον οποίο κατευθύνονται τα μηνύματα που προκλήθηκαν από τα γεγονότα και ο οποίος γνωρί- ζειγνωρίζει από πριν ποια αντικείμενα αφορά κάθε γεγονός και τα ενημε- ρώνειενημερώνει κατάλληλα.

Μια ειδική περίπτωση κεντρικού ελέγχου είναι αυτή του μοντέλου *Παρατηρητή ή εγγραφής/ δημοσίευσης*, επίσης γνωστή και με το όνομα *μοντέλο του ηθοποιού*. Η γενική ιδέα παραμένει ίδια, όμως ο μοναδικός διανεμητής δε γνωρίζει προκαταβολικάπροκαταβολι- κά, από τον αρ- χικόαρχικό του προγραμματισμό, ποια αντικείμενα αφορά κάθε γεγονός. Τα αντικείμενα, που το γνωρίζουν, εγγράφονται σε μια κατάσταση -στην Python μπορεί να πρόκειται για ένα λεξικό- χειριστών μιας κατηγορίας γεγονότων, που τηρείται στοστον διανεμητή και του επιτ- ρέπειεπιτρέπει, όταν προκύψει γεγονός αυτής της κατηγορίας, να γνωρίζει τα αντικείμενα που ενδιαφέρονται γι'ψι' αυτό.

Το όνομα *μοντέλο του ηθοποιού* προέκυψε από τον παραλληλισ- μόπαραλληλισμό με την πρακτικήπρακτι- κή του Hollywood όπου, ενώ οι ηθοποιοί εμφανί- ζονταιεμφανίζονται σε πρακτορεία και εταιρίεςεταιρί- ες παραγωγής εκδηλώνοντας το ενδιαφέρον τους για ρόλους, δε συνηθίζεται να παίρνουν τηλέφω- νατηλέφωνα για να ενημερωθούν για την εξέλιξη. Οι εταιρίες συνήθως τους

ενημερώνουν «μη μας καλέσετε, θα σας καλέσουμε εμείς», για να γλυτώσουν τα αναρίθμητα τηλέφωνα ηθοποιών που απορρίφθηκαν~~απορρίφθηκαν~~ για το ρόλο. Ο διανεμητής σε ρόλο εταιρίας παραγωγής καλεί όσα αντικείμενα–ηθοποιούς κρίνει ότι πρέπει.

Το μοντέλο του ηθοποιού χρησιμοποιείται πολύ συχνά σε γραφικές~~γραφικές~~ διεπαφές χρήστη~~χρή στη~~ (GUIs) και ο λόγος είναι πολύ απλός: ότι όλα τα γραφικά στοιχεία ενός GUI, όπως κουμπιά, check-box, πεδία κειμένων, ετικέτες, μενού, μπάρες κύλισης και ό,τι άλλο υπάρχει σε ένα γραφικό περιβάλλον μπορούν, ειδικά σε συνδυασμό με μια είσοδο από το ποντίκι ή το πληκτρολόγιο, να παράγουν μια ποικιλία~~ποικιλία~~ γεγονότων.

Είναι τόσα πολλά τα είδη γεγονότων για τα οποία απαιτείται να προγραμματιστούν χειριστές, που, αν ο έλεγχος έπρεπε να τα χειριστείχειριστεί κατά περίπτωση, ο βρόχος της επανάληψης του διανεμητή θα γινόταν απίστευτα πολύπλοκος. Κατά συνέπεια, γίνεται η επι- λογή~~επιλογή~~ να ενημερωθούν για κάθε κατηγορία γεγονότος όλα τα εγ- γεγραμμένα~~εγγεγραμ μένα~~ αντικείμενα και αυτά να κρίνουν αν τα αφορά, βάσει για παράδειγμα της θέσης στην οθόνη ενός κλικ του ποντικιού ή της πληροφορίας ποιο γραφικό στοιχείο ή παράθυρο είναι εκείνη τη στιγμή ενεργό.

Η

Ή εγγραφή χειριστών γεγονότων στην Python

Μια γραφική διεπαφή χρήστη που περιλαμβάνεται στις βιβλιοθή- κεςβιβλιοθήκες της Python και ήδη γνωρίσαμε, είναι η Tkinder. Θα τη χρησι- μοποιήσουμεχρησιμοποιήσουμε εδώ για να δώσουμε ένα παράδειγμα. Σε αυτήν όλα τα γεγονότα γραφικής διεπαφής ανήκουν σε μια κλάση γεγονότων. Όλοι οι χειριστές γεγονότων εγγράφονται σε GUI widgets, που περιλαμβάνουν τα γραφικά στοιχεία της διεπαφής, όπως κουμπιά, ετικέτες κ.λπκλπ. Τα GUI widgets χρησιμοποιούνται το καθένα για το χειρισμό συγκεκριμένων τύπων γεγονότωνγε- γονότων, όπως τα κλικ στο ποντίκι ή η πίεση κάποιου πλήκτρου στο πληκτρολόγιο.

Ένα πρόγραμμα, που απλά, ανοίγει ένα frame που περιλαμβάνει ένα κουμπί εγκατάλειψηςεγκα- τάλειψης/κλεισίματος (quit button), θα δείχνει κά- πωςκάπως έτσι:

```
#!/usr/bin
/env
python
import
Tkinter
as tk
class
Application(tk.Fr
ame):
        def __init__(self,
                master=None):
                tk.Frame.__init__
                (self, master)
                self.grid()
                self.crea
                teWidget
                    s()
        def createWidgets(self):
                self.quitButton=    =    tk.Button(self,      text='Quit', commanc
                self.quitB
utton.grid() app =
Application()
app.master.title('Sample
application')
app.mainloop()
```

Τα widgets του Tkinder είναι υλοποιημένα ως κλάσεις. Έτσι, αν θέλουμε να υλοποιήσουμε ~~υλοποι- ήσουμε~~ ένα δικό μας κουμπί, αρκεί, για παρά-

δειγμαπαράδειγμα, να δημιουργήσουμε το σχετικό αντικείμενο το οποίο όμως, θα πρέπει να συνδέεται με κάποιο παράθυρο ή πλαίσιο.

Στον παραπάνω κώδικα έχουμε υλοποιήσει έναν αυτόνομα εκτε- λούμενοεκτελούμενο κώδικα (script) σε Python, ο οποίος εισάγει το Tkinder και τα ονόματα του module, αφού προηγουμένως, για διευκόλυνση το μετονομάσει σε tk. Η κλάση Application κληρονομείκληρο- νομεί από την έτοιμη κλάση Frame του Tkinter, οπότε και η μέθοδος κατασκευαστής της συνάρτησης καλεί την αντίστοιχη της Frame. Κατόπιν ορίζονται τα widgets που θα χρησιμοποιηθούν και ορίζεται το κουμπί κλεισίμα- τοςκλεισίματος (quitButton).

Οι γραμμές self.grid() και self.quitButton.grid(), απλώς ρυθμίζουν την εμφάνιση στην οθόνη των δύο αντικειμένων. Στις τρεις τελευ- ταίεςτελευταίες γραμμές ορίζεται ένα αντικείμενο κλάσης Application με όνο- μαόνομα app, του οποίου το κείμενο τίτλου ορίζεται σε ‘Sample Application’ και εκκινεί ο κύριος βρόχος της εφαρμογής σε αναμο- νήαναμονή γεγονότων από το ποντίκι ή το πληκτρολόγιο.

