ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

Αρχές Προγραμματισμού Υπολογιστών

Β΄ ΕΠΑ.Λ.

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ «ΔΙΟΦΑΝΤΟΣ»

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

Αράπογλου Α., Βραχνός Ε., Κανίδης Ε., Μακρυγιάννης Π.,
Μπελεσιώτης Β., Τζήμας Δ.

Αρχές Προγραμματισμού Υπολογιστών

Σημειώσεις Μαθητή

Β΄ ΕΠΑ.Λ.

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ «ΔΙΟΦΑΝΤΟΣ»

ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

Πρόεδρος: **Κουζέλης Γεράσιμος**, Καθηγητής ΕΚΠΑ
Επιστημονικά Υπεύθυνος: **Τσαπέλας Θεοδόσιος**, Σύμβουλος Β΄ Πληροφορικής Ι.Ε.Π.

ΣΥΓΓΡΑΦΙΚΗ ΟΜΑΔΑ
Αράπογλου Αριστείδης, Εκπαιδευτικός Πληροφορικής
Βραχνός Ευριπίδης, Εκπαιδευτικός Πληροφορικής
Κανίδης Ευάγγελος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής
Μακρυγιάννης Παναγιώτης, Εκπαιδευτικός Πληροφορικής
Μπελεσιώτης Βασίλειος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής
Τζήμας Δημήτριος, Εκπαιδευτικός Πληροφορικής

ΕΠΙΜΕΛΕΙΑ-ΣΥΝΤΟΝΙΣΜΟΣ ΟΜΑΔΑΣ
Κανίδης Ευάγγελος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής
Μπελεσιώτης Βασίλειος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής

ΦΙΛΟΛΟΓΙΚΗ ΕΠΙΜΕΛΕΙΑ
Ευφροσύνη Δεληγιάννη, Σχολική Σύμβουλος ΠΕ02

ΕΠΙΤΡΟΠΗ ΚΡΙΣΗΣ
Βογιατζής Ιωάννης, Επίκουρος Καθηγητής Τ.Ε.Ι. Αθηνών
Μελετίου Γεώργιος, Μηχανικός Πληροφορικής MSc
Πράπας Λεωνίδας, Εκπαιδευτικός Πληροφορικής

Έκδοση 2η, 2017

Εισαγωγικό σημείωμα

Το παρόν διδακτικό υλικό σχετίζεται με το Α.Π.Σ του Ιουλίου 2015 για τα μαθήματα, Αρχές Προγραμματισμού Υπολογιστών της Β' και Γ' τάξης των Ημερησίων και Εσπερινών ΕΠΑ.Λ. και Προγραμματισμός Υπολογιστών της Γ' και Δ' τάξης Ημερησίων και Εσπερινών ΕΠΑ.Λ. αντίστοιχα, για τις ειδικότητες του Τομέα Πληροφορικής και ειδικότερα για το μάθημα Αρχές Προγραμματισμού Υπολογιστών.

Οι συγγραφείς του ευελπιστούν στην καλύτερη υποστήριξη των μαθημάτων αυτών για την επίτευξη των μαθησιακών στόχων, με δεδομένες τις συνθήκες ανάπτυξής τους.

Αν παρατηρήσουμε τον κόσμο γύρω μας, θα διαπιστώσουμε ότι κατακλυζόμαστε από ποικίλες εφαρμογές υπολογιστών, που χρησιμοποιούμε καθημερινά μέσα από διαφορετικές συσκευές. Παράλληλα παρατηρούμε μια ραγδαία εξέλιξη στη δημιουργία ψηφιακού περιεχομένου, που μπορεί να μεταφέρεται σε ελάχιστο χρόνο από το ένα μέρος της γης στο άλλο εκμηδενίζοντας τις αποστάσεις. Συντελούνται ισχυρές μεταβολές στην οικονομία, στην κοινωνική ζωή, στο χώρο της εργασίας. Σε μια μεταβαλλόμενη οικονομία λόγω των τεχνολογικών εξελίξεων, είναι σημαντικό να αναπλαισιώσουμε το ρόλο μας ως προς τη δυνατότητα πρόσβασης στις Τεχνολογίες της Πληροφορικής και των Επικοινωνιών που την αξιοποιούν σε άλλους τομείς και να μεταβούμε από την απλή χρήση και την κατανάλωσή τους στο επίπεδο του προγραμματισμού, της δημιουργίας των δικών μας εφαρμογών και περιεχομένου, στην αξιοποίηση του υπολογιστή με δυνατότητα επίδρασης σε αυτόν.

Το διδακτικό υλικό που έχετε στα χέρια σας έχει ως στόχο να σας βοηθήσει να εμπλουτίσετε τις γνώσεις και να αποκτήσετε τις απαραίτητες δεξιότητες σε θέματα επίλυσης υπολογιστικών προβλημάτων με αλγοριθμικό τρόπο. Παράλληλα ευελπιστούμε να αποτελέσει το έναυσμα για να εισαχθείτε ολοκληρωμένα στον κόσμο του προγραμματισμού των υπολογιστών, να κατανοήσετε περαιτέρω βασικές αρχές του και να αποκτήσετε δεξιότητες στη δημιουργία δικών σας ολοκληρωμένων προγραμμάτων. Μέσα από τις ποικίλες δραστηριότητες που προτείνονται, έχετε την ευκαιρία να πειραματιστείτε με μια σύγχρονη γλώσσα προγραμματισμού, τη γλώσσα Python, να εμβαθύνετε σε αρχές του προγραμματισμού, να τροποποιήσετε έτοιμα προγράμματα και στη συνέχεια να κατασκευάσετε τα δικά σας.

Η υλοποίησή του, καθώς και η 2η έκδοση, έγιναν αμισθί από το σύνολο των εμπλεκομένων, βάσει των αποφάσεων του ΔΣ του ΙΕΠ: 5900/20-05-2015 (ΑΔΑ 7ΣΑΠΟΞΛΔ-Δ-ΣΠ) και 2268/5-4-2017 (ΑΔΑ ΩΞ37ΟΞΛΔ-ΞΘ0).

Στα κείμενα του παρόντος υλικού και για λόγους απλοποίησης και μη διάσπασης της προσοχής, χρησιμοποιείται το δεύτερο πληθυντικό πρόσωπο, καλύπτοντας και τα δύο γένη.

Περιεχόμενα

1

Από το πρόβλημα στην ανάπτυξη αλγόριθμου

1. Από το πρόβλημα στην ανάπτυξη αλγόριθμου

Στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- αναγνωρίζουμε ένα σύνθετο πρόβλημα και τα πλεονεκτήματα που προκύπτουν από την ανάλυσή του σε απλούστερα
- συνθέτουμε τη λύση ενός προβλήματος, επιλύοντας τα απλούστερα επιμέρους προβλήματα
- περιγράφουμε τη λύση ενός προβλήματος με μια πεπερασμένη σειρά αυστηρά καθορισμένων ενεργειών
- περιγράφουμε έναν αλγόριθμο με εναλλακτικούς τρόπους διαγραμματικής αναπαράστασης-παρουσίασης (ψευδοκώδικα, διάγραμμα ροής).

Λέξεις κλειδιά

Πρόβλημα, ανάλυση προβλήματος, αφαίρεση, επίλυση προβλήματος, αλγόριθμος, αναπαράσταση αλγόριθμου.

Εισαγωγή

Στην ενότητα αυτή προσεγγίζονται θέματα για τη δομή ενός προβλήματος και την πολυπλοκότητά του, για τη διαδικασία της αφαίρεσης κατά την απλοποίηση ενός προβλήματος, για την ανάλυση ενός προβλήματος σε απλούστερα υποπροβλήματα και τέλος την περιγραφή της λύσης του με αλγοριθμικό τρόπο, εκφρασμένη με ψευδοκώδικα ή διάγραμμα ροής.

Η σχεδίαση αλγορίθμων σχετίζεται με τη διαδικασία επίλυσης του προβλήματος, με την ανάλυσή του σε απλούστερα, τη σύνθεση των λύσεων των επιμέρους προβλημάτων και κυρίως με την κατασκευή και μορφοποίηση της λύσης του, ώστε να μπορεί να αναπαρασταθεί σε μορφή που μπορεί να υλοποιηθεί αποτελεσματικά από τον υπολογιστή.

Διδακτικές ενότητες

1.1 Εισαγωγή στη διαχείριση της πολυπλοκότητας ενός προβλήματος

Κάθε άνθρωπος στη ζωή του, σε οποιαδήποτε ηλικία και αν είναι, αντιμετωπίζει καθημερινά μικρά ή μεγάλα προβλήματα. Από την αρχή της σχολικής μας ζωής, έχουμε λύσει πολλά προβλήματα στα Μαθηματικά και στη Φυσική. Το ίδιο συμβαίνει σε όλους τους τομείς της επιστήμης και της τεχνολογίας.

Με τον όρο **Πρόβλημα** προσδιορίζεται μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι γνωστή, ούτε προφανής. Σε αρκετά προβλήματα η λύση είναι εύκολο να βρεθεί, αλλά υπάρχουν και προβλήματα που η λύση τους είναι δύσκολο να βρεθεί. Για παράδειγμα προβλήματα όπως: η ρύπανση του περιβάλλοντος, η αντιμετώπιση των ναρκωτικών, η θεραπεία ασθενειών, η ασφαλής πλοήγηση στο Διαδίκτυο είναι σύνθετα προβλήματα που δεν επιδέχονται μια εύκολη λύση.

Ως *σύνθετο* χαρακτηρίζεται ένα πρόβλημα που αποτελείται από πολλά μέρη και στη λύση του συμμετέχουν πολλοί παράγοντες που συχνά αλληλεπιδρούν μεταξύ τους. Ορισμένα από τα προβλήματα που αντιμετωπίζουμε μπορούν να επιλυθούν με την βοήθεια ενός υπολογιστή. Στην περίπτωση αυτή το πρόβλημα χαρακτηρίζεται **υπολογιστικό**. Στο κεφάλαιο αυτό θα ασχοληθούμε με υπολογιστικά προβλήματα.

Το πρώτο στάδιο για την επίλυση ενός προβλήματος είναι η **κατανόησή** του. Όταν το πρόβλημα είναι σύνθετο, η κατανόησή του προϋποθέτει την ανάλυση της δομής του. 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. Γενικά με τον όρο **Ανάλυση** εννοείται ο διαχωρισμός ενός συνόλου στα συστατικά του στοιχεία. Για την ανάλυση του προβλήματος και τον εντοπισμό των κύριων στοιχείων είναι απαραίτητο το πρόβλημα να μορφοποιηθεί, ώστε να απαλλαγεί από τις περιττές λεπτομέρειες που αυξάνουν τον όγκο των στοιχείων που πρέπει να διαχειριστούμε. Η λειτουργία αυτή ονομάζεται αφαίρεση.

Η αξία της αφαίρεσης

Αφαίρεση είναι η νοητική ικανότητα εντοπισμού των βασικών χαρακτηριστικών ενός αντικείμενου ή γενικότερα μιας κατάστασης. Η ικανότητα αυτή επιτρέπει την κριτική επεξεργασία δεδομένων και την ανακάλυψη σχέσεων μεταξύ αντικειμένων ή καταστάσεων.

Παραδείγματα

Στον υπολογισμό μιας διαδρομής ενός οχήματος, το χρώμα του οχήματος δεν είναι βασικό χαρακτηριστικό. Αντίθετα η ταχύτητα που μπορεί να αναπτύξει είναι βασικό χαρακτηριστικό.

Η ικανότητα της αφαίρεσης μάς επιτρέπει να θεωρήσουμε ότι ένα όχημα της Φόρμουλα 1 και ένα τζιπ, είναι και τα δύο αυτοκίνητα, παρόλο που έχουν τελείως διαφορετικά χαρακτηριστικά.

Στα Μαθηματικά, όταν λύνουμε ένα πρόβλημα, η απόδοση του αγνώστου στον χαρακτήρα Χ (Έστω Χ …) είναι μια αφαιρετική διαδικασία. Η αφαιρετική ικανότητα είναι απαραίτητη για τη σωστή ανάλυση ενός προβλήματος σε απλούστερα υποπροβλήματα και τη σαφή διατύπωσή της δομής του.

1.2 Ανάλυση ενός προβλήματος σε απλούστερα υποπροβλήματα

Υπάρχουν διάφορες επιστημονικές μέθοδοι για την ανάλυση της δομής ενός προβλήματος και την εύρεση των τμημάτων (υποπροβλημάτων) που το αποτελούν. Οι μέθοδοι αυτές είναι γνωστές με τα ονόματα *Αναλυτική* (Από Πάνω προς τα Κάτω-Top Down), *Συνθετική* (Από Κάτω προς τα Πάνω-Bottom Up), και Μικτή (Mixed) μέθοδος.

Στο κεφάλαιο αυτό θα γνωρίσουμε την **Αναλυτική μέθοδο επίλυσης προβλήματος** (Top Down problem solving) η οποία είναι η ευρύτερα εφαρμοζόμενη μέθοδος.