Έχουμε καλέσει έτσι τον κατασκευαστή της κλάσης Button και έ- χουμε δημιουργήσειέχουμε δημιουργή- σει το αντικείμενο quitButton, ενώ ταυτόχρονα το έχουμε συνδέσει με το γραφικό

αντικείμενο πρώτου επιπέδου στο οποίο ανήκει. Δεν μπορούμε να έχουμε κουμπιάκου μπιά που «πλέουν» στη διεπαφή μας, οπότε τα τοποθετούμε μέσα σε παράθυρα ή frames. Για να γίνουν τα πράγματα χειρότερα, ο μοναδικός μας χειριστής είναι επίσης τετριμμένος: είναι η εντολή quit που ενερ- γοποιείταιενεργοποιείται, όταν ενεργοποιηθεί το κουμπί quitButton.

Το Tkinter μας δίνει τη δυνατότητα να κάνουμε και πιο ενδιαφέ- ροντα πράγματα το ίδιο εύκολα. Τα widgets προσφέρουν τη μέθο- δομέθοδο bind με την οποία μπορούμε να συνδέσουμε γεγονότα με συγ- κεκριμένασυγκεκριμένα widgets, στην πραγματικότητα μας επιτρέπειεπιτρέ πει να συνδέ- σουμεσυνδέσουμε τρία πράγματα:

- έναν τύπο γεγονότος, για παράδειγμα το αριστερό κλικ του ποντικιού ή το πάτημα του κουμπιού ENTER στο πληκτρολόγιο
- ένα widget, όπως ας πούμε ένα συγκεκριμένο τύπο κουμπιού στο GUI μας
- και μια συνάρτηση χειριστή γεγονότων.

Δείτε για παράδειγμα τον κώδικα:

```
#!/usr/bin/env python
import Tkinter as tk
class Application(tk.Frame):
  def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.grid()
        self.createWidgets()
  def createWidgets(self):
        self.quitButton = tk.Button(self, text='Quit')
        self.quitButton.bind('<Return>', quitButtonEventHandler)
        self.quitButton.bind('<Button-1>', quitButtonEventHandler)
        self.quitButton.grid()
 quitButtonEventHandler)
          self.quitButton.grid()
 def quitButtonEventHandler(self):
        self.quit app = Application()
app.master.title('Sample application')
app.mainloop()
```

Εδώ, στον ορισμό του quitButton δεν περιλαμβάνεται εντολή. Ό- μωςΌμως, στις δύο επόμενεςεπό- μενες γραμμές τόσο το κουμπί ENTER όσο και το αριστερό κλικ του ποντικιού, μέσα από τα τυπικά ονόματα που τους δίνει το Tkinter, συνδέονται αφενός με το widget quitButton και αφετέρου με τη μέθοδο quitButtonEventHandler. Μέθοδο την οποία, για λόγους απλότητας του παραδείγματος, έχουμε αφήσει σε απλοϊκή μορφή, δηλαδήδηλα- δή να κλείνει την εφαρμογή.

Αν συμβεί οποιοδήποτε από τα δύο γεγονότα, το πάτημα του ENTER στο πληκτρολόγιοπλη- κτρολόγιο ή το πάτημα του αριστερού κουμπιού του ποντικιού πάνω στο κουμπί quitButton, η εφαρμογή μας θα κλείσει.

Δραστηριότητες - Ερωτήσεις - Ασκήσεις κεφαλαίου

11.56
Δραστηριότητες

Δραστηριότητα 1

Να ορίσετε κλάση με όνομα Akeraios, η οποία θα έχει την ιδιότητα timi και τις παρακάτωπα- ρακάτω μεθόδους:

I. Anathese_timi(timi), η οποία θα αναθέτει τιμή στην ιδιότητα του αντικειμένου. II. Emfanise_timi(), η οποία θα εμφανίζει την τιμή της ιδιότη-
ταςιδιότητας του αντικειμένου.

Στη συνέχεια να δημιουργήσετε στιγμιότυπο της κλάσης Akeraios με όνομα Artios και να χρησιμοποιήσετε τις παραπάνω μεθόδους για να δώσετε την τιμή 14 στην ιδιότητα του αντικειμένου και να την εμφανίσετε.
~~ιδιότητα του αντικειμένου και να την εμφανίσετε.~~

Δραστηριότητα 2

Να ορίσετε κλάση με όνομα Student η οποία θα έχει τρεις ιδιότητες για τον αριθμό μητρώου, το ονοματεπώνυμο και τους βαθμούς σε ~~8 μαθήματα (πίνακας).~~ 8 μαθήματα (πίνακας).

Επίσης να ορίσετε τις παρακάτω μεθόδους της κλάσης:

I. Μια μέθοδο για την ανάθεση τιμών στις ιδιότητες ενός αντικειμένου~~αντικειμένου~~.

II. Μια μέθοδο για την εμφάνιση των τιμών των ιδιοτήτων ενός ~~ενός~~ αντικειμένου.

III. Μια μέθοδο για τον υπολογισμό και την επιστροφή του μέσου ~~μέσου~~ όρου βαθμολογίας~~βαθμολογίας~~ στα 8 μαθήματα.

Στη συνέχεια να ορίσετε ένα στιγμιότυπο της κλάσης Student με όνομα Chris και να χρησιμοποιήσετε τις παραπάνω μεθόδους για να δώσετε τιμή στις ιδιότητες του αντικειμένου, να τις εμφανίσετε και να υπολογίσετε το μέσο όρο βαθμολογίας του.

Δραστηριότητα 3

Συζητήστε, συμπληρώστε ή επεκτείνετε την παρακάτω σύνοψη βασικών όρων του κεφαλαίου.

Αντικειμενοστρεφής προγραμματισμός: ένας τρόπος προγραμματισμού~~προγραμματισμού~~ στον οποίο τα δεδομένα και οι λειτουργίες (συναρτήσεις~~συναρτήσεις~~) είναι οργανωμένα σε αντικείμενα~~αντικείμενα~~.

Κλάση: Ένας τύπος ορισμένος από το χρήστη. Μια συνταγή δημιουργίας αντικειμένων~~δημιουργίας αντικειμένων~~.

Αντικείμενο: μια δομή δεδομένων και επεξεργαστική ενότητα που έχει δημιουργηθεί~~δημιουργηθεί~~ βάσει κάποιας κλάσης.

Στιγμιότυπο μιας κλάσης: ένα αντικείμενο το οποίο δημιουργήθηκε χρησιμοποιώντας~~δημιουργήθηκε χρησιμοποιώντας~~ τη συγκεκριμένη κλάση.

Ιδιότητα αντικειμένου: μία από τις μεταβλητές που σχετίζονται με ένα αντικείμενο.

Μέθοδος: μια συνάρτηση η οποία ορίζεται μέσα σε έναν ορισμό κλάσης και χρησιμοποιείται~~χρησιμοποιείται~~, συνήθως, από τα στιγμιότυπά της.

~~Κεφ.11: Αντικειμενοστρεφής προγραμματισμός~~

Κληρονομικότητα: η μεταφορά χαρακτηριστικών μιας κλάσης σε άλλες κλάσεις (υποκλάσεις), που ρητά προέρχονται από την αρχι- κή~~αρχική~~. Οι υποκλάσεις μπορούν να επαναχρησιμοποιήσουν τις μετα- βιβάσιμες~~μεταβιβάσιμες~~ μεθόδους και ιδιότητες της γονικής τους κλάσης, αλλά και να προσθέτουν δικές τους.

Πολυμορφισμός: αναφέρεται στη δυνατότητα των αντικειμενοσ- τρεφών μεταγλωττιστών~~αντικειμενοστρεφών μεταγλωτ τιστών~~ / διερμηνευτών να αποφασίζουν δυναμι- κά~~δυναμικά~~ ποια από τις ομώνυμες μεθόδους είναι κατάλληλη να κληθεί, ανάλογα με το αντικείμενο που την καλεί ή τον τύπο και τον αριθ- μό~~αριθμό~~ των παραμέτρων της.

Ενθυλάκωση: καλείται η ιδιότητα που προσφέρουν οι κλάσεις, να «κρύβουν» τα ιδιωτικά δεδομένα τους από το υπόλοιπο πρόγραμ- μα~~πρόγραμμα~~ και να εξασφαλίζουν πως, μόνο μέσω των δημοσίων μεθόδων τους, θα μπορούν αυτά να προσπελαστούν.

11.~~6~~7 Ερωτήσεις~~-~~ ~~Ασκήσεις~~

1. Ποια είναι τα βασικά χαρακτηριστικά του αντικειμενοστρε- φούς προγραμματισμού~~αντικειμενοστρεφούς προγραμμα τισμού~~;
2. Σε τι διαφέρει ο αντικειμενοστρεφής από το δομημένο~~Δομημένο~~ προγραμματισμό;
3. Τι είναι η κλάση και τι το αντικείμενο (στιγμιότυπο) μιας κλάσης;
4. Πώς δηλώνουμε μια κλάση στην Python;
5. Τι σημαίνει κληρονομικότητα και τι πολυμορφισμός;
6. Δώστε ένα παράδειγμα κληρονομικότητας.
7. Τι είναι η ενθυλάκωση και ποια τα βασικά της πλεονεκτή- ματα~~πλεονεκτήματα~~.
8. Τι είναι ένα γεγονός; Δώστε παραδείγματα.

Σύνοψη - ανακεφαλαίωση

Στο κεφάλαιο αυτό παρουσιάστηκαν οι βασικές αρχές του Αντικει- μενοστρεφούς~~Αντικειμενοστρεφούς~~ Προγραμματισμού. Γνωρίσαμε τη βασική ορολογία και μάθαμε να δημιουργούμε κλάσεις και να χρησιμοποιούμε αντι- κείμενα.~~αντικείμενα.~~ Τέλος παρουσιάστηκαν οι έννοιες της κληρονομικότητας και της ενθυλάκωσης και πώς τα γεγονότα καθορίζουν τη συμπε- ριφορά των αντικειμένων.

12

Εισαγωγή στην υπολογιστική σκέψηΣκέψη

12. Εισαγωγή στην Υπολογιστική Σκέψη

Εισαγωγή

Η σύγχρονη εποχή χαρακτηρίζεται από ραγδαίες τεχνολογικές ε- ξελίξειςεξελίξεις, που επιφέρουνεπι- φέρουν σημαντικές αλλαγές σε διαφόρους τομείς της οικονομικής, εργασιακής και κοινωνικής ζωής. Καθημερινά, ακόμα και σε απλές ασχολίες, αυξάνεται η απαίτησηαπαίτη- ση χρήσης πλη- ροφοριακώνπληροφοριακών συστημάτων, υπολογιστικών συσκευών και υπηρε- σιώνυπηρεσι- ών του Διαδικτύου. Σε ένα μεταβαλλόμενο τεχνολογικά περιβάλ- λονπεριβάλλον, ο σύγχρονος πολίτης καλείται να αναπτύξει ικανότητες και γνώσεις, που να του επιτρέπουν να αξιοποιεί κατά τον καλύτερο δυνατό τρόπο την τεχνολογία της εποχής του.

Στο κεφάλαιο αυτό θα αναπτυχθεί μια σημαντική -για τη σύγχρονη εποχή- νοητικήνοητι- κή διαδικασία οργάνωσης και επίλυσης προβλήμα- τοςπροβλήματος: η *υπολογιστική σκέψη*, που ενισχύει το συνδυασμό των ψηφι- ακώνψηφιακών τεχνολογιών με τις ανθρώπινες ιδέες. Μέσα από απλά πα- ραδείγματαπαραδείγματα θα προσεγγίσουμε τα βασικά χαρακτηριστικά της *υπο- λογιστικής**υπολο- γιστικής* *σκέψης* και θα εξοικειωθούμε με τις σχετικές έννοιες.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- περιγράφουμε την έννοια της υπολογιστικής σκέψης (ΥΣ)
- αναφέρουμε τα χαρακτηριστικά της ΥΣ
- εξηγούμε τη σημασία της ΥΣ στην επίλυση προβλημάτων της καθημερινής ζωής
- αναλύουμε ένα υπολογιστικό πρόβλημα, περιγράφοντας τις βασικές διαδικασίεςδιαδικα- σίες με τις οποίες αντιμετωπίζεται, όπως τη διάσπασή του σε απλούστερα, τη λογική οργάνωση των δεδομένων, την αναγνώριση προτύπων, την αναγνώριση αναγνώ- ριση πιθανών λύσεων, την περιγραφή της λύσης του με αλγοριθμικό τρόπο, τη γενίκευση της λύσης του.

Λέξεις κλειδιά

Υπολογιστική Σκέψη, επίλυση προβλήματος, διάσπαση προβλή- ματοςπροβλήματος, αφαίρεση, οπτική αναπαράσταση δεδομένων, αλγοριθμική λύση, γενίκευση λύσης.

Διδακτικές ενότητες

Η αύξηση της πολυπλοκότητας λόγω της αξιοποίησης προηγμέ- νης~~προηγμένης~~ τεχνολογίας σε μεγάλη κλίμακα και η αναγκαιότητα υπηρεσι- ών~~υπηρεσιών~~, που βασίζονται σε υπολογιστικά συστήματα και στο Διαδίκτυο, απαιτούν ολοένα και μεγαλύτερες ικανότητες ανάλυσης, σχεδίαανάλυ- σης~~, σχεδίασης~~, υλοποίησης, ελέγχου, διάγνωσης και αντιμετώπισης λαθών. Για να μπορεί ο άνθρωπος να χρησιμοποιεί ανάλογα εργαλεία και να σχεδιάζει εφαρμογές, πρέπει να διαθέτει ικανότητες επίλυσης προβλήματος με τη βοήθεια υπολογιστικών συστημάτων, όπως είναι η Υπολογιστική Σκέψη.