Η γενική αρχή της είναι ότι για να λύσουμε κάποιο σύνθετο πρόβλημα πρέπει:

1. Να καθορίσουμε τα υποπροβλήματα.
2. Να επαναλάβουμε τη διαδικασία αυτή για κάθε ένα από τα υποπροβλήματα, όσο αυτό είναι αναγκαίο.
3. 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.

Ας υποθέσουμε ότι τίθεται ως **πρόβλημα** το θέμα αντιμετώπισης των ναρκωτικών.

Η αντιμετώπιση του προβλήματος θα γίνει απλούστερη αν μπορέσουμε να αναλύσουμε το πρόβλημα σε άλλα απλούστερα. Το αρχικό πρόβλημα είναι "Αντιμετώπιση ναρκωτικών". Αυτό θα μπορούσε να αναλυθεί καταρχήν σε τρία επιμέρους προβλήματα:

(1) Πρόληψη.

(2) Θεραπεία.

(3) Επανένταξη.

Τα τρία αυτά επιμέρους προβλήματα πιθανό να μην είναι ιδιαίτερα λεπτομερή έτσι ώστε να επιτρέπουν την εύκολη αντιμετώπισή τους. Πρέπει λοιπόν κάθε ένα από αυτά να αναλυθεί σε ακόμα απλούστερα.

Έτσι λοιπόν το επιμέρους πρόβλημα (1) Πρόληψη μπορεί να αναλυθεί σε:

(1.1) Σωστή ενημέρωση των πολιτών σχετικά με το θέμα.

(1.2) Υποβοήθηση προς την κατεύθυνση ανάπτυξης ενδιαφερόντων, οραμάτων και στόχων εκ μέρους των εφήβων.

(1.3) Υποστήριξη ομάδων αυξημένης θεωρητικά "προδιάθεσης".

Όμοια το επιμέρους πρόβλημα (2) Θεραπεία μπορεί να αναλυθεί ως εξής:

(2.1) Δημιουργία νέων κρατικών θεραπευτικών κοινοτήτων.

(2.2) Ενίσχυση υπαρχουσών θεραπευτικών κοινοτήτων.

(2.3) Δημιουργία κατάλληλων τμημάτων στα δημόσια νοσοκομεία.

Παρόμοια το επιμέρους πρόβλημα (3) Επανένταξη μπορεί να αναλυθεί ως ακολούθως:

(3.1) Καταπολέμηση της κοινωνικής προκατάληψης απέναντι στους απεξαρτημένους.

(3.2) Επιδότηση θέσεων εργασίας για απεξαρτημένους πρώην χρήστες.

Στη συνέχεια και το πρόβλημα (1.1) μπορεί να αναλυθεί σε απλούστερα:

(1.1.1) Ενημέρωση των εφήβων μέσα από κατάλληλα προγράμματα στα σχολεία.

(1.1.2) Ενημέρωση των γονέων με προγράμματα του Δήμου.

(1.1.3) Ενημέρωση κάθε άλλου ενδιαφερόμενου πολίτη με προγράμματα του Υπουργείου Υγείας.

Μια παρόμοια παραπέρα ανάλυση θα μπορούσε να γίνει και για το πρόβλημα (1.2), το οποίο θα μπορούσε να αναλυθεί στα εξής απλούστερα προβλήματα:

(1.2.1) Οργάνωση πολιτιστικών δραστηριοτήτων στα σχολεία.

(1.2.2) Δημιουργία δημόσιων χώρων άθλησης στις γειτονιές για τους νέους.

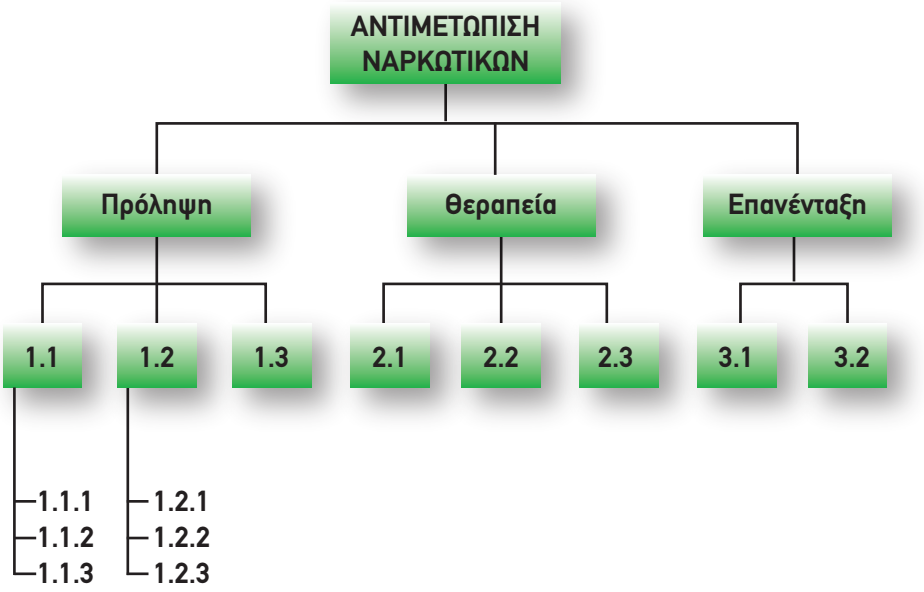
(1.2.3) Παροχή κινήτρων στα παιδιά και στους νέους για παρακολούθηση και συμμετοχή σε καλλιτεχνικά γεγονότα.

Αν η ανάλυση του αρχικού προβλήματος θεωρείται επαρκής, η διάσπαση των επιμέρους προβλημάτων σε άλλα απλούστερα μπορεί να τερματιστεί. Ο παραπάνω τρόπος περιγραφής και ανάλυσης ενός προβλήματος γίνεται φραστικά. Ο ενδιαφερόμενος για την αντιμετώπιση του αρχικού προβλήματος έχει πλέον μπροστά του να αντιμετωπίσει μια σειρά από επιμέρους προβλήματα τα οποία, στο σύνολό τους, εκφράζουν και αντιστοιχούν στο αρχικό πρόβλημα.

Η ανάλυση αυτή του προβλήματος σε άλλα απλούστερα αναδύει, παράλληλα, και τη δομή του προβλήματος. Για τη γραφική απεικόνιση της δομής ενός προβλήματος χρησιμοποιείται συχνότατα η διαγραμματική αναπαράσταση, σύμφωνα με την οποία:

- Το αρχικό πρόβλημα αναπαρίσταται από ένα ορθογώνιο παραλληλόγραμμο.
- Κάθε ένα από τα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα αναπαρίσταται επίσης από ένα ορθογώνιο παραλληλόγραμμο.
- Τα παραλληλόγραμμα που αντιστοιχούν στα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα 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 του παραδείγματος που παρατίθεται προηγούμενα φαίνεται στο σχήμα 1-1.



Σχήμα 1-1. Διαγραμματική αναπαράσταση προβλήματος "Αντιμετώπιση ναρκωτικών"

Η διαγραμματική αναπαράσταση προσφέρει μια απτή απεικόνιση της δομής του προβλήματος. Η δημιουργία του σχετικού διαγράμματος βοηθάει τόσο στην καλύτερη κατανόηση του ίδιου του προβλήματος, όσο και στη σχεδίαση της λύσης του.

1.2.1 Κατανόηση του προβλήματος και της δομής του - Διαδικασία επίλυσης προβλημάτων

Στις προηγούμενες παραγράφους παρουσιάστηκε η διαδικασία κατανόησης της δομής ενός προβλήματος, διαδικασία στην οποία δεν περιορίζεται μόνον η επίλυση ενός προβλήματος. Αυτό, διότι αποτελεί μια σύνθετη λειτουργία την οποία μπορούμε να διαιρέσουμε στα παρακάτω στάδια:

- Κατανόηση του προβλήματος και απλοποίηση της περιγραφής του, χωρίς απώλεια χρήσιμης πληροφορίας (διαδικασία αφαίρεσης).
- Ανάλυσή του σε απλούστερα υποπροβλήματα.

- Διατύπωση σκέψεων σχετικά με το είδος του προβλήματος. Στο στάδιο αυτό τίθενται ερωτήματα όπως: εντάσσεται σε μια γενικότερη ομάδα προβλημάτων, είναι ειδική περίπτωση ενός γενικού προβλήματος;
- Λογική οργάνωση και ανάλυση δεδομένων.
- Αναγνώριση, ανάλυση και υλοποίηση πιθανών λύσεων. Ανίχνευση γνωστών προτύπων. Στο στάδιο αυτό θέτονται ερωτήματα όπως: Η λύση του προβλήματος περιλαμβάνει τμήματα τα οποία τα έχουμε συναντήσει και σε άλλα προβλήματα; Γνωστό τμήμα, για παράδειγμα, είναι η ταξινόμηση ενός πλήθους αριθμών.
- Κατασκευή του αλγόριθμου περιγράφοντας τις ενέργειες για τη λύση του.
- Αξιολόγηση του αλγόριθμου και της λύσης που δόθηκε. Στο στάδιο αυτό τίθενται ερωτήματα όπως: Είναι σωστή η λύση που βρέθηκε; Μήπως ο αλγόριθμος που δημιουργήσαμε μπορεί να βελτιωθεί έτσι ώστε, για παράδειγμα, ο χρόνος εκτέλεσης του αλγόριθμου να μειωθεί;
- Επιστροφή, αν χρειαστεί, σε προηγούμενα στάδια και ανακατασκευή της λύσης, μέχρι την επίλυση του προβλήματος με αποτελεσματικό τρόπο.
- Γενίκευση της λύσης, ώστε να μπορεί να εφαρμοστεί σε παρόμοια προβλήματα.

Τόσο το στάδιο της αξιολόγησης του αλγόριθμου όσο και η αξιολόγησης της λύσης, είναι δύο πολύ σημαντικά στάδια. Η αξιολόγηση του αλγόριθμου είναι συχνά ένα σύνθετο θέμα και η ερώτηση: υπάρχει αποδοτικότερος αλγόριθμος που λύνει το ίδιο πρόβλημα; απαντάται συχνά από τη θεωρητική Πληροφορική, η οποία μελετά την αποδοτικότητα των αλγορίθμων. Η αποδοτικότητα αναφέρεται κυρίως στην ταχύτητα εκτέλεσης του αλγόριθμου (πόσο χρόνο δηλαδή χρειάζεται για να λύσει το πρόβλημα) και στην ποσότητα της κύριας μνήμης που χρησιμοποιεί. Ο έλεγχος της λύσης αντιμετωπίζεται συχνά με τυποποιημένα υποθετικά δεδομένα σύμφωνα με τις απαιτήσεις του προβλήματος, τα οποία οδηγούν σε μια γνωστή λύση. Στη συνέχεια, εφαρμόζουμε τον αλγόριθμο και συγκρίνουμε τα αποτελέσματα που προκύπτουν με τα δικά μας. Εάν διαπιστώσουμε λάθος, εντοπίζουμε το τμήμα της λύσης που εκτελεί τη λανθασμένη λειτουργία, το διορθώνουμε και επαναλαμβάνουμε τη διαδικασία ελέγχου, έως ότου τα αποτελέσματα να μη διαφέρουν από τα δικά μας.

1.3 Περιγραφή με ψευδοκώδικα/διάγραμμα ροής

Στη βιβλιογραφία συναντώνται διάφοροι τρόποι αναπαράστασης ενός αλγόριθμου, όπως οι ακόλουθοι που θα παρουσιαστούν στη συνέχεια:

- Με **φυσική γλώσσα**, η οποία αποτελεί τον πιο απλό, ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης ενός αλγόριθμου, που με απλά λόγια και ελεύθερες εκφράσεις, περιγράφουμε τα βήματά του. Ωστόσο, ο τρόπος αυτός έκφρασης ενέχει αυξημένη πιθανότητα λάθους ή ασάφειας.
- Με **διαγραμματικές τεχνικές** (diagramming techniques), οι οποίες συνιστούν ένα γραφικό τρόπο παρουσίασης του αλγόριθμου. Από τις διάφορες διαγραμματικές τεχνικές που έχουν επινοηθεί η πιο παλιά και η πιο γνωστή ίσως είναι το διάγραμμα ροής (flow chart).
- Με **κωδικοποίηση** (coding), δηλαδή με ένα πρόγραμμα γραμμένο είτε σε μία ψευδογλώσσα είτε σε κάποια γλώσσα προγραμματισμού που, όταν εκτελεσθεί, θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.
- Με **Ψευδογλώσσα** ή **Ψευδοκώδικας**, που είναι μια υποθετική δομημένη γλώσσα με στοιχεία από υπαρκτές γλώσσες προγραμματισμού, με λίγες εντολές και απλοποιημένη σύνταξη. Χρησιμοποιείται, κυρίως, στα πρώτα στάδια εκμάθησης του προγραμματισμού.

Στο βιβλίο αυτό θα χρησιμοποιηθεί ως γλώσσα προγραμματισμού η Python, καθώς επίσης η ψευδογλώσσα και τα σύμβολα του διαγράμματος ροής που χρησιμοποιούνται και στο Βιβλίο «Εισαγωγή στις Αρχές της Επιστήμης των Υπολογιστών» της Β' τάξης ΓΕ.Λ. και ΕΠΑ.Λ.