Η **Υπολογιστική Σκέψη** (Computational Thinking) αποτελεί μια νοητική λειτουργία με χαρακτηριστικά, όπως είναι:

- Η αναπαράσταση δεδομένων, μέσω αφαιρέσεων, σε μεγάλα και πολύπλοκα προβλήματα. Για παράδειγμα, μοντέλα για την καθημερινή διανομή φρέσκου γάλατος, σε όλα τα σημεία πώλησης μιας μεγάλης πόλης, ή η προσομοίωση ενός φυσικού φαινομένου για την πρόγνωσή του.
- Η αυτοματοποίηση λύσεων μέσω της αλγοριθμικής σκέψης.
- Η βελτιστοποίηση λύσης.
- Η γενίκευση και μεταφορά μιας διαδικασίας επίλυσης προβλημάτων σε μια ποικιλία προβλημάτων, όπως για παράδειγμα η εύρεση εξόδου σε λαβυρίνθους ~~λαβυρίν θους~~ με διαφορετικό σχήμα.

Όταν κάποιος προσπαθεί να λύσει ένα πρόβλημα με αλγοριθμικό τρόπο, ώστε στη συνέχεια να περιγράψει τη λύση του και να την υλοποιήσει, για να επιλύεται αυτόματααυτό- ματα από κάποια υπολογιστική μηχανή, καλείται να απαντήσει σε ερωτήματα σχετικάσχε- τικά με τη βέλ- τιστηβέλτιστη λύση του. Για παράδειγμα, πόσο δύσκολο είναι στην επίλυσή του ή ποιος είναι ο ταχύτερος αλγόριθμος επίλυσής του (πολυπ- λοκότηταπολυπλοκότητα χρόνου). Η *πολυπλοκότητα αλγορίθμων* αποτελεί ένα σημαντικό τομέα της Επιστήμης των Υπολογιστών και μελετά πό- σοπόσο αποδοτικά μπορεί να λυθεί ένα συγκεκριμένο πρόβλημαπρό- βλημα. Το μοντέλο της *μηχανής Turing* (θυμίζουμε ότι ο Alan Turing είναι από τους θεμελιωτές της Επιστήμης των Υπολογιστών) αξιοποιεί- ταιαξιοποιείται για τη διερεύνηση της δυνατότητας επίλυσης ενός προβλήμα- τοςπροβλήματος με αλγοριθμικό τρόπο.

12.1 ~~Η~~Η έννοια της Υπολογιστικής Σκέψης

Η Wing, που έκανε δημοφιλή την ιδέα της Υπολογιστικής Σκέψης στο χώρο της εκπαίδευσης, επεξηγεί ότι η υπολογιστική σκέψη περιγράφει τη νοητική δραστηριότητα~~δραστηριό τητα~~ μορφοποίησης ενός προβ-λήματος~~προβλήματος~~, ώστε να επιδέχεται υπολογιστικές λύσεις. Οι λύσεις αυ- τές~~αυτές~~ μπορεί να υλοποιηθούν είτε από τον άνθρωπο, είτε από τη μηχανή ή από το συνδυασμό ανθρώπων και μηχανών. Η ΥΣ είναι μια προσέγγιση επίλυσης~~επίλυ σης~~ προβλήματος που ενισχύει το συνδυ- ασμό~~συνδυασμό~~ των ψηφιακών τεχνολογιών με τις ανθρώπινες ιδέες. Δεν αντικαθιστά την έμφαση στη δημιουργία, στη λογική και στην κριτι- κή~~κριτική~~ σκέψη, αλλά αντίθετα, τονίζει αυτές τις δεξιότητες μέσα από την ανάδειξη τρόπων για την οργάνωση ενός προβλήματος, ώστε να μπορεί να υποστηριχθεί η επίλυσή τους από υπολογιστές.

Ας σκεφτούμε τα ακόλουθα **παραδείγματα**.

- Εύρεση ενός λήμματος σε μια πολυσέλιδη εγκυκλοπαίδεια. Επίλυση του προβλήματος με εφαρμογή των αλγορίθμων της σειριακής και της Δυαδικής αναζήτησης. Αναγνώριση και υλοποίηση της αποδοτικότερης και αποτελεσματικότερης~~αποτελε σματικότερης~~ λύσης. Γενίκευσή της για επίλυση παρόμοιων προβλημάτων αναζήτησης.
- Εύρεση της πιο σύντομης διαδρομής για τη μετάβαση από το σπίτι στο σχολείο~~σχο λείο~~. Μορφοποίηση του προβλήματος, αφα- ίρεση~~αφαίρεση~~ περιττών στοιχείων και επιλογή~~επι- λογή~~ της απαραίτητης πλη-ροφορίας~~πληροφορίας~~. Επισήμανση των βασικών κόμβων (σημείωση~~ση μείων~~) των εναλλακτικών διαδρομών σε ένα χάρτη της πόλης. Χάραξη της κάθε διαδρομής ενώνοντας τους βασικούς κόμβους ως διαδικασία αφαίρεσης και μοντελοποίησης του προβλήματος. Αναπαράσταση της κάθε διαδρομής με χρή-σηχρήση κατάλληλης δομής δεδομένων.
- Ανάλυση της διαδικασίας της ουράς μπροστά σε ένα ταμείο supermarket για το τρόπο εξυπηρέτησης. Αναγνώριση των βασικών λειτουργιών της ώθησης και απώθησης σε μία ου-

ράθυρά. Μοντελοποίηση των διαδικασιών και περιγραφή τους με αλγοριθμικό τρόπο.

- Ο προγραμματισμός ενός εκπαιδευτικού ρομπότ για την εύρεση της εξόδου σε ένα λαβύρινθο. Αναγνώριση, ανάλυση και υλοποίηση πιθανών λύσεων του προβλήματος. Ανάλυση και υλοποίηση του αλγορίθμου (κανόνας κατεύθυνσης, όπωςκατεύθυνσης, όπως δεξιού ή αριστερού χεριού).

- Σχεδίαση ενός σπιτιού με γεωμετρικά σχήματα πάνω σε χαρτί. Εννοιολογική αφαίρεση και επιλογή των βασικών σχημάτων από τα οποία αποτελείται το σχέδιο του σπιτιού (τετράγωνο- ισόπλευρο τρίγωνο). Αναγνώριση των βασικώνβασι κών ιδιοτήτων κάθε σχήματος που απαιτούνται για το σχεδιασμό τους. ΕντοπισμόςΕντο- πισμός των διαδικασιών που επαναλαμβάνονται κάθε φορά για το σχεδιασμό του κάθε σχήματος. Αυτομα- τοποίησηΑυτοματοποίηση της λύσης σχεδιασμού ενός σπιτιού από υπολο- γιστή υπολογιστή με αλγοριθμικό τρόπο. Υλοποίηση της λύσης σε γλώσσα προγραμματισμούπρο-

~~γραμματισμού~~, για παράδειγμα σε γλώσσα Py- thon~~Python~~ με χρήση της βιβλιοθήκης turtle.

Η υπολογιστική σκέψη αποτελεί μια ικανότητα των ανθρώπων να είναι ικανοί να λύνουν προβλήματα και όχι σε μια προσπάθεια προσαρμογής τους, ώστε να σκέφτονται~~σκέ- φτονται~~ σαν υπολογιστές. Η υπο- λογιστική~~υπολογιστική~~ σκέψη συνδυάζει την επιστήμη, την τεχνολογία~~τε- χνολογία~~ και την κοινωνία, που αλληλεπιδρούν. Η επιστημονική έρευνα τροφοδοτεί την τεχνολογική εξέλιξη, η οποία τροφοδοτεί νέες κοινωνικές ε- φαρμογές~~εφαρμογές~~ κ.ο.κ.