1.3.1 Βασικές συνιστώσες/εντολές ενός αλγόριθμου

Στη συνέχεια δίνονται παραδείγματα αλγορίθμων, στους οποίους εξετάζονται οι διάφορες συνιστώσες ενός αλγόριθμου, δηλαδή οι απαραίτητες εντολές, ξεκινώντας από τις απλούστερες και προχωρώντας προς τις συνθετότερες. Πιο συγκεκριμένα, θα εξετασθούν περιπτώσεις σειριακών εντολών, αναθέσεων τιμών, επιλογής με κριτήρια, διαδικασιών επανάληψης, ενεργειών πολλαπλών επιλογών, καθώς και συνδυασμού εμφωλευμένων περιπτώσεων. Για κάθε περίπτωση παρουσιάζονται σχετικά παραδείγματα με την εκφώνηση, την παρουσίαση των βημάτων που πρέπει να ακολουθηθούν σε ψευδογλώσσα, καθώς και το αντίστοιχο διάγραμμα ροής.

Για τη διδασκαλία της παραγράφου αυτής θα χρησιμοποιηθεί η παράγραφος 4.4 από το βιβλίο «Προγραμματισμός Υπολογιστών» των Σιδερίδη κ.ά. του ΕΠΑ.Λ., καθώς και το Βιβλίο «Εισαγωγή στις Αρχές της Επιστήμης των Υπολογιστών» της Β' τάξης ΓΕ.Λ. και ΕΠΑ.Λ. των Δουκάκη κ.ά. **Προσοχή:** η σύνταξη των εντολών ακολουθεί το δεύτερο βιβλίο «Εισαγωγή στις Αρχές της Επιστήμης των Υπολογιστών» της Β' τάξης ΓΕ.Λ. και ΕΠΑ.Λ. των Δουκάκη κ.ά.

1.4 Δραστηριότητες

Δραστηριότητα 1. Διαχείριση προβλημάτων

Για τα προβλήματα που δίνονται στη συνέχεια, να συμπληρωθεί η ακόλουθη πληροφορία:

α) Κατανόηση του προβλήματος:

- Ποια δεδομένα είναι γνωστά;
- Τι δεν είναι γνωστό;
- Ποιο είναι το ζητούμενο;
- Ποιες είναι οι συνθήκες;
- Να σχεδιαστεί η λύση.
- Ποιο είναι το πλάνο εργασίας για την επίλυση του προβλήματος;

β) Υλοποίηση της λύσης

- Χρησιμοποιώντας το πλάνο εργασίας, να παρουσιαστεί η όλη εργασία και η λύση.

Αφού ολοκληρώσετε τη δραστηριότητα, να συζητήσετε στην τάξη για το πρόβλημα και τους τρόπους λύσης του.

Σημείωση: Να γίνει καταγραφή των συλλογισμών που έγιναν πάνω στη λύση αυτή.

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 9 γραμμάρια το καθένα. Πώς μπορούμε, με μία μόνο ζύγιση σε μια 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 έναν μαθητή σε ένα δωμάτιο και πρέπει ο μαθητής να ανταλλάξει χειραψία με κάθε έναν από αυτούς. Με πόσους ανθρώπους τελικά, θα ανταλλάξει χειραψία; Εάν υπάρχουν N (N>0) άνθρωποι μαζί με το μαθητή στο δωμάτιο, με πόσους τελικά θα έρθει αυτός σε επαφή;

Δραστηριότητα 2. Μελέτη προβλήματος «Ασφαλής πλοήγηση στο Διαδίκτυο»

Χωριστείτε σε ομάδες των 5 ατόμων. Κάθε ομάδα αναλαμβάνει να μελετήσει το πρόβλημα: «Ασφαλής πλοήγηση στο Διαδίκτυο – δημιουργία ενός δεκαλόγου συμβουλευτικών οδηγιών». Για το σκοπό αυτό:

- Συλλέξτε πληροφορίες από το Διαδίκτυο, επιλέξτε την ουσιώδη πληροφορία, αξιοποιήστε διαθέσιμα στατιστικά δεδομένα, οργανώστε τα δεδομένα που εντοπίσατε. Για τη συλλογή και την οργάνωση των δεδομένων, αξιοποιήστε τους υπολογιστές του εργαστηρίου Πληροφορικής και πηγές στο Διαδίκτυο.
- Στη συνέχεια αναλύστε το πρόβλημα σε απλούστερα προβλήματα.
- Συντάξτε έναν συμβουλευτικό οδηγό με τις δέκα βασικότερες οδηγίες και παρουσιάστε την εργασία σας στη τάξη.

Σημείωση: Για την υποστήριξη της συνεργασίας των ομάδων, μπορεί να αξιοποιηθεί ένα Διαδικτυακό εργαλείο συνεργατικής γραφής. Για παράδειγμα μπορεί να αξιοποιηθεί το εργαλείο wiki της ηλεκτρονικής σχολικής τάξης (http://eclass.sch.gr/). Οργανώστε το υλικό σας και δημιουργήστε ένα κοινό έγγραφο που χωρίζεται σε δύο μέρη: Το πρώτο μέρος περιέχει την ανάλυση του προβλήματος και το δεύτερο το δεκάλογο των συμβουλευτικών οδηγιών.

Πηγές για την υλοποίηση της δραστηριότητας

- Δικτυακός κόμβος του Συνήγορου του παιδιού (http://www.0-18.gr). Δείτε τη «συνταγή» για ασφαλή πλοήγηση στο Διαδίκτυο, όπως προέκυψε από το ηλεκτρονικό φόρουμ της Κοινότητας Εφήβων Συμβούλων του Συνήγορου του Παιδιού http://www.0-18.gr/gia-paidia/nea/syntagi-gia-asfali-ploigisi-sto-internet (τελευταία πρόσβαση 16/07/2015).
- Ελληνικό Κέντρο Ασφαλούς Διαδικτύου με δραστηριότητες, άρθρα, εκπαιδευτικούς πόρους, προτάσεις (http://www.saferinternet.gr, τελευταία πρόσβαση 16/07/2015).
- Ασφάλεια στο Διαδίκτυο: Ενημερωτικός κόμβος Πανελλήνιου Σχολικού Δικτύου (http://internet-safety.sch.gr/, τελευταία πρόσβαση 16/07/2015).

Δραστηριότητα 3. Σχεδιασμός ενός εκπαιδευτικού παιχνιδιού

Για το πρόβλημα της διαδικασίας λήψης σημαντικών αποφάσεων για το σχεδιασμό ενός εκπαιδευτικού παιχνιδιού με μορφή quiz, κάντε τα ακόλουθα:

1. Αρχικά διατυπώστε με σαφήνεια το πρόβλημα και αναλύστε το σε απλούστερα υποπροβλήματα. Για την καταγραφή της συλλογιστικής σας και την αναπαράσταση των κατηγοριών του προβλήματος μπορεί να χρησιμοποιηθούν εργαλεία εννοιολογικής χαρτογράφησης (όπως το Cmaptools http://cmaptools.en.softonic.com/) ή ένα απλό υπολογιστικό φύλλο.
2. Στη συνέχεια περιγράψτε με απλά βήματα τη διαδικασία λήψης των αποφάσεων σας για το σχεδιασμό του εκπαιδευτικού παιχνιδιού, λαμβάνοντας υπόψη διάφορες μεταβλητές, γνωστές και άγνωστες, όπως: οι κανόνες του παιχνιδιού, ο τρόπος βαθμολογίας, το γραφικό περιβάλλον κ.ά.
3. Για ένα μικρό τμήμα του παιχνιδιού περιγράψτε έναν αλγόριθμο που για μια ερώτηση, αρχικά, θα διαβάζει την απάντηση και θα ελέγχει την ορθότητά της. Στη συνέχεια, θα επιστρέφει στον παίκτη του παιχνιδιού ανάλογο μήνυμα. Το παιχνίδι θα σταματάει, αν ο παίκτης απαντήσει πάνω από τρεις φορές λάθος. Προσπαθήστε να γενικεύσετε τη λύση που περιγράψατε στον αλγόριθμο, ώστε να εφαρμοστεί για μια μεγαλύτερη ομάδα ερωτήσεων.

Δραστηριότητα 4

Μια ομάδα μαθητών αποφάσισαν να δημιουργήσουν μια μικρογραφία σε μακέτα ενός σπιτιού για να προσομοιώσουν τη λειτουργία μιας «έξυπνης» κατοικίας. Για το σκοπό αυτό, συνέδεσαν διάφορους αισθητήρες με ένα μικροεπεξεργαστή, που θα ελέγχει το εξωτερικό περιβάλλον και ανάλογα θα ενεργοποιούνται διάφοροι αυτοματισμοί (άνοιγμα παραθύρου, άνοιγμα κουρτινών, ενεργοποίηση ανεμιστήρα κ.ά.) για την προσαρμογή του σπιτιού, ώστε να διατηρείται μια σταθερή θερμοκρασία γύρω στους 20 με 25 βαθμούς ^{o}C με το ελάχιστο δυνατό κόστος.

- Να αναλύσετε το πρόβλημα σε απλούστερα υποπροβλήματα, ώστε να υποστηρίξετε το σχεδιασμό της διάταξης.
- Να καταγράψτε τις κατηγορίες και τις παραμέτρους που πρέπει να λάβετε υπόψη σας για την υλοποίηση της λύσης. αφού κρατήσετε τις χρήσιμες πληροφορίες.
- Να υλοποιήσετε αλγόριθμο, εκφρασμένο σε λογικό διάγραμμα, που να περιγράφει τα βασικά βήματα που πρέπει να εκτελέσει ο μικροεπεξεργαστής, ώστε, αν η

θερμοκρασία αυξηθεί, να ανοίξει αυτόματα το παράθυρο της σοφίτας για να μπει φρέσκος αέρας. Από ποιον αισθητήρα θα πάρει αυτήν την πληροφορία ως είσοδο; Ποιες συνθήκες πρέπει να ικανοποιηθούν ώστε να κλείσει το παράθυρο;

Επέκταση: Ποια νέα προβλήματα μπορεί να δημιουργηθούν, όταν ανοίξει το παράθυρο, ώστε να ληφθούν υπόψη οι συνθήκες που πρέπει τελικά να ικανοποιηθούν για να δοθεί η αντίστοιχη εντολή και να ανοίξει το παράθυρο;

Σημείωση: Εναλλακτικά μπορεί να αποφασίσετε να επιλύσετε κάποιο άλλο υποπρόβλημα που συντελεί στην διατήρηση της επιθυμητής θερμοκρασίας.

1.5 Ερωτήσεις

1. Τι είναι ένα πρόβλημα;
2. Ποια είναι τα βασικά βήματα για την επίλυση ενός προβλήματος;
3. Μπορείτε να περιγράψετε με ένα δικό σας παράδειγμα την αξία της αφαίρεσης;
4. Με ποιους τρόπους μπορούμε να αναπαραστήσουμε έναν αλγόριθμο;

1.6 Αναφορές – Βιβλιογραφία

** Το παράδειγμα της ανάλυσης ενός προβλήματος στην παράγραφο 1.3 προέρχεται από το βιβλίο «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον» της Γ' Τάξης ΓΕ.Λ. των Βακάλη κ.ά.

http://demonstrations.wolfram.com/topics.html?Algorithms#123 (τελευταία προσπέλαση 15/07/2015)

Δικτυακός τόπος για τη διδασκαλία της επιστήμης της Πληροφορικής χωρίς υπολογιστές Computer Science Unplugged: http://www.csunplugged.org (τελευταία προσπέλαση 12/07/2015).

Δικτυακός τόπος με ψηφιακό υλικό για την εισαγωγή στην αλγοριθμική σκέψη: http://www.teaching-materials.org/algorithms/ (τελευταία προσπέλαση 05/01/2015).

Εκπαιδευτικά παιχνίδια με θέμα την υπολογιστική σκέψη και την επίλυση προβλήματος: http://games.thinkingmyself.com/ (τελευταία προσπέλαση 12/07/2015).

Επίδειξη αλγορίθμων με οπτικοποίηση και δυνατότητα ανατροφοδότησης. Ο Δικτυακός Κόμβος αποτελεί τον κόμβο διανομής του Wolfram Demonstrations Project, έμπνευση του Stephen Wolfram. Το πρόγραμμα έχει ως στόχο την παραγωγή και διανομή ΕΛ/ΛΑΚ εκπαιδευτικών διαδραστικών και οπτικοποιημένων επιδείξεων για την επιστήμη, την τεχνολογία, τα μαθηματικά, την τέχνη και για αρκετά άλλα επιστημονικά πεδία. Για την εκτέλεση των αρχείων επίδειξης στον υπολογιστή απαιτείται η εγκατάσταση του Wolfram CDF Player (παρέχεται δωρεάν). Στη συνέχεια, μπορεί κανείς να κατεβάσει και να εκτελέσει το αρχείο επίδειξης ή να το εκτελέσει απευθείας (on-line) από το φυλλομετρητή-browser.

Εργαλεία εννοιολογικής χαρτογράφησης (όπως Cmap-tool http://cmaptools.en.softonic.com/, τελευταία προσπέλαση 15/07/2015).