Όλα τα επιλύσιμα προβλήματα μπορούν να λυθούν με πολύ πε- ρισσότερους τρόπους~~περισσότερους τρό- πους~~ από αυτούς που θεωρούμε στην αρχή ως τους μόνους δυνατούς. Συχνά, αρκεί~~αρ- κεί~~ να επανεξετάσουμε το ερώ- τημα~~ερώτημα~~, αναδιατυπώνοντάς το και να σκεφτούμε ελεύθερα~~ελεύ- θερα~~. Αυτό συμβαίνει διότι, πολλές φορές, όταν αντιμετωπίζουμε ένα πρόβ- λημα~~πρόβλημα~~ ψάχνουμε τη λύση, που μας υποδεικνύει η αρχική του δια- τύπωση.~~διατύπωση.~~

12.2 Χαρακτηριστικά της Υπολογιστικής Σκέψης

Η υπολογιστική σκέψη περιλαμβάνει, χωρίς να περιορίζεται, τα παρακάτω βασικά χαρακτηριστικά:

- Μορφοποίηση προβλημάτων με τρόπο που επιτρέπει τη χρήση υπολογιστή και άλλων εργαλείων για την επίλυσή τους.
- Εννοιολογική αφαίρεση και επιλογή της απαραίτητης πληροφορίας για την αντιμετώπιση της πολυπλοκότητας των σύνθετων προβλημάτων.
- Διάσπαση ενός προβλήματος σε απλούστερα υποπροβλήματα.
- Λογική οργάνωση και ανάλυση δεδομένων.

- Αναπαράσταση δεδομένων μέσα από μοντέλα και προσομοιώσεις.
- Αναγνώριση, ανάλυση και υλοποίηση πιθανών λύσεων με σκοπό την επίτευξη ~~επίτευ- ξη~~ των αποτελεσματικότερων και αποδοτικότερων συνδυασμών, βημάτων και πόρων.
- Αυτοματοποίηση των λύσεων μέσα από την περιγραφή τους με αλγοριθμικό τρόπο.
- Αξιολόγηση του αλγορίθμου και της λύσης που δόθηκε.
- Γενίκευση και μεταφορά της διαδικασίας επίλυσης του συγκεκριμένου προβλήματος~~προ- βλήματος~~ σε μια ευρύτερη ποικιλία από προβλήματα.
- Κατανόηση της ανθρώπινης συμπεριφοράς κατά την αντιμετώπιση ενός προβλήματος~~προ- βλήματος~~.

Για την ανάλυση του προβλήματος και τον εντοπισμό των δομικών ή κύριων στοιχείων~~στοι- χείων~~ που το αποτελούν, είναι απαραίτητο το πρόβ- λημα~~πρόβλημα~~ να μορφοποιηθεί, ώστε να είναι απαραίτητο να απαλλαγεί από περιττές λεπτομέρειες που αυξάνουν, χωρίς λόγο, τον όγκο των στοιχείων που πρέπει να διαχειριστούμε Η λειτουργία αυτή είναι
γνωστή ως *αφαίρεση* (abstraction) και αποτελεί μια νοητική ικανότητα εντοπισμού

των βασικών χαρακτηριστικών ενός αντικεί- μενουαντικείμενου ή γενικότερα μιας κατάστασης. Η αξία της αφαίρεσης έγκει- ταιέγκειται στη δυνατότητα κριτικής επεξεργασίας δεδομένων και στην ανακάλυψη σχέσεων μεταξύ αντικειμένων ή καταστάσεων. Η αφα- ιρετική ικανότηταΗ αφαιρετική ικα- νότητα είναι απαραίτητη για τη σωστή ανάλυση ενός προβλήματος σε απλούστερα υποπροβλήματα και τη σαφή δια- τύπωσηδιατύπωση της δομής του.

Η υπολογιστική σκέψη περιέχει τη χρήση της αφαίρεσης, τη διάσ- πασηδιάσπαση ενός προβλήματοςπρο- βλήματος σε πιο απλά, το χρονοπρογραμματισμό ενός έργου, τη χρήση μεγάλου όγκου δεδομένων (big data), όπως είναι για παράδειγμα η διαχείριση petabytes δεδομένων στο Cern με χρήση της Python. Χαρακτηριστικό της Υπολογιστικής ΣκέψηςΣκέ- ψης είναι η σύλληψη εννοιών, η οποία απαιτεί σκέψη σε πολλαπλά επίπεδα αφαίρεσηςαφαί- ρεσης. Άλλο ένα χαρακτηριστικό της, είναι ότι συν- δυάζεισυνδυάζει τη μαθηματική σκέψη με τη σκέψη του μηχανικού. Αυτό, διότι η Επιστήμη των Υπολογιστών στηρίζεται στα Μαθηματικά, αλλά παράλληλα ελέγχει συστήματα τα οποία αλληλεπιδρούν με τον πραγματικό κόσμο, με αποτέλεσμα να χρειάζεται συνδυασμός μηχανικής και μαθηματικήςμαθη- ματικής σκέψης.

Παραδείγματα

Κατά τον υπολογισμό του χρόνου μετάβασης ενός οχήματος από μια τοποθεσία σε μια άλλη, το χρώμα του οχήματος δεν αποτελεί βασικό χαρακτηριστικό. Αντίθετα, η ταχύτητα που μπορεί να α- ναπτύξειαναπτύξει το όχημα και οι εναλλακτικές διαδρομές που μπορεί να ακολουθήσει, είναι βασικά στοιχεία. Η ικανότητα της αφαίρεσης μας επιτρέπειεπι- τρέπει να θεωρήσουμε ότι ένα όχημα της Φόρμουλα 1 και ένα τετρακίνητο τζιπ είναι και τα δύο αυτοκίνητα, παρόλο που έχο- υνέχουν τελείως διαφορετικά χαρακτηριστικά.

Στα Μαθηματικά, όταν λύνουμε ένα πρόβλημα, η απόδοση του αγνώστου στο χαρακτήραχαρα- κτήρα X (έστω X), είναι μια αφαιρετική διαδι- κασίαδιαδικασία.

12.3 Υπολογιστική σκέψη και επίλυση προβλημάτων

Η επίλυση ενός προβλήματος με τον υπολογιστή περιλαμβάνει τα ακόλουθα σημαντικάση- μαντικά στάδια:

- Τον ακριβή προσδιορισμό του προβλήματος.
- Την περιγραφή της λύσης του με την ανάπτυξη του αντίστοιχου αλγορίθμου.
- Τη διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον υπολογιστή για την υλοποίηση και αυτοματοποίηση της~~λύσης.~~ λύσης.

Μέσα από τη διαδικασία επίλυσης ενός προβλήματος, καλλιεργεί- ταικαλλιεργείται η αυστηρότητα και η σαφήνεια της έκφρασης και της διατύπω- σηςδιατύπωσης, η απόκτηση δεξιοτήτων αλγοριθμικήςαλγο- ριθμικής προσέγγισης, η ανίχνε- υσηανίχνευση και η δυνατότητα διάκρισης των μερών ενός προβλήματος και η

ανάπτυξη ικανοτήτων αναζήτησης και εύρεσης εναλλακτικών λύσεων.