Εργαλείο on-line για τη κατασκευή διαγραμμάτων ροής https://www.gliffy.com (τελευταία προσπέλαση 15/07/2015).

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 Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ. ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π.), Ιανουάριος 2015.

Ψηφιακό Σχολείο-Αποθετήριο Φωτόμετρο. Περιέχει μεταξύ άλλων μαθησιακά αντικείμενα για την προσέγγιση της αλγοριθμικής σκέψης (τελευταία προσπέλαση 05/07/2015) http://photodentro.edu.gr/lor/subject-search?locale=el

2

Από τον αλγόριθμο στην ανάπτυξη προγράμματος

2. Από τον αλγόριθμο στην ανάπτυξη προγράμματος

Στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- περιγράφουμε την πορεία από τον αλγόριθμο στο πρόγραμμα
- επεξηγούμε τη λογική συγγραφής προγραμμάτων ανά είδος προγραμματισμού (Διαδικαστικός, Αντικειμενοστρεφής και Συναρτησιακός προγραμματισμός).

Λέξεις κλειδιά

Κύκλος ανάπτυξης προγράμματος, μοντέλα ανάπτυξης λογισμικού, προγραμματιστικά υποδείγματα, πρότυπα και τεχνικές προγραμματισμού, γλώσσες προγραμματισμού.

Διδακτικές ενότητες

2.1 Κύκλος ανάπτυξης προγράμματος/λογισμικού

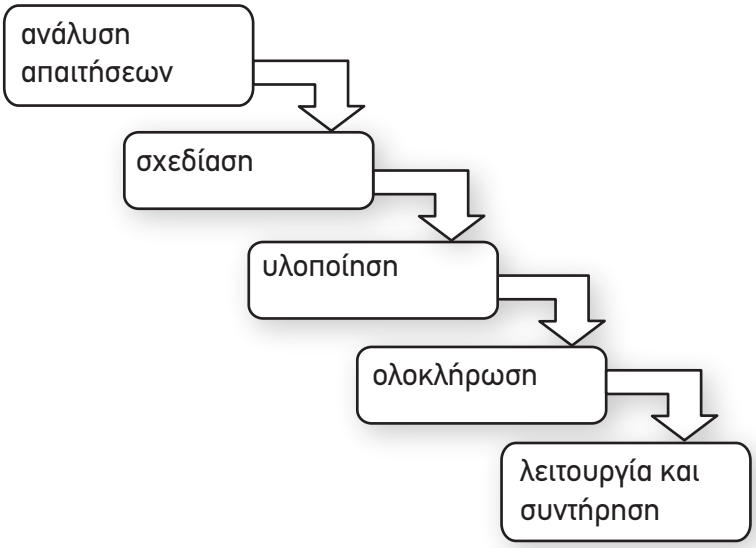
Η διαδικασία ανάπτυξης λογισμικού αποτελεί μια εργασία που εξελίσσεται σε διακριτές φάσεις ή στάδια και θεωρείται υποσύνολο του κύκλου ζωής ενός συστήματος Λογισμικού που ξεκινά από την ανάλυση απαιτήσεων και τελειώνει με την παύση λειτουργίας του.

Μεταξύ των βασικών μεθοδολογιών (μοντέλων) -που έχουν προταθεί και ακολουθούνται- είναι το μοντέλο του Καταρράκτη (Waterfall model) και αυτό 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:

- Ανάλυση απαιτήσεων.
- Σχεδίαση.
- Υλοποίηση.
- Ολοκλήρωση.
- Λειτουργία και συντήρηση.



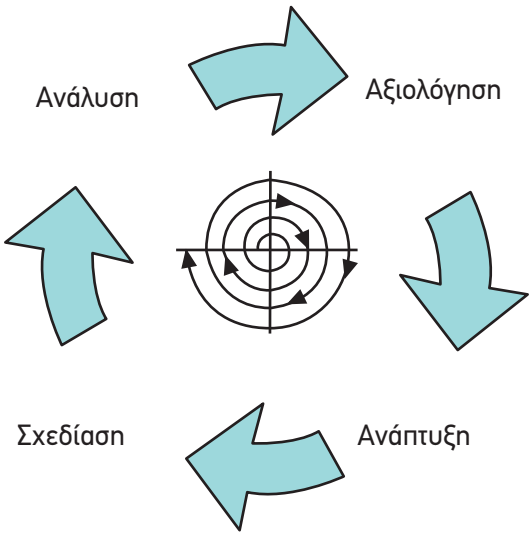
2.1.2 Μοντέλο σπείρας

Στο μοντέλο της σπείρας, η ανάπτυξη ακολουθεί μια εξελικτική διαδικασία με την επαναληπτική εκτέλεση ενός κύκλου φάσεων. Σε καθεμία φάση 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.

Παρόμοια με τη διαδικασία ανάπτυξης ενός λογισμικού, εργαζόμαστε και κατά την ανάπτυξη ενός προγράμματος, ακολουθώντας πάλι μια μεθοδολογία υλοποίησης του *κύκλου ανάπτυξης προγράμματος* (program development life cycle-PDLC).

Μια από τις μεθοδολογίες ανάπτυξης προγράμματος αποτελείται από φάσεις, όπως:

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



2.2 Η λογική συγγραφής προγράμματος ανάλογα με το είδος προγραμματισμού

Από τη δεκαετία του 1960 μέχρι σήμερα, έχουν αναπτυχθεί διάφορα είδη προγραμματισμού που τα υποστήριξαν πολλές γλώσσες και θα μπορούσαμε να τα κατηγοριοποιήσουμε σε μεγάλες κατευθύνσεις, τα λεγόμενα **Προγραμματιστικά Υποδείγματα** (programming paradigms).

Τα βασικά προγραμματιστικά υποδείγματα είναι τα ακόλουθα:

- Ο Προστακτικός προγραμματισμός (imperative programming).
- Ο Δηλωτικός προγραμματισμός (declarative programming).

2.2.1 Προστακτικός προγραμματισμός

Ο **Προστακτικός προγραμματισμός** βασίζεται σε εντολές που υλοποιούν τα βήματα ενός αλγόριθμου, ενεργώντας σε μεταβλητές και αλλάζοντας την κατάστασή τους. Βρίσκεται πιο κοντά στη λογική λειτουργίας του υπολογιστή. Γλώσσες που ακολούθησαν το είδος αυτό είναι οι κλασικές γλώσσες προγραμματισμού, όπως Cobol, Fortran, Pascal, C κ.ά.

2.2.1.1 Δομημένος και μη προγραμματισμός

Κατά την αρχική περίοδο του προγραμματισμού, η διακλάδωση της ροής γίνονταν με την εντολή goto, κατάσταση που οδηγούσε σε μη δομημένα προγράμματα (**μη δομημένος προγραμματισμός** - unstructured programming). Στη συνέχεια δημιουργήθηκε το πρότυπο του **Δομημένου προγραμματισμού** (structured programming), με τις εντολές σε ομάδες (blocks) να ακολουθούν την **Ιεραρχική** λογική ροής και τη δυνατότητα χρήσης υπορουτινών και διάφορων άλλων δομών, όπως η if-then-else.

Ο Διαδικαστικός προγραμματισμός (procedural programming) αποτελεί μια υποκατηγορία του Δομημένου προγραμματισμού, με το πρόγραμμα να αποτελείται από αυτοτελείς ομάδες εντολών, τις **διαδικασίες** (procedures). Η γλώσσα προγραμματισμού που αξιοποίησε αρχικά το είδος αυτό είναι η Pascal, από το 1970.

Ο **Αντικειμενοστραφής** προγραμματισμός (object-oriented programming) βασίζεται, σε αντίθεση με το Διαδικαστικό προγραμματισμό, σε *αντικείμενα* που αλληλεπιδρούν μεταξύ τους, αποτελώντας πρότυπο που ταιριάζει περισσότερο στη λογική οργάνωσης και λειτουργίας του πραγματικού κόσμου.

2.2.2 Δηλωτικός προγραμματισμός

Ο **Δηλωτικός** προγραμματισμός (declarative programming paradigm) βασίζεται στην περιγραφή του σκοπού, τον οποίο ζητείται από το πρόγραμμα να επιτύχει. Στο γενικό αυτό υπόδειγμα ανήκουν διάφορες υποκατηγορίες προγραμματισμού, όπως είναι ο Συναρτησιακός και ο Λογικός.

Ο **Συναρτησιακός** προγραμματισμός (functional programming) βασίζεται σε μαθηματικές συναρτήσεις, με γλώσσες όπως Lisp, Logo κ.ά.

Στο **Λογικό** προγραμματισμό (logic programming), ένα πρόγραμμα είναι ένα σύνολο από αξιώματα ή κανόνες οι οποίοι καθορίζουν σχέσεις ανάμεσα σε αντικείμενα. Υπολογισμός ενός λογικού προγράμματος είναι ένα συμπέρασμα που συνάγεται από τα αποτελέσματά του.

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).

2.2.3 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 τις οποίες θα αναφέρουμε στη συνέχεια.

Παράλληλος προγραμματισμός (parallel programming). Επιτρέπει ταυτόχρονη εκτέλεση διαδικασιών από διαφορετικούς επεξεργαστές.

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. Παράδειγμα αποτελεί η Microsoft Visual-Basic.

Οπτικός προγραμματισμός (visual programming). Δεν αποτελεί υπόδειγμα, αλλά εκφράζει τη δυνατότητα γλωσσών ή περιβαλλόντων προγραμματισμού να παρέχουν τη δυνατότητα δημιουργίας του προγράμματος μέσω γραφικών αντικειμένων, αντί της πληκτρολόγησης του κειμένου που αντιστοιχεί σε εντολές. Οι γλώσσες οπτικού προγραμματισμού βασίζονται, άλλες σε γραφικά με τη μορφή εικονιδίων (icon-based languages), άλλες σε διαγράμματα (diagram languages) και τέλος, άλλες σε φόρμες (form based languages).

Στην κατηγορία αυτή ανήκουν περιβάλλοντα όπως το Authorware της Adobe, περιβάλλοντα δημιουργίας σεναρίων όπως το Kodu της Microsoft και το Alice, το MIT Scratch, το Greenfoot.

Προγραμματισμός δέσμης ενεργειών (script programming) είναι τύπος - και όχι υπόδειγμα- προγραμματισμού δημιουργίας μικρών τμημάτων κώδικα και όχι ολοκληρωμένων προγραμμάτων. Είναι υψηλού επιπέδου προγραμματισμός που διερμηνεύεται κατά την εκτέλεση από ένα άλλο πρόγραμμα, όπως ένας φυλλομετρητής.

Αρθρωτός ή Τμηματικός Προγραμματισμός (modular programming). Σχετίζεται περισσότερο με τεχνική σχεδίασης 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, ευκολία κωδικοποίησης και συντήρησης. Γενικά ως τμήμα (module) θεωρούμε ένα σύνολο ενεργειών το οποίο εκτελεί μια καθορισμένη λειτουργία ενός προγράμματος και είναι κατά το 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 επίπεδο και στη συνέχεια το αναλύουμε σε όλο και χαμηλότερα, έως ότου φθάσουμε στο κατώτερο επίπεδο ανάλυσης. Η τεχνική αυτή ονομάζεται **ιεραρχικός σχεδιασμός** (top down design).

2.2.4 Ενδεικτικά περιβάλλοντα και γλώσσες προγραμματισμού

Pascal. Η πλέον κλασική γλώσσα δομημένου προγραμματισμού στην κλασική της έκδοση.

Visual Basic (http://www.microsoft.com). Περιβάλλον προγραμματισμού που ακολουθεί μικτό πρότυπο υποδειγμάτων.

C++. Επέκταση της C. Αποτελεί γλώσσα αντικειμενοστραφούς προγραμματισμού, αν και μπορεί να χρησιμοποιηθεί και για διδασκαλία διαδικαστικού προγραμματισμού.

Java. Σύγχρονη γλώσσα αντικειμενοστραφούς προγραμματισμού.

Python (http://python.org/about/). Γλώσσα που ανήκει ουσιαστικά σε μικτά υποδείγματα προγραμματισμού, όπως Συναρτησιακό, Αντικειμενοστραφές.

Prolog (Programming in Logic). Γλώσσα Λογικού προγραμματισμού.

2.2.5 Δραστηριότητα. Κατάταξη γλωσσών προγραμματισμού στα προγραμματιστικά υποδείγματα

Βήμα 1. Δημιουργήστε έναν πίνακα δύο διαστάσεων, με τις στήλες να αντιστοιχούν στις έννοιες που ήδη αναφέρθηκαν στην ενότητα του αντικειμενοστρεφούς προγραμματισμού και τις γραμμές του σε Γλώσσες προγραμματισμού ή και περιβάλλοντα που γνωρίζετε, λειτουργώντας σε ομάδες.

Βήμα 2. Βρείτε, με αναζήτηση, γλώσσες ή περιβάλλοντα προγραμματισμού που σας ενδιαφέρουν και γράψτε τις ονομασίες τους στις γραμμές.

Βήμα 3. Αναζητήστε στο Διαδίκτυο χαρακτηριστικά κάθε γλώσσας ή περιβάλλοντος σε σχέση με τις έννοιες της ενότητας, σημειώνοντας στο αντίστοιχο κελί του πίνακα που αντιστοιχούν. Να λάβετε υπόψη ότι πολλά περιβάλλοντα ανήκουν σε μικτό σχήμα Προγραμματιστικού Υποδείγματος (multiparadigm environments).

Βοήθεια. Μεταξύ των αναζητήσεών σας, μπορείτε να δείτε τις: Object Pascal - Apple Computer, Delphi της Borland έως το 2006, C, C#, C++, Java, Eiffel, Kodu Microsoft, AppInventor, Greenfoot.

2.3 Αντικειμενοστρεφής προγραμματισμός

Στην παράγραφο αυτή θα αναφερθούμε συνοπτικά στον Αντικειμενοστρεφή Προγραμματισμό (Object-oriented programming - OOP). Πρόκειται για είδος προγραμματισμού που περιστρέφεται γύρω από την έννοια της **Κλάσης** (Class), η οποία περιγράφει **Αντικείμενα** (Objects), τα οποία περιέχουν δεδομένα στη μορφή **Ιδιοτήτων** (Properties) και κώδικα στη μορφή **Μεθόδων** (Methods).

Την ανάλυση των παραπάνω εννοιών θα την βρείτε στο βιβλίο της Γ' ΓΕ.Λ. "Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον" των Βακάλη Α., κ.ά. Ο.Ε.Δ.Β. στο κεφάλαιο 11 και στις παραγράφους 11.1 και 11.2.

Παράδειγμα 1

Κλάση: Όχημα

Ιδιότητες: Χρώμα, Τιμή, Αριθμός Τροχών, Ταχύτητα

Μέθοδοι: Επιτάχυνε, Φρέναρε

Ένα Αντικείμενο έχει συγκεκριμένες Ιδιότητες, όπως για παράδειγμα το Αντικείμενο "Αυτοκίνητό 1" με Ιδιότητες: Χρώμα: "Μαύρο", Τιμή: 10000, Αριθμός τροχών: 4, Ταχύτητα: 120 km/h, ή το Αντικείμενο "Μοτοσικλέτα 2" με Ιδιότητες: Χρώμα: Κόκκινο, Τιμή: 4000, Αριθμός τροχών: 2, Ταχύτητα: 100 km/h. Έτσι, μια κλάση αποτελεί ένα αφηρημένο σχήμα το οποίο αποκτά συγκεκριμένη υπόσταση όταν δημιουργούνται αντικείμενα όπως το "Αυτοκίνητο 1" και η "Μοτοσικλέτα 2", καθένα με τα δικά του χαρακτηριστικά. Οι μέθοδοι που περιγράφονται στην κλάση εμπεριέχουν κώδικα. Για παράδειγμα, οι μέθοδοι Επιτάχυνε και Φρέναρε, αυξάνουν και μειώνουν την ταχύτητα του αντικειμένου αντίστοιχα.

Παράδειγμα 2

Κλάση: Ζώο

Ιδιότητες: Ηλικία,Ύψος

Μέθοδοι: Μεγάλωσε, Ψήλωσε

Η μεταβλητή "Καμηλοπάρδαλη 1" είναι ένα αντικείμενο της Κλάσης Ζώο. Όταν το αντικείμενο δημιουργείται, οι ιδιότητες Ηλικία και Ύψος αρχικοποιούνται. Οι μέθοδοι της κλάσης ορίζουν τη συμπεριφορά των αντικειμένων και τον τρόπο με τον οποίο με-

ταβάλλεται η κατάστασή τους. Για παράδειγμα, οι μέθοδοι Μεγάλωσε και Ψήλωσε, αυξάνουν τις τιμές Ηλικία και Ύψος ενός αντικειμένου αντίστοιχα.

2.3.1 Δραστηριότητα. Εισαγωγή στον αντικειμενοστρεφή προγραμματισμό και στην οδήγηση από τα γεγονότα

Σε μια πλατφόρμα γραφικής ανάπτυξης παιχνιδιών, όπως π.χ. στο www.sploder.com, επιλέξτε κάποια από τις γραφικές μηχανές ανάπτυξης και κατασκευάστε ένα παιχνίδι που να ικανοποιεί τις εξής προδιαγραφές:

Να υπάρχουν σε αυτό, εκτός από το χαρακτήρα, τουλάχιστον 3 ακόμα **αντικείμενα** που να ανήκουν σε διαφορετικές κατηγορίες. Από τα αντικείμενα αυτά, τουλάχιστον 2 πρέπει να επιδρούν στο παιχνίδι με τρόπο που να μην περιορίζεται στο να αποτελούν εμπόδια πρόσβασης. Επίσης πρέπει να υπάρχουν τουλάχιστον 2 πίστες ή τμήματα της πίστας που να *ανοίγουν*, αφού ικανοποιηθούν κάποιες προϋποθέσεις. Η μετάβαση αυτή πρέπει να γίνεται μέσα στη ροή του παιχνιδιού και όχι, για παράδειγμα, μετά από οθόνη σκορ για την πίστα που ολοκληρώθηκε. Επιδιώξτε, αν η μηχανή που επιλέξατε επιτρέπει, να έχετε διαφορετικά εδάφη (terrains) στο παιχνίδι σας.

Υλοποιήστε σε στάδια:

Στάδιο 1. Σχεδιάστε την πρώτη πίστα σας, ώστε να περιλαμβάνει όλα τα διαφορετικά εδάφη. Τοποθετήστε τα *αντικείμενα* που επιλέξατε σε πολλαπλά *στιγμιότυπα* (παραπάνω από μια φορές το καθένα).

Στάδιο 2. Επιλέξτε το χαρακτήρα σας και τοποθετήστε τον στην πίστα. Μετατρέψτε το παιχνίδι σε εκτελέσιμο και δοκιμάστε το.

Παρατηρήστε πώς επηρεάζει το έδαφος την κίνηση του χαρακτήρα σας. Πώς πιστεύετε ότι υλοποιείται αυτό;

Παρατηρήστε τι συμβαίνει, όταν ο χαρακτήρας σας έρχεται σε επαφή με κάποιο από τα αντικείμενα.

Χρησιμοποιήσατε κάποιο όπλο ή εργαλείο; Αν ναι, τι συνέβη στο χαρακτήρα, όταν ήρθε σε επαφή μαζί του;

Στάδιο 3. Σχεδιάστε την επόμενη πίστα του παιχνιδιού σας

Με ποιο τρόπο αλλάζει πίστα ο χαρακτήρας; Εμπλέκεται κάποιο άλλο αντικείμενο; Υπάρχουν κάποιες προϋποθέσεις;

Στάδιο 4. Απαντήστε, όσο καλύτερα μπορείτε, στις παρακάτω ερωτήσεις:

Πόσα αντικείμενα έχει συνολικά το παιχνίδι σας; Πόσα είναι τα στιγμιότυπα του κάθε αντικειμένου;

Είναι ο χαρακτήρας σας αντικείμενο;

Ποια είναι τα βασικά **γεγονότα** που δίνουν ενδιαφέρον στο παιχνίδι σας;

Αν σας πει κάποιος ότι η αντίδραση σε ένα γεγονός είναι προγραμματισμένη ως *μέθοδος* μέσα σε κάποιο αντικείμενο, σε ποιο αντικείμενο θεωρείτε ότι ταιριάζει καλύτερα να ανήκει η μέθοδος που αντιστοιχεί σε κάθε γεγονός από αυτά που απαντήσατε στην προηγούμενη ερώτηση;

Είναι η αλλαγή πίστας γεγονός και αν ναι, με ποια αντικείμενα συνδέεται; Σε ποιο από αυτά θα ανήκει η αντίστοιχη μέθοδος;

Αν μια *κλάση* είναι ένα πρότυπο για την κατασκευή αντικειμένων, μπορείτε να εντοπίσετε κλάσεις στο περιβάλλον της μηχανής ανάπτυξης που χρησιμοποιήσατε;

Αν σας πει κάποιος ότι οι κλάσεις έχουν *κληρονομικότητα*, δηλαδή μεταφέρουν τα χαρακτηριστικά και τις μεθόδους τους στις κλάσεις και τα αντικείμενα που κατασκευάζονται από αυτές, αυτό σας βοηθά να εντοπίσετε κλάσεις και υποκλάσεις στο περιβάλλον αυτό;

Στάδιο 5. Παρουσιάστε το παιχνίδι σας στους συμμαθητές σας και συζητήστε τις απαντήσεις που δώσατε στις παραπάνω ερωτήσεις.

2.4 Ερωτήσεις

1. Ποιες είναι οι βασικές φάσεις για τη διαδικασία ανάπτυξης ενός συστήματος λογισμικού στο μοντέλο του καταρράκτη;
2. Ποια είναι τα βασικά προγραμματιστικά υποδείγματα;
3. Ποια είναι τα βασικά χαρακτηριστικά του δομημένου προγραμματισμού;
4. Ποια είναι τα βασικά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού;
5. Αναφέρετε μερικές σύγχρονες γλώσσες προγραμματισμού.

2.5 Αναφορές - Βιβλιογραφία

Εφαρμογές Πολυμέσων, Αβραντινής κ.ά., Βιβλίο Μαθητή ΕΠΑ.Λ. ΙΤΥΕ/ΔΙΟΦΑΝΤΟΣ

Προγραμματισμός Υπολογιστών, Σιδερίδης, κ.ά., Τ.Ε.Ε., Ο.Ε.Δ.Β. Κεφάλαιο 7, παράγραφο 3 (7.3) Είδη προγραμματισμού και κεφ. 17 Έλεγχος και εκσφαλμάτωση προγράμματος και Κεφ. 18. Τεκμηρίωση προγράμματος.

3

Βασικά στοιχεία γλώσσας προγραμματισμού

3. Βασικά στοιχεία γλώσσας προγραμματισμού

Στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- χρησιμοποιούμε βασικές εντολές της γλώσσας προγραμματισμού Python
- υλοποιούμε απλά προγράμματα στη γλώσσα προγραμματισμού Python
- χρησιμοποιούμε το ολοκληρωμένο προγραμματιστικό περιβάλλον της γλώσσας προγραμματισμού Python, ώστε να γράφουμε τον κώδικα ενός απλού προγράμματος, να το αποθηκεύουμε, να το εκτελούμε, να ελέγχουμε αν έχει σφάλματα και να κάνουμε αν χρειαστεί τις απαραίτητες διορθώσεις για να έχουμε το επιθυμητό αποτέλεσμα
- περιγράφουμε τι είναι μεταβλητή και πώς τη χρησιμοποιούμε στη γλώσσα προγραμματισμού Python
- περιγράφουμε τους βασικούς τύπους δεδομένων και να υλοποιούμε αριθμητικές και λογικές πράξεις
- αναγνωρίζουμε διαφορετικά είδη σφαλμάτων και να κάνουμε τις απαραίτητες ενέργειες για να τα διορθώσουμε.

Λέξεις κλειδιά

Προγραμματιστικό Περιβάλλον γλώσσας προγραμματισμού, βασικές λογικές και αριθμητικές πράξεις, τελεστές, τύποι δεδομένων, μεταβλητή, βασικές εντολές, έτοιμες συναρτήσεις, συγγραφή και επεξεργασία κώδικα, διερμηνευτής, έλεγχος προγράμματος.

Εισαγωγή

Στο κεφάλαιο αυτό θα έχουμε την ευκαιρία να γνωρίσουμε και να εξοικειωθούμε με τα βασικά χαρακτηριστικά της γλώσσας προγραμματισμού Python, μιας σύγχρονης γλώσσας προγραμματισμού, που χρησιμοποιείται για την ανάπτυξη πολλών προγραμμάτων, όπως απλά προγράμματα, εκπαιδευτικά παιχνίδια, σύνθετες εμπορικές εφαρμογές, πλατφόρμες κοινωνικής δικτύωσης. Είναι μια διερμηνευόμενη, υψηλού επιπέδου γλώσσα, απλή και εύκολη στην εκμάθησή της. Άλλες διαδεδομένες σύγχρονες γλώσσες υψηλού επιπέδου είναι η C++, η Java, η Perl, η Ruby, η Processing.

Οι γλώσσες υψηλού επιπέδου βασίζονται γενικά σε λέξεις κλειδιά, συνήθως της Αγγλικής γλώσσας και ακολουθούν μια τυπική αυστηρή γραμματική και συντακτικό που καθορίζονται κατά τη φάση σχεδιασμού της γλώσσας. Έχουν το πλεονέκτημα ότι είναι εύκολα κατανοητές από τον άνθρωπο-προγραμματιστή και χαρακτηρίζονται από μεγάλη *φορητότητα*, δηλαδή τα προγράμματα που φτιάχνονται σε αυτές τις γλώσσες μπορούν να εκτελεστούν σε διάφορα είδη υπολογιστών χωρίς καθόλου ή με μικρές μόνο τροποποιήσεις.