~~*Κεφ.12: Εισαγωγή στην Υπολογιστική Σκέψη*~~

Πολλά από τα ανθρώπινα τεχνουργήματα, φυσικά ή τεχνητά, στηρίζονται~~στηρίζονται~~ στην ιδέα της επαναχρησιμοποίησης–συνδυασμού μικρό- τερων~~μικρότερων~~ δομικών μονάδων, με αποτέλεσμα~~απο τέλεσμα~~ την απλότητα και την ποικιλία αντικειμένων και χαρακτηριστικών. Στην Python η τεχνική αυτή υλοποιείται με τις συναρτήσεις, δίνοντας καλύτερο έλεγχο και υψηλό επίπεδο αφαίρεσης. Θυμίζουμε ότι οι συναρτήσεις είναι επαναχρησιμοποιήσιμα~~επαναχρησιμοποι- ήσιμα~~ τμήματα προγραμμάτων, που τα χρησι-

μοποιούμεχρησιμοποιούμε για να μην επαναλαμβάνουμεεπαναλαμβά- νουμε κώδικα μέσα στα προγ- ράμματαπρογράμματα μας, εξασφαλίζοντας μέσα από διαδικασίες αφαίρεσης την αυτοτέλειά τους, με κατάλληλη χρήση παραμέτρων.

Δραστηριότητες - Ερωτήσεις - Ασκήσεις

12.4 Δραστηριότητες κεφαλαίου

Δραστηριότητα 1. Διαχείριση προβλημάτων

Για τα τρία διαφορετικά προβλήματα, που παρουσιάζονται στη συνέχεια, να συμπληρωθείσυ- μπληρωθεί η ακόλουθη πληροφορία:

Κατανόηση του προβλήματος (χώρος του προβλήματος):

- Ποια δεδομένα είναι γνωστά;
- Τι δεν είναι γνωστό;
- Ποιο είναι το ζητούμενο;
- Ποιες είναι οι συνθήκες;
- Ποιο είναι το πλάνο εργασίας για την επίλυση του προβλήματος;
- Να σχεδιαστεί η λύση.

- ~~Ποια δεδομένα είναι γνωστά;~~
- ~~Τι δεν είναι γνωστό;~~
- ~~Ποιο είναι το ζητούμενο;~~
- ~~Ποιες είναι οι συνθήκες;~~
- ~~Ποιο είναι το πλάνο εργασίας για την επίλυση του προβλήματος;~~
- ~~Να σχεδιαστεί η λύση.~~

Υλοποίηση της λύσης: Χρησιμοποιώντας το πλάνο εργασίας, να παρουσιαστεί η όλη εργασία και η λύση.

Να γίνει ανακεφαλαίωση και συζήτηση της λύσης.

Πρόβλημα 1. Εύρεση των κάλπικων νομισμάτων με μια ζύγιση

Υπάρχουν δέκα σακιά που περιέχουν 100 νομίσματα το καθένα. Το κάθε νόμισμα ζυγίζει 10 γραμμάρια. Το ένα από τα δέκα σακιά έχει μέσα μόνο κάλπικα νομίσματα, τα οποία ζυγίζουν εννέα (9) γραμμάρια το καθένα. Πώς μπορούμε με μία μόνο ζύγισηζύγι- ση σε μία ηλεκτρονική ζυγαριά ακριβείας, να βρούμε ποιο σακί περιέχει τα κάλπικα νομίσματα;

Πρόβλημα 2. Το πρόβλημα της Χειραψίας

Ας υποθέσουμε ότι είκοσι άνθρωποι βρίσκονται μαζί με ένα μαθη- τήμαθητή σε ένα δωμάτιο και πρέπει ο μαθητής να ανταλλάξει χειραψία με κάθε έναν από αυτούς. Με πόσους ανθρώπους τελικά θα αν- ταλλάξειανταλλάξει χειραψία; Εάν υπάρχουν N (N>0) άνθρωποι μαζί με το μαθητή στο δωμάτιο, με πόσους τελικά θα έρθει αυτός σε επαφή;

Πρόβλημα 3.
Βελτιστοποίηση

Να μελετηθεί το πρόβλημα που θα υπολογίζει τα ρέστα που πρέπει~~πρέπει~~ να δώσει ένα αυτόματο μηχάνημα έκδοσης εισιτηρίων. Τα εισι- τήρια~~εισιτήρια~~ κοστίζουν 0.95€. Ο αλγόριθμος~~αλγόριθ μος~~ θα δέχεται το ποσό που πληρώνει ο πελάτης και θα επιστρέφει τον αριθμό των κερμάτων με αξία 2€, 1€, 0.50€, 0.20€, 0.10€. Ο αλγόριθμος πρέπει να λει- τουργεί~~λειτουργεί~~ με τέτοιο τρόπο, ώστε τα ρέστα να δίνονται με το μικρότε- ρο~~μικρότερο~~ δυνατό (βελτιστοποίηση~~βελτιστοποίη ση~~) αριθμό νομισμάτων.

Δραστηριότητα
2. Γενίκευση

Ο σημερινός αριθμός αυτοκινήτων που κυκλοφορούν σε μια πόλη είναι 90000. Αν ο αριθμός αυτός αυξάνεται με ετήσιο ρυθμό 5%, να γραφεί σενάριο σε Python που υπολογίζει σε πόσα χρόνια ο αριθμός των αυτοκινήτων θα ξεπεράσει τις 160000. Να εμφανίζει πόσα θα είναι τότε τα αυτοκίνητα. Γενικεύστε το πρόγραμμα~~πρό- γραμμα~~ για Arith_Aytok αυτοκίνητα, που αυξάνονται με ετήσιο ρυθμό (Rythmos) και ξεπερνούν ένα δοσμένο όριο (Orio). Τέλος, να οπ- τικοποιηθεί~~οπτικοποιηθεί~~ ο κώδικας στο http://www.pythontutor.com/

Δραστηριότητα
3. Γενίκευση

Μελετήστε το πρόβλημα των πύργων του Ανόι βλέποντας την οπ- τικοποίησή~~ οπτικοποίησή~~ του από το Εθνικό Αποθετήριο Εκπαιδευτικού Περιε- χομένου~~Περιεχομένου~~ «Φωτόδεντρο»: (http://photodentro.edu.gr/aggregator/lo/photodentro-lor-8521-1010).
).

Το πρόβλημα να επιλυθεί αρχικά για τρεις δίσκους, σε φυσική γλώσσα. Να αξιοποιηθεί~~αξιοποι- ηθεί~~ η λύση αυτή για να λυθεί το πρόβλημα με τέσσερις δίσκους. Τέλος, να γενικευθεί~~γενικευ θεί~~ η λύση για οποιοδήποτε πλήθος δίσκων.

12.5 Ερωτήσεις - Ασκήσεις

- Να αναφερθούν πέντε βασικά χαρακτηριστικά της υπολογιστικής σκέψης.
- Να αναφερθούν τρία καθημερινά παραδείγματα υπολογιστικής σκέψης.
- Να περιγραφεί σύντομα η λειτουργία της Αφαίρεσης για την κατανόηση σύνθετωνσύν- θετων προβλημάτων.

- Να περιγραφεί σύντομα η λειτουργία της Γενίκευσης της διαδικασίας επίλυσης προβλήματος.