Το ολοκληρωμένο προγραμματιστικό περιβάλλον της γλώσσας Python, που θα χρησιμοποιείται στο μάθημα αυτό, διατίθεται μέσω Διαδικτύου και μπορούμε να το εγκαταστήσουμε στον υπολογιστή μας, καθώς αποτελεί Ελεύθερο Λογισμικό Ανοικτού Κώδικα. Η γλώσσα προγραμματισμού Python είναι πολύ διαδεδομένη, με μια μεγάλη κοινότητα προγραμματιστών να την υποστηρίζει, δίνοντας συμβουλές και υλικό ελεύθερα στο Διαδίκτυο. Επιπρόσθετα, περιέχει πλούσιες βιβλιοθήκες από έτοιμο κώδικα προγραμματισμού που μπορούμε να τον χρησιμοποιούμε στα προγράμματά μας και μεθοδολογία που μας επιτρέπει την εύκολη και γρήγορη συγγραφή σύνθετων προγραμμάτων.

Για να επικοινωνήσουμε με τον υπολογιστή με τη γλώσσα Python, δίνοντας τις κατάλληλες εντολές για να τις εκτελέσει, χρειαζόμαστε να μάθουμε ένα πολύ μικρό σύνολο λέξεων-ρημάτων -μοιάζουν με λέξεις της αγγλικής γλώσσας- που αντιστοιχούν στις εντολές, καθώς και τον τρόπο σύνταξής τους. Το πλεονέκτημα είναι ότι μόλις γράψουμε μια εντολή μπορούμε να ελέγξουμε το αποτέλεσμα, μια και το προγραμματιστικό περιβάλλον διερμηνεύει τις εντολές άμεσα και μας ειδοποιεί αν έχουμε κάνει κάποιο λάθος στη σύνταξή τους. Αυτή η αμεσότητα μάς δίνει τη δυνατότητα να μπορούμε να μαθαίνουμε μόνοι μας, χωρίς να διστάζουμε να πειραματιστούμε.

Ξεκινήστε έχοντας λίγη υπομονή στην αρχή. Προσέξτε ιδιαίτερα στη σύνταξη των εντολών καθώς είναι ιδιαίτερα αυστηρή. Δώστε την ανάλογη προσοχή στη λεπτομέρεια, χρησιμοποιήστε τον υπολογιστή. Μόλις μπείτε στο συναρπαστικό κόσμο του προγραμματισμού και αρχίζετε να μαθαίνετε να προγραμματίζετε, δείξτε και σε άλλους, συνεργαζόμενοι σε ομάδες. Όταν προσπαθείτε να επεξηγήσετε κάτι σε κάποιο συμμαθητή σας, θα βοηθήσει και σας να κατανοήσετε σε μεγαλύτερο βάθος κάποια σημεία που μπορεί να μην είχατε αρχικά προσέξει.

Διδακτικές ενότητες

3.1 Γνωριμία με το ολοκληρωμένο περιβάλλον ανάπτυξης της γλώσσας προγραμματισμού

Πριν προχωρήσουμε να μαθαίνουμε πώς να προγραμματίζουμε στη γλώσσα προγραμματισμού Python, είναι χρήσιμο να εξοικειωθούμε με το πώς αυτή δουλεύει. Έτσι, θα ασχοληθούμε στη συνέχεια με το:

- πώς να εγκαθιστάμε το ολοκληρωμένο προγραμματιστικό περιβάλλον της Python
- πώς να το χρησιμοποιούμε για να γράψουμε το πρώτο μας απλό πρόγραμμα και στη συνέχεια να το αποθηκεύσουμε.

Για να ξεκινήσουμε να χρησιμοποιούμε τη γλώσσα προγραμματισμού Python, χρειάζεται πρώτα, να μεταφορτώσουμε (κατεβάσουμε) από το Διαδίκτυο και να εγκαταστήσουμε στον υπολογιστή μας το προγραμματιστικό περιβάλλον της. Για τις ανάγκες του μαθήματος θα χρησιμοποιήσουμε την έκδοση 2.7.10 της Python και ο χρωματισμός του κώδικα ακολουθεί το IDLE της Python.

Το ολοκληρωμένο περιβάλλον ανάπτυξης προγραμμάτων IDLE (Integrated Development Environment) της Python είναι ένα δωρεάν πρόγραμμα που μπορούμε εύκολα να εγκαταστήσουμε στον υπολογιστή μας, κατεβάζοντας και εκτελώντας το κατάλληλο αρχείο, ανάλογα με το λειτουργικό σύστημα του υπολογιστή μας, μια και είναι διαθέσιμο για διάφορα λειτουργικά συστήματα, όπως MS Windows, Linux, MAC OS X.

3.1.1 Εγκατάσταση για λειτουργικό σύστημα Linux

Αρχικά ελέγχουμε ποια έκδοση είναι ήδη εγκατεστημένη, αφού σε αρκετές διανομές Linux το προγραμματιστικό περιβάλλον της Python είναι προεγκατεστημένο. Αν δεν είναι, τότε μπορούμε να χρησιμοποιήσουμε το αποθετήριο λογισμικού (Software Centre) για την εγκατάστασή του.

3.1.2 Εγκατάσταση IDLE Python για Microsoft Windows

Το προγραμματιστικό περιβάλλον **IDLE** είναι διαθέσιμο στον επίσημο δικτυακό τόπο υποστήριξης της γλώσσας Python https://www.python.org/. Αρχικά επιλέγουμε την έκδοση της γλώσσας (2.7.10), μεταφορτώνουμε το κατάλληλο αρχείο, ανάλογα με την έκδοση των Windows και το εγκαθιστάμε εκτελώντας το στον υπολογιστή και αποδεχόμενοι τους προτεινόμενους όρους. Στη συνέχεια ελέγχουμε αν έγινε σωστά η εγκατάσταση. Τέλος, ανοίγουμε το μενού επιλογών ως εξής: Έναρξη→ Όλα τα προγράμματα→ Python→ και επιλέγουμε IDLE (εικόνα 3.1).

Εικόνα 3-1. *Επιλογή IDLE για να ξεκινήσει η εκτέλεση του προγραμματιστικού περιβάλλοντος*

Επιλέγοντας IDLE με το ποντίκι, ξεκινάει η εκτέλεση του προγραμματιστικού περιβάλλοντος και ανοίγει ένα παράθυρο στο μέσο της οθόνης. Τώρα, μπορούμε άμεσα να ξεκινήσουμε να πληκτρολογούμε το πρώτο μας πρόγραμμα, αρκεί να γράψουμε την κατάλληλη εντολή δίπλα στα τρία σύμβολα >>>.

Το Python Shell είναι μέρος του ολοκληρωμένου περιβάλλοντος προγραμματισμού και ουσιαστικά επιτρέπει τη συγγραφή κάθε εντολής ξεχωριστά (η εντολή ακολουθεί μετά τα τρία σύμβολα >>>) και την άμεση εκτέλεσή της με τη βοήθεια του διερμηνευτή της γλώσσας.

Δραστηριότητα. Το πρώτο μας πρόγραμμα

Ας δοκιμάσουμε να δημιουργήσουμε το πρώτο μας πρόγραμμα με στόχο να εμφανιστεί στην οθόνη του υπολογιστή το μήνυμα χαιρετισμού: «Χαίρε, κόσμε!». Για το πρόγραμμά μας θα χρησιμοποιήσουμε την εντολή εμφάνισης **print**, μια εντολή εξόδου με τη δυνατότητα να εμφανιστεί στην έξοδο του υπολογιστή (οθόνη) ένα μήνυμα ή το αποτέλεσμα μιας πράξης.

```
>>> print "Χαίρε, κόσμε!"
Χαίρε, κόσμε!
```

Για την αποθήκευση της εργασίας μας στην Python, πρέπει να χρησιμοποιήσουμε από το οριζόντιο μενού, στο πάνω μέρος του παραθύρου, το File→Save As (Αρχείο, αποθήκευση ως) δίνοντας ένα όνομα στο πρώτο μας αυτό πρόγραμμα. Είναι χρήσιμο να φτιάξουμε ένα φάκελο, όπου εκεί θα αποθηκεύουμε όλα μας τα προγράμματα. Στο φάκελο μπορούμε για παράδειγμα να δώσουμε το όνομα «Ασκήσεις σε Python». Αν θέλουμε αργότερα να επεξεργαστούμε ή/και να εκτελέσουμε το πρόγραμμά μας από το μενού File επιλέγουμε άνοιγμα (Open), βρίσκουμε το όνομα του αρχείου και το επιλέγουμε. Η Python δίνει στα προγράμματα την κατάληξη .py.

Όταν θα γράφουμε μεγαλύτερα προγράμματα θα χρησιμοποιούμε τον επεξεργαστή κώδικα (editor) που μας προσφέρει το προγραμματιστικό περιβάλλον IDLE για σύνταξη κώδικα και επεξεργασία. Η διαδικασία έχει ως εξής:

- από το μενού επιλογών File→ Άνοιγμα νέου αρχείου (New File)
- σύνταξη του κώδικα του προγράμματος
- αποθήκευση του προγράμματος (Save, όνομα αρχείου με κατάληξη .py)
- εκτέλεση του προγράμματος από την επιλογή Run→Run Module (ή πατάμε το πλήκτρο F5), εμφάνιση των αποτελεσμάτων στο Python Shell.

3.2 Μεταβλητές και τύποι δεδομένων

3.2.1 Τύποι δεδομένων

Το πρόγραμμα "Χαίρε_Κόσμε" ήταν αρκετά απλό. Στην πραγματικότητα ένα πρόγραμμα επεξεργάζεται δεδομένα τα οποία μπορεί να είναι αποθηκευμένα στη μνήμη του υπολογιστή, στο σκληρό δίσκο ή σε κάποιο άλλο αποθηκευτικό μέσο, ή η εισαγωγή τους γίνεται από το πληκτρολόγιο ή κάποια άλλη συσκευή εισόδου (για παράδειγμα barcode reader). Οι τύποι δεδομένων προσδιορίζουν τον τρόπο παράστασης των δεδομένων εσωτερικά στον υπολογιστή καθώς και το είδος της επεξεργασίας τους από τον υπολογιστή. Στην Python δε δηλώνουμε ρητά τι τύπο δεδομένων χρησιμοποιούμε.

Στην Python χαρακτηριστικοί τύποι δεδομένων είναι **οι αριθμοί, οι λογικοί (booleans) και οι συμβολοσειρές** (ή αλφαριθμητικά strings)

Οι **αριθμοί** στην Python είναι κυρίως τριών τύπων: **οι ακέραιοι** (integers), **οι αριθμοί κινητής υποδιαστολής** (floating point) **και οι μιγαδικοί αριθμοί** (complex numbers) (τον τύπο αυτόν των αριθμών απλά τον αναφέρουμε και δεν θα μας απασχολήσει στη Β' τάξη).

Παραδείγματα

- Ο αριθμός 3 αποτελεί παράδειγμα ακέραιου (και είναι απλά ένας ακέραιος αριθμός).
- Οι 3.14 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 $28.2 * 10^{-5}$.
- Ο αριθμός (-2+3j) αποτελεί παράδειγμα μιγαδικού αριθμού.

Ο **λογικός τύπος** (boolean) έχει μόνο δύο τιμές, την τιμή True (Αληθής) και τη τιμή False (Ψευδής) και έχει σκοπό την καταγρα-φή του αποτελέσματος ενός ελέγχου.

Οι **συμβολοσειρές** είναι μια ακολουθία από χαρακτήρες και μπορεί να αποτελείται από περισσότερες από μία λέξεις. Παράδειγμα "Καλημέρα σε όλους". Οι λέξεις μπορούν να είναι στην Ελληνική Γλώσσα, στην Αγγλική ή σε κάθε γλώσσα που υποστηρίζεται από το πρότυπο Unicode. Μπορούμε να ορίσουμε μια συμβολοσειρά με μονά εισαγωγικά, για παράδειγμα 'Σήμερα είναι μία ηλιόλουστη μέρα!' ή με διπλά εισαγωγικά "Σήμερα είναι μια ηλιόλουστη μέρα!", αλλά όχι ανάμικτα. Με ό,τι ξεκινάμε θα πρέπει πάλι να κλείνουμε.

Όπως θα δούμε και σε επόμενη ενότητα ανάμεσα στις τιμές κάθε τύπου δεδομένων μπορούμε να κάνουμε διάφορες **πράξεις** χρησιμοποιώντας τους αντίστοιχους τελεστές (σύμβολα). Για παράδειγμα 34 + 56 ή 3.14*8.

3.2.2 Μεταβλητές

Αρκετές φορές, σε ένα πρόγραμμα απαιτείται να αποθηκεύσουμε προσωρινά κάποια δεδομένα στη μνήμη του υπολογιστή. Για το σκοπό αυτό χρησιμοποιούμε τις **μεταβλητές**.

Δραστηριότητα εμπέδωσης

Μπορούμε να παρομοιάσουμε τις μεταβλητές, με τις θήκες για τις κάρτες σε ένα πορτοφόλι. Σε κάθε θήκη μπορούμε να βάλουμε προσωρινά μία κάρτα. Στη συνέχεια μπορούμε να αντικαταστήσουμε την κάρτα με μία άλλη. Κάθε φορά όμως μπορούμε

να βάλουμε σε κάθε θήκη μόνο μία κάρτα. Οι κάρτες μπορούν να είναι διαφορετικών τύπων. Κάρτες επαγγελματικές, κάρτες πιστωτικές, κάρτες supermarket. Κάθε τύπος κάρτας έχει διαφορετικές τιμές. Έτσι στον τύπο καρτών "επαγγελματικές" μπορούμε να έχουμε τις τιμές υδραυλικός, κομμωτήριο, φωτοτυπίες κ.ά.

Παρόμοια, οι μεταβλητές στον προγραμματισμό αντιστοιχούν σε μία θέση μνήμης του υπολογιστή. Κάθε φορά, στη θέση αυτή μπορεί να αποθηκευτεί μόνο μία τρέχουσα τιμή. Οι μεταβλητές μπορούν να παίρνουν τιμές από διάφορους τύπους δεδομένων. Με τον όρο τιμή εννοούμε μια ακολουθία από bit (0,1) η οποία ερμηνεύεται σύμφωνα με κάποιον τύπο δεδομένων. Είναι δυνατό η ίδια ακολουθία από bits να έχει διαφορετική ερμηνεία ανάλογα με τον τύπο δεδομένων του οποίου ερμηνεύεται. Οι μεταβλητές χρησιμεύουν, ώστε εύκολα να μπορούμε να έχουμε πρόσβαση στο περιεχόμενό τους. Το περιεχόμενο αυτό βρίσκεται προσωρινά αποθηκευμένο στη θέση μνήμης του υπολογιστή που έχει δεσμευτεί για τη μεταβλητή αυτή.

Η γλώσσα Python παρέχει εντυπωσιακές εναλλακτικές δυνατότητες έκφρασης για τη διαχείριση μεταβλητών που διευκολύνουν τον προγραμματιστή. Για τη χρησιμοποίηση μιας μεταβλητής **δεν απαιτείται η δήλωσή της**, ενώ μπορεί να εκχωρήσουμε διαφορετικούς τύπους τιμών, όπως ακέραιες, κινητής υποδιαστολής, συμβολοσειρές.

3.2.3 Εκχώρηση τιμής σε μεταβλητή και εμφάνιση του περιεχομένου της

Για να χρησιμοποιήσουμε μια μεταβλητή, χρειαζόμαστε να της δώσουμε ένα όνομα και στη συνέχεια να της εκχωρήσουμε κάποια τιμή. Στη γλώσσα Python υπάρχουν ορισμένοι κανόνες που πρέπει να ακολουθήσουμε για να δώσουμε ένα όνομα σε μια μεταβλητή. Συνηθίζουμε να δίνουμε ένα όνομα σχετικό με το είδος της μεταβλητής με λατινικούς χαρακτήρες, που μπορεί να συνοδεύεται από κάποιον αριθμό. Για παράδειγμα, onoma1, onoma_2, timi, mesi_timi, embado κ.ά. Στην Python δεν επιτρέπεται να ξεκινάμε το όνομα μιας μεταβλητής με αριθμό, ενώ θα πρέπει να είμαστε βέβαιοι ότι το όνομα δε είναι όμοιο με κάποιο όνομα ενσωματωμένης συνάρτησης ή εντολής.

Για να εκχωρήσουμε μια τιμή σε μεταβλητή χρησιμοποιούμε το σύμβολο « = ». Προσοχή! το σύμβολο αυτό δε είναι το ίδιο με το σύμβολο της ισότητας που χρησιμοποιούμε στα μαθηματικά. Το «=» σημαίνει ότι δίνουμε στη μεταβλητή μια τιμή ενός τύπου δεδομένων που είναι αποθηκευμένη σε μια θέση μνήμης. Η μεταβλητή λειτουργεί ως αναφορά στην τιμή αυτή. Για παράδειγμα a=234 σημαίνει ότι η μεταβλητή a αναφέρεται στην τιμή 234 (ακέραιος αριθμός) που είναι αποθηκευμένη στη μνήμη του υπολογιστή.

Αν θέλουμε να εμφανίσουμε την τιμή της μεταβλητής, τότε μπορούμε να χρησιμοποιήσουμε την εντολή print μαζί με το όνομα της μεταβλητής (για το προηγούμενο παράδειγμα η εντολή print a θα εμφανίσει στην οθόνη 234).

Δραστηριότητα εμπέδωσης

Πειραματιστείτε με τα παρακάτω παραδείγματα:

```
>>> x = 10    # εκχωρεί την ακέραια τιμή 10 στο x
>>> print x   # εμφανίζει το περιεχόμενο της x
10
>>> x=x+15
# προσθέτει στο περιεχόμενο της x τον ακέραιο 15 και το αποτέλεσμα το εκχωρεί ξανά στη x
>>> print x
25
>>> x=x *0.1
# πολλαπλασιάζει στο νέο περιεχόμενο της x τον αριθμό κινητής υποδιαστολής
```

```
0.1 και το # αποτέλεσμα, που είναι πλέον αριθμός κινητής υποδιαστολής, το εκ-
χωρεί ξανά στη x.
>>> print x
2.5
>>> print x*100
250.0
>>> onoma = "Μυρτώ"     # εκχωρεί την τιμή της συμβολοσειράς Μυρτώ στη
μεταβλητή onoma
>>> print onoma
Μυρτώ
>>> metavliti1=metavliti2=metavliti3=15
# πολλαπλή εκχώρηση της τιμής 15 σε τρεις  μεταβλητές
>>> x, y = 10, 18    # ανταλλαγή τιμών των x, y
>>> print x,y
10 20
>>> x, y, z = 10, 20, "Μάγια"    # εκχωρεί την τιμή 10 στη x, την τιμή 20 στη y και
την τιμή Μάγια στη z.


>>> print x,y,z
10 20 Μάγια
>>> print 'Καλημέρα ' + z
Καλημέρα Μάγια
```

3.2.4 Δραστηριότητες - Ασκήσεις

Άσκηση 1

Έστω ότι έχουμε τη μεταβλητή x με περιεχόμενο τον ακέραιο αριθμό 12 και τη μεταβλητή y με περιεχόμενο τον ακέραιο αριθμό 20. Χρησιμοποιώντας ένα υπολογιστικό φύλλο, αντιστοιχήστε τη μεταβλητή x στο κελί με διεύθυνση B(2) και εκχωρήστε (στο κελί) την τρέχουσα ακέραια τιμή της 12 (περιεχόμενο). Παρόμοια, αντιστοιχήστε τη μεταβλητή y στο κελί με διεύθυνση B(3) και εκχωρήστε (στο κελί) την τρέχουσα ακέραια τιμή 20 (περιεχόμενο). Προσπαθήστε να αντιμεταθέσετε τις τιμές αυτές, ώστε η μεταβλητή x, στη θέση B(2), να πάρει τη τιμή της μεταβλητής y και η μεταβλητή y, στη θέση B(3), την τιμή της μεταβλητής x.

Στη συνέχεια περιγράψτε, με βήματα, τη διαδικασία με φυσική γλώσσα. Προσπαθήστε να γράψετε τις κατάλληλες εντολές εκχώρησης σε γλώσσα Python, ώστε να δώσετε τις αρχικές τιμές στις μεταβλητές x, y και στη συνέχεια να αντιμεταθέσετε τις τιμές τους.

Συζητήστε στην τάξη, αναλύοντας τα συμπεράσματά σας, για το βασικό τρόπο που λειτουργούν οι μεταβλητές σε σχέση με τη μνήμη του υπολογιστή. Πρόσθετες πληροφορίες, για να τεκμηριώσετε τις απόψεις σας, μπορείτε να βρείτε στο Διαδίκτυο.

Άσκηση 2

Ακολουθήστε τα παρακάτω στο περιβάλλον της Python:

- Να γράψετε μια απλή γραμμή σχολίων της αρεσκείας σας (προσοχή να ξεκινάει με #).
- Να οριστεί η μεταβλητή arithmos, που να περιέχει την τιμή 1.234 (αριθμός κινητής υποδιαστολής).
- Να οριστεί η μεταβλητή arithmos_tetragwno, που να περιέχει το τετράγωνο της arithmos.
- Να εμφανιστούν στην οθόνη οι τιμές των μεταβλητών arithmos, arithmos_tetragwno.
- Να οριστεί η μεταβλητή logikh που να περιέχει την τιμή True.
- Να οριστεί η μεταβλητή paixnidia που να περιέχει την τιμή 10.
- Να γίνουν λογικοί έλεγχοι για το περιεχόμενο της μεταβλητής paixnidia, με χρήση των τελεστών σύγκρισης σε συνδυασμό με τους τελεστές λογικών πράξεων,

όπως για παράδειγμα (paixnidia == 8, paixnidia > 8, paixnidia == 8 or paixnidia == 10, paixnidia == 9 and paixnidia == 10).

Επαληθεύστε τις προσπάθειες σας με τις παρακάτω απαντήσεις.

```
>>> arithmos = 1.234
>>> arithmos_tetragwno = arithmos ** 2
>>> print arithmos, arithmos_tetragwno
Αποτέλεσμα στην οθόνη: 1.234 1.522756
>>> logikh = True
>>> print logikh
 Αποτέλεσμα στην οθόνη: True
>>> paixnidia = 10
>>> paixnidia == 8
False
>>> paixnidia == 8 or paixnidia == 10
True
>>> not paixnidia == 8
True
>>> paixnidia >= 8
True
>>> paixnidia != 8
True
>>> paixnidia <= 8
False
```

Συζητήστε στην τάξη για τη χρήση των λογικών πράξεων not, or και and, καθώς και για τους λογικούς τελεστές σύγκρισης. Φτιάξτε έναν πίνακα με όλους τους αριθμητικούς τελεστές, τους λογικούς τελεστές σύγκρισης και τους τελεστές λογικών πράξεων, σημειώνοντας, εν συντομία και δίπλα από κάθε τελεστή, τι κάνει ο καθένας.

Δραστηριότητα 1. Αυτοαξιολόγησης

Η δραστηριότητα αυτή σκοπεύει σε αυτοαξιολόγηση, σχετικά με τους τύπους δεδομένων και τις μεταβλητές.

Δραστηριότητα 2. Φύλλο αυτοαξιολόγησης

Καταγράψετε στη δεξιά στήλη του πίνακα τι πιστεύετε ότι θα εμφανιστεί στην οθόνη, μετά την εκτέλεση των παρακάτω προγραμμάτων. Στη συνέχεια, επαληθεύστε τις απαντήσεις σας, εκτελώντας τα προγράμματα μέσα από το περιβάλλον της γλώσσας Python.

Προγράμματα	Αποτελέσματα στην οθόνη
x = 45 x = 45.5 **print** x print x + x	
x= "Μυρτώ" y = "Βασίλη" **print** "Καλημέρα", y, "και", x **print** "Καλημέρα", x + y	
x = 12 **print** x + 3	
x = 26 x = y = z = 23 **print** x, y, z x = y = "Ελευθερία" print x, y	

x , y, z = "Ελευθερία", "Πέτρος", 2 **print** x, y, z **print** x, y * z	
x , y = 8, 12 **print** x, y	
Για συζήτηση στην τάξη a = 010 **print** a	
Για συζήτηση στην τάξη a = 0xA **print** a	

3.3 Βασικές εντολές, τελεστές, αριθμητικές και λογικές πράξεις

Στην παράγραφο αυτή παραθέτουμε συνοπτικά τις βασικές εντολές, τους τελεστές και τις αριθμητικές και λογικές πράξεις της γλώσσας Python.

Αριθμητικοί τελεστές (Arithmetic operations): +, -, *, /, **, %

Λογικοί τελεστές σύγκρισης: ==, !=, <, >, <=, >=

Τελεστές λογικών πράξεων: not, or, and, με τις ακόλουθες λογικές λειτουργίες

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

Δραστηριότητα. Εμπέδωση βασικών στοιχείων ενότητας

Συμπληρώστε κατάλληλα τον παρακάτω πίνακα.

Στην αριστερή στήλη παρουσιάζεται μια πράξη που πρέπει να εκτελεστεί στον υπολογιστή χρησιμοποιώντας το προγραμματιστικό περιβάλλον της γλώσσας Python.

- Αρχικά να συμπληρωθεί η μεσαία στήλη με τα αποτελέσματα που πιστεύετε ότι θα εμφανιστούν στην οθόνη του υπολογιστή μετά την εκτέλεση της πράξης.
- Να γίνει επαλήθευση των αποτελεσμάτων που συμπληρώθηκαν πληκτρολογώντας και εκτελώντας κάθε πράξη ξεχωριστά μέσα στο περιβάλλον της γλώσσας Python.

Πράξεις	Αναμενόμενο αποτέλεσμα	Αποτέλεσμα στην οθόνη
2 + 3		
3560 – 130		
5 * 3 + 2		
5 * (3 + 2)		
5 * (3 + 2) / 10		
3 * 3.75 / 1.5		
5 ** 2		
3 ** 4		
.....		