Σύνοψη

Στο κεφάλαιο αυτό ασχοληθήκαμε με τα χαρακτηριστικά της υπο-λογιστικής~~υπολογιστικής~~ σκέψης στην επίλυση προβλημάτων. Δόθηκε, με τη βοήθεια παραδειγμάτων, έμφαση στη σημασία της υπολογιστικής σκέψης στην επίλυση προβλημάτων της καθημερινής ζωής.

~~Αναφορές - Βιβλιογραφία~~

(Η τελευταία προσπέλαση στις ηλεκτρονικές πηγές έγινε τον Απρίλιο του 2016)

Alice, (2016), 3D περιβάλλον κινούμενων γραφικών, ~~γραφικ~~ών, http://www.alice.org/ Ben Shneiderman and Catherine Plaisant. Designing the User In- terface ~~Interface~~. Pearson Education, Boston, 4th edition, 2005

Berkeley, (2015), http://python.berkeley.edu/..

Donald A. Norman. The Design of Everyday Things. Double- day ~~Doubleday~~/Currency, New York, NY, 1990. First published as *The Psychology of Everyday Things*.

Downey, A. (2012) «Think Python, How to think like a computer scientist», O' Reilly. http://www.greenteapress.com/thinkPython/..

Ferg, S. (2006) "Event-Driven Programming: Introduction, Tutorial, History". Διαθέσιμο στη θέση: http://Tutorial_EventDrivenProgramming.sourceforge.net

Jeannette M. Wing, "Computational Thinking," Communications of the ACM, CACM vol. 49, no. 3, March 2006, pp. 33-35.

Levitin, Anany (2007), The Design & Analysis of Algorithms. Pear- son Education. ~~Pearson Education.~~

Miller, N., M., and Ranum, L., D. (2011). Problem Solving with Al- gorithms ~~Algorithms~~ and Data Structures using Python. Franklin, Beedle & Associates; 2nd edition.

Object-Oriented Systems Analysis and Design Using UML (2nd edition). S. Bennett, S. McRobb, R. Farmer, McGraw Hil, 2002.

Python Tutorial http://python-tutorial-~~-greek.readthedocs.org/en/latest/oop_general.html~~

Shipman, J, W. (2013) "Tkinter 8.5 reference: a GUI for Python"

Swaroop, C., H. (2013) "A byte of Python", http://www.swaroopch.com/notes/Python/ & ~~&~~ και στα Ελληνικά http://ubuntu-gr.org/ ~~–~~Ελληνική κοινότητα του Ubuntu

Swaroop, C., H. (2013) "A byte of Python", Διαδικτυακή έκδοση ηλεκτρονικού βιβλίου http://www.swaroopch.com/notes/Python/ με άδεια Creative Commons Attributions –ShareAlike 4.0 Inter- national ~~International~~ License. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από την Ελληνική κοινότητα του Ubuntu. http://ubuntu-gr.org/ και είναι διαθέσιμο στο Διαδίκτυο.

Thomas T. Cormen, Charles E. Leiserson, and Ronald L. Rivest. 1990. Introduction to Algorithms. MIT Press, Cambridge, MA, USA.

Tutorial από το δικτυακό τόπο της γλώσσας Python για την γνωρι- μία με τη γλώσσα Python – Δομή Ελέγχου και Δομές Επανά- ληψης Επανάληψης https://docs.Python.org/2/tutorial/controlflow.htm

Αβούρης Ν. (2000) «Εισαγωγή στην επικοινωνία ανθρώπου υπο- λογιστή», εκδ. Δίαυλος.

Αβούρης Ν. Κουκιάς Μ., Παλιουράς Β, Σγάρμπας Κ. (2013) «Ει- σαγωγή στους υπολογιστές με τη γλώσσα Python», Εκδόσεις Πανεπιστημίου Πατρών, Πάτρα.

Ανάπτυξη Γραφικής Διεπαφής με τη βιβλιοθήκη Tkinter της Python

ΑΠΣ_Τομέα_Πληρ_ΕΠΑΛ (2015). Αναλυτικά Προγράμματα Σπου- δών του μαθήματος Γενικής Παιδείας «Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ» της Β΄ και Γ΄ τάξης Ημερήσιων και Γ΄ και Δ΄ τάξης Εσπερινών ΕΠΑ.Λ. και των μαθημάτων ειδικοτή- των του Τομέα Πληροφορικής της Ομάδας Προσανατολισμού Τεχνολογικών Εφαρμογών των τάξεων Β΄ και Γ Ημερήσιων και Β΄, Γ και Δ΄ Εσπερινών ΕΠΑ.Λ., ΦΕΚ 2010. 16/9/2015.

Αρχές *Προγραμματισμού Υπολογιστών*, (2015). Διδακτικό υλικό των Αράπογλου Α., Βραχνός Ε., Κανίδης Ε., Μακρυγιάννης Π., Μπελεσιώτης Β., Τζήμας Δ, ISBN 978-960-06-5141-6

Βιβλιοθήκη οπτικοποίησης της Google με Python

ΒΙΚΙΠΑΙΔΕΙΑ Αντικειμενοστρεφής Προγραμματισμός https://el.wikipedia.org/wiki/Αντικειμενοστρεφής_προγραμματισμός

Βικιπαίδεια, υλικό και χρήσιμοι σύνδεσμοι, με έλεγχο ποιότητας, http://el.wikipedia.org/wiki/Python

Δικτυακός κόμβος με παράδειγμα χρήσης της sqlite στην Python

Δικτυακός κόμβος υποστήριξης με πλούσιο υλικό πολυμέσων για τη διδασκαλία της γλώσσας Python, http://www.Pythonschool.net/

Δικτυακός κόμβος υποστήριξης της γλώσσας Python, https://www.Python.org/

Δικτυακός κόμβος υποστήριξης της γλώσσας Py- t- Python, http://www.pythonschool.net και ειδικότερα των Βάσεων Δεδομένων http://www.pythonschool.net/category/databases.html

Δικτυακός τόπος για τη διδασκαλία της επιστήμης της Πληροφορι- κής χωρίς

Πληροφορικής χωρίς υπολογιστές Computer Science Unplugged: http://www.csunplugged.org

Δικτυακός τόπος εκπαίδευσης στη γλώσσα Python: https://www.codecademy.com/learn/python

Δικτυακός τόπος με πλούσιο σχετικό υλικό http://www.python-course.eu/course.php_ http://www.python- course.eu/course.php

Δικτυακός τόπος με πλούσιο σχετικό υλικό http://www.tutorialspoint.com/python/python_files_io.htm

Δικτυακός τόπος με σχετικό υλικό υποστήριξης http://www.tutorialspoint.com/python/python_modules.htm

Δικτυακός τόπος με ψηφιακό υλικό για την εισαγωγή στην αλγο- ριθμική σκέψη: αλγοριθμική σκέψη: http://www.teaching- materials.org/algorithms/

Δικτυακός τόπος οπτικοποίησης κώδικα: http://www.pythontutor.com/

Ενδεικτικές Βιβλιοθήκες προγραμματισμού για οπτικοποίηση δε- δομένωνδεδομένων

Εργαλείο on-line για τη κατασκευή διαγραμμάτων ροής https://www_ διαγραμμάτων ροής.gliffy.com

Ευρετήριο διεπαφών προγραμματισμού εφαρμογών της Python,

Λεβεντέας, Δ., (2010), «Taspython. Εκμάθηση Python Βήμα, Βή- μα.

Βήμα. Οδηγός Python Μέσω Παραδειγμάτων», Ομάδα TasPython.

Οδηγός για τον Εκπαιδευτικό για το Πρόγραμμα Σπουδών του Μαθήματος «Πληροφορική» Γ' Τάξης Γενικού Λυκείου, στο πλαίσιο του έργου «ΝΕΟ ΣΧΟΛΕΙΟ (Σχολείο 21ου αιώνα) – Νέο Πρόγραμμα Σπουδών», Υποέργο 9: «Εκπόνηση Προγ- ραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχ- νικών Λυκείων», Υ.ΠΟ.ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτι- κής (Ι.Ε.Π), Ιανουάριος 2015.
9: «Εκπόνηση Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ.ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π), Ιανουάριος 2015.

Οδηγός εκπαιδευτικού - Κεφάλαιο 1, ΈΡΓΟ «Νέο Σχολείο (Σχολείο 21ου αιώνα) – – Νέο Πρόγραμμα Σπουδών» (κωδ. ΟΠΣ: 295450/Υποέργο 9/ Δράση: Εκπόνηση των Προγραμμάτων Σπουδών και Οδηγού Εκπαιδευτικού Μαθήματος «Πληροφο- ρική «Πληροφορική» Γ΄ τάξης Γενικού Λυκείου – ΦΕΚ 189/23-1-2015).

ΠΣ_Γ_ΓΕΛ (2015). Πρόγραμμα Σπουδών και Οδηγός Εκπαιδευ- τικούΕκπαιδευτικού Μαθήματος

«Πληροφορική» Γ΄ τάξης Γενικού Λυκείου (ΦΕΚ189/23-1-2015) (ΠΣ_Γ_ΓΕΛ, 2015). *Έργο ΙΕΠ «Νέο Σχο- λείο~~Σχολείο~~ (Σχολείο 21ου αιώνα) – Νέο Πρόγραμμα Σπουδών» (κωδ. ΟΠΣ: 295450/Οριζόντια Πράξη στις 8 Π.Σ., 3 Π.Στ. Εξ., 2 Π.Στ. Ε- ισ~~Εισ~~./Υποέργο 9 Δράση: Εκπόνησης των Προγραμμάτων Σπουδών και Οδηγού Εκπαιδευτικού Μαθήματος «Πληροφορική» Γ΄ τάξης Γενικού*

Λυκείου. Μέλη (αξιολογητές, συντονιστής, εμπειρογνώμονες~~εμπειρογνώμονες~~): (εδώ αλφ/κά- αναλυτικά στο κάθε υλικό/παραδοτέα): Αράπογ- λου~~Αράπογλου~~ Αριστείδης, εκπαιδευτικός ΠΕ19, υπ. ΚΕ.ΠΛΗ.ΝΕ.Τ - Βαρζά- κας~~Βαρζάκας~~ Παναγιώτης, μέλος Ε.Π. -Α.Τ.Ε.Ι. -Βραχνός Ευριπίδης, εκπαι- δευτικός~~εκπαιδευτικός~~ ΠΕ19 - Κανίδης Ευάγγελος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής - Λέκκα Δήμητρα, εκπαιδευτικός ΠΕ19 - Μαραγκός Κωνσταντίνος, εκπαιδευτικός ΠΕ19 - Μαυρίδης Ιωάννης, μέλος ΔΕΠ - Μπελεσιώτης Βασίλειος, Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής - Παπαδάκης Σπυρίδων, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής - Τζήμας Δημήτριος, εκπαιδευτικός ΠΕ19.~~Πληροφορικής - Παπαδάκης Σπυρίδων, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής - Τζήμας Δημήτριος, εκπαιδευτικός ΠΕ19.~~

Σημειώσεις επικοινωνίας Ανθρώπου – Μηχανής (2005), Καθηγη- τής ~~Καθηγητής~~ Γιάννης Ιωαννίδης, Δρ. Γιώργος Λέπουρας, διαθέσιμο στο http://cgi.di.uoa.gr/~ys08/EAM_simeiwseis.pdf(Πρόσβαση Ιού- νιος 2015) ~~(Πρόσβαση Ιούνιος 2015)~~

Συγγραφείς_Προγραμματισμός. (2016). Ομάδες συγγραφής δι- δακτικού ~~διδακτικού~~ υλικού μαθήματος Προγραμματισμός, Γ'Γ΄ ΕΠΑ.Λ., Το- μέας~~Τομέας~~ Πληροφορικής, ΑΔΑ ΩΓΣΝΟΞΛΔ-Τ6Ψ και 6ΥΓΖΟΞΛΔ-ΛΨΥ, διαθέσιμες στο https://et.diavgeia.gov. gr/

Το διαδικτυακό μάθημα http://www.highschoolelearning.com/freelessons/oop/ObjectOrientedProgram-~~-~~ming.htm

Υπολογιστική σκέψη και την επίλυση προβλήματος - Εκπαιδευτι- κά~~Εκπαιδευτικά~~ παιχνίδια: http://games.thinkingmyself.com/

Ψηφιακό Σχολείο - Αποθετήριο Φωτόδεντρο. Περιέχει μεταξύ άλ- λων~~άλλων~~ μαθησιακά αντικείμενα για την προσέγγιση της αλγοριθμι- κής ~~αλγοριθμικής~~ σκέψης “http://photodentro.“ ~~- edu.gr/lor/subject-~~search?locale=el”

Ωρολόγιο Πρόγραμμα (2015). των μαθημάτων Γενικής Παιδείας της Α΄ τάξης Εσπερινών ΕΠΑ.Λ. και των Β΄, Γ΄ τάξεων Ημερη- σίων~~Ημερησίων~~ και Α΄, Β΄ και Γ΄ τάξεων Εσπερινών ΕΠΑ.Λ. ανά Ειδικότη- τα~~Ειδικότητα~~ Τομέα Ομάδας Προσανατολισμού, ΦΕΚ1053/2015.

Χρήσιμοι Διαδικτυακοί Τόποι

http://blog.yhathq.com/posts/ggplot-for-Python.html

http://matplotlib.org/users/pyplot_tutorial.html

http://sebastianraschka.com/Articles/2014_sqlite_in_python_tutorial.html

http://stanford.edu/~mwaskom/software/seaborn/

http://www.blog.pythonlibrary.org/2012/07/18/python-a-simple-step-by-step-sqlite-tutorial/

http://www.greenteapress.com/thinkPython/.

http://www.greenteapress.com/thinkPython/.

http://www.Python-course.eu/Python_tkinter.php

http://www.swaroopch.com/notes/Python/

http://www.swaroopch.com/notes/Python/. .

https://code.google.com/p/google-visualization-Python/

https://pypi.Python.org/pypi

https://pypi.Python.org/pypi/ggplot

https://wiki.Python.org/moin/TkInter

~~Ευρετήριο όρων~~

Ευρετήριο όρων

Κωδικός Βιβλίου: 24-0576
ISBN 978-960-06-5309-0

(01) 000000 0 24 0576 6