Ερωτήσεις εμπέδωσης ενότητας

Από τα παραπάνω, ποιοι πιστεύετε ότι είναι οι βασικοί κανόνες για την προτεραιότητα των πράξεων στη γλώσσα Python; Σημειώστε τις απαντήσεις στις κενές γραμμές που ακολουθούν.

- .....
- .....
- .....

Ποιοι οι βασικοί αριθμητικοί τελεστές που χρησιμοποιούνται στην Python; Ποια πράξη κάνει ο καθένας από αυτούς; Σημειώστε τις απαντήσεις σας στις κενές γραμμές που ακολουθούν.

- .....
- .....
- .....

3.3.1 Βασικές εντολές

Στις προηγούμενες παραγράφους παρουσιάστηκαν μερικές από τις βασικές εντολές της γλώσσας Python.

Για την εμφάνιση τιμών στην οθόνη του υπολογιστή χρησιμοποιούμε την εντολή print, με ποικίλες μορφές, όπως: **print** όνομα_μεταβλητής, ή print αριθμός, ή print ΄συμβολοσειρά΄, ή print onoma_μεταβλητής (τελεστής) αριθμός

Για την **εκχώρηση τιμής σε μια μεταβλητή** χρησιμοποιούμε το «=», με μορφή: ονομα_μεταβλητής=τιμή μεταβλητής.

Σχετικά με την **εισαγωγή τιμής σε μια μεταβλητή από το πληκτρολόγιο** -κατάσταση όπου αναμένει από το χρήστη να εισάγει μια τιμή από το πληκτρολόγιο- η οποία αποδίδεται αυτόματα στη μεταβλητή, χρησιμοποιούμε δύο βασικές συναρτήσεις: την input() και την *raw_**input***().

Όνομα_μεταβλητής= **input()**

```
x = input("Δώσε έναν αριθμό: " )
```

Αν θέλουμε να εισάγουμε ένα αλφαριθμητικό, χρησιμοποιούμε την εντολή raw_input:

```
name = raw_input("Δώσε το όνομά σου : ")
```

Γενικά, ό,τι εισάγεται με τη raw_input θεωρείται αυτόματα αλφαριθμητικό, ενώ η input προσπαθεί να το υπολογίσει. Για παράδειγμα αν δώσουμε στην input το όνομα μια μεταβλητής θα επιστρέψει το περιεχόμενο της μεταβλητής.

Για την **εισαγωγή σχολίων** -κατάσταση όπου μπορούμε να εισάγουμε επεξηγηματικά σχόλια στο πρόγραμμά μας- θέτουμε μπροστά από το κείμενο, το σύμβολο **#**. Με αυτό τον τρόπο όταν κάποιος δει το πρόγραμμά μας, θα καταλάβει πιο εύκολα τι ακριβώς κάνει και πώς σκεφτήκαμε να το φτιάξουμε.

3.4 Βασικές συναρτήσεις (ενσωματωμένες)

Η Python παρέχει μια ποικιλία **ενσωματωμένων συναρτήσεων** οι οποίες μετατρέπουν τιμές από έναν τύπο σε έναν άλλο.

Δραστηριότητα εμπέδωσης

Στο περιβάλλον της γλώσσας Python, επαληθεύστε τα παρακάτω:

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

```
>>> float(10)
10.0
>>> int(5.678)
5
>>> abs(-45)
45
>>> divmod (10,3)
(3, 1)
>>> pow (2,3)
8
```

Αρκετές φορές συνηθίζουμε, αν θέλουμε να διαβάσουμε από το πληκτρολόγιο έναν ακέραιο αριθμό μαζί με τη συνάρτηση input(), να χρησιμοποιούμε και τη συνάρτηση int()

```
a=int(input('Δώσε ένα αριθμό :'))
```

Παράδειγμα

```
>>> a=int(input('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 ενότητας

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

Η γλώσσα Python διαθέτει μια Μαθηματική μονάδα λογισμικού (math module) η οποία περιέχει τις περισσότερο γνωστές μαθηματικές συναρτήσεις. Μια **μονάδα** ή **άρθρωμα λογισμικού** (module) είναι ένα αρχείο το οποίο περιέχει μια συλλογή από σχετικές συναρτήσεις. Προτού χρησιμοποιήσουμε μια μονάδα, πρέπει να την εισάγουμε: >>> import math.

Για να έχουμε πρόσβαση σε μια από τις συναρτήσεις, θα πρέπει να προσδιορίσουμε το όνομα της μονάδας και το όνομα της συνάρτησης χωρισμένα με μία τελεία. Αυτή η μορφή ονομάζεται **συμβολισμός με τελεία** (dot notation). Ας δούμε ένα παράδειγμα για τη συνάρτηση 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
```

Εκτός από τις ενσωματωμένες βιβλιοθήκες (μονάδες) συναρτήσεων που περιλαμβάνονται στη γλώσσα Python, μπορεί κανείς να βρει στους δικτυακούς τόπους υποστήριξης της γλώσσας και εξωτερικές μονάδες λογισμικού με πληθώρα επιπλέον συναρτήσεων για τη δημιουργία ποικίλων προγραμμάτων. Χαρακτηριστικό παράδειγμα είναι τα projects για τη δημιουργία γραφικών και παιχνιδιών με ένα σύνολο πρόσθετων συναρτήσεων και έτοιμου λογισμικού. Στο επόμενο κεφάλαιο θα μάθουμε να δημιουργούμε τις δικές μας συναρτήσεις. Σε επόμενο κεφάλαιο θα δούμε επίσης, διάφορα παραδείγματα από έτοιμες βιβλιοθήκες λογισμικού, για να υλοποιούμε ελκυστικά προγράμματα στο χρήστη.

3.5 Δομή προγράμματος και καλές πρακτικές

Στη δραστηριότητα που ακολουθεί και στη δεξιά στήλη του πίνακα, εμφανίζεται μια κλασική δομή ενός προγράμματος σε Python. Παρότι δεν είναι αναγκαίο, είναι ιδιαίτερα χρήσιμο με τη μορφή σχολίων, όπου ξεκινάμε με το σύμβολο #, να δίνουμε ένα χαρακτηριστικό τίτλο στο πρόγραμμα και να προσθέτουμε, όπου κρίνουμε χρήσιμο, επεξηγηματικά σχόλια μέσα στον κώδικα. Χρειάζεται ιδιαίτερη προσοχή στα κενά διαστήματα πριν τις εντολές, καθώς, όπως θα δούμε στο επόμενο κεφάλαιο, η Python τα χρησιμοποιεί για να ορίσει ομάδες εντολών. Επιλέγουμε τους κατάλληλους τελεστές και χρησιμοποιούμε τα ίδια εισαγωγικά, δηλαδή μονά εισαγωγικά με μονά και διπλά εισαγωγικά με διπλά.

Δραστηριότητα. Υλοποίηση απλού προγράμματος σε Python

Να γραφεί αλγόριθμος που να υπολογίζει και να εκτυπώνει το εμβαδό τριγώνου βάσης 10 και ύψους 15. Το εμβαδό δίνεται από τον τύπο Ε= (β*υ)/2. Στη συνέχεια να γραφεί πρόγραμμα που να υλοποιεί τον αλγόριθμο εμβαδό τριγώνου σε γλώσσα Python.

Ψευδοκώδικας
Αλγόριθμος Εμβαδό_Τριγώνου ΒΑΣΗ← 10 ΥΨΟΣ← 15 ΕΜΒΑΔΟ← (ΒΑΣΗ*ΥΨΟΣ)/2 Εκτύπωσε «ΤΟ ΕΜΒΑΔΟ ΤΟΥ ΤΡΙΓΩΝΟΥ ΕΙΝΑΙ:», ΕΜΒΑΔΟ ΤΕΛΟΣ Εμβαδό_Τριγώνου
Πρόγραμμα σε Python
ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΕΜΒΑΔΟΥ ΤΡΙΓΩΝΟΥ BASH = 10 YPSOS = 15 EMBADO = (BASH * YPSOS)/ 2 **print** 'ΤΟ ΕΜΒΑΔΟ ΤΟΥ ΤΡΙΓΩΝΟΥ ΕΙΝΑΙ: ', EMBADO Αποτέλεσμα στην οθόνη: ΤΟ ΕΜΒΑΔΟ ΤΟΥ ΤΡΙΓΩΝΟΥ ΕΙΝΑΙ: 75

Να γραφεί αλγόριθμος και το αντίστοιχο πρόγραμμα σε γλώσσα Python, που να υπολογίζει το εμβαδό τετραγώνου πλευράς α. Το πρόγραμμα να διαβάζει από το πληκτρολόγιο το μήκος της πλευράς α, ζητώντας από το χρήστη να το πληκτρολογήσει. Στη συνέχεια να υπολογίζει το εμβαδόν του τετραγώνου και να εμφανίζει το αποτέλεσμα στην οθόνη με το ανάλογο μήνυμα.

3.6 Διαδικασία συγγραφής, μετάφρασης και εκτέλεσης προγράμματος

3.6.1 Διερμηνέας και μεταγλωττιστής

Όταν γράψουμε κώδικα σε μορφή κειμένου, για να μπορέσει να εκτελεστεί στον υπολογιστή, θα πρέπει να μετατραπεί σε γλώσσα μηχανής που είναι κατανοητή από την Κεντρική Μονάδα Επεξεργασίας (CPU) του υπολογιστή. Τα προγράμματα που μετατρέπουν τις εντολές μας, μπορούν να χωριστούν σε δύο κατηγορίες:

- στους **μεταγλωττιστές** (compilers) και
- στους **διερμηνείς** (interpreters).

Ένας διερμηνευτής διαβάζει και ελέγχει μία εντολή τη φορά, την εκτελεί και μετά προχωράει στην επόμενη. Ένας μεταγλωττιστής διαβάζει ολόκληρο το πρόγραμμα και το μεταφράζει, πριν ξεκινήσει η εκτέλεσή του. Σε αυτό το πλαίσιο, το πρόγραμμα υψηλού επιπέδου ονομάζεται **πηγαίος κώδικας** (source code) και στη γενική περίπτωση, το μεταφρασμένο πρόγραμμα ονομάζεται **εκτελέσιμο** (executable). Όταν ένα πρόγραμμα μεταγλωττιστεί, μπορεί να εκτελεστεί επανειλημμένα, χωρίς περαιτέρω μετάφραση.

Η Python αποτελεί γλώσσα με δυνατότητα τα προγράμματά της να εκτελούνται από διερμηνευτή. Υπάρχουν δύο τρόποι χρήσης του διερμηνευτή: **διαδραστική λειτουργία** (interactive mode) και **σεναριακή λειτουργία** (script mode). Στη διαδραστική λειτουργία, πληκτρολογούμε προγράμματα σε Python και ο διερμηνευτής εμφανίζει το αποτέλεσμα:

```
>>> 15 + 1
16
```

Το σύμβολο >>> είναι ο **προτροπέας** (prompt) που χρησιμοποιεί ο διερμηνευτής για να υποδείξει ότι είναι έτοιμος. Όταν πληκτρολογήσουμε 15 + 1, ο διερμηνευτής ελέγχει την έκφραση, τη μεταφράζει ώστε να εκτελεστεί και στην οθόνη εμφανίζεται το αποτέλεσμα 16. Εναλλακτικά, μπορούμε να αποθηκεύσουμε κώδικα σε ένα φάκελο και να χρησιμοποιήσουμε το διερμηνευτή για να εκτελέσει τα περιεχόμενα του φακέλου. Η διαδικασία αυτή ονομάζεται **σενάριο**.

3.6.2 Είδη σφαλμάτων στον προγραμματισμό

Σε ένα πρόγραμμα μπορούν να συμβούν διαφόρων ειδών σφάλματα και είναι χρήσιμο να γίνει διάκριση μεταξύ τους, προκειμένου να μπορούμε να τα εντοπίσουμε γρηγορότερα:

- Τα **συντακτικά λάθη**, που παράγονται από την Python όταν διερμηνεύει τον πηγαίο κώδικα. Συνήθως, υποδεικνύουν ότι υπάρχει κάποιο λάθος στη σύνταξη της εντολής. Σε αυτή την περίπτωση ο διερμηνευτής εμφανίζει κατάλληλο διαγνωστικό μήνυμα, το οποίο διευκολύνει τον εντοπισμό και τη διόρθωση του συντακτικού λάθους, όπως φαίνεται παρακάτω:

```
>>> print "81" + 19
1.41421356237
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print "2" + 3
TypeError: cannot concatenate 'str' and 'int' objects
>>> print "1" + "007"
1007
>>> print 1 + 007
8
```

- Τα σφάλματα **χρόνου εκτέλεσης** που παράγονται από το διερμηνευτή, αν πάει κάτι στραβά κατά την εκτέλεση του προγράμματος. Τα περισσότερα μηνύματα αυτών των σφαλμάτων περιέχουν πληροφορίες σχετικά με το πού συνέβη το σφάλμα και τι συναρτήσεις εκτελούνταν. Παράδειγμα: εξαντλήθηκε η μνήμη, δεν έγινε σωστός χειρισμός και απαιτείται άπειρος χρόνος κ.ά.
- Τα **σημασιολογικά ή λογικά σφάλματα** που αποτελούν προβλήματα σε ένα πρόγραμμα το οποίο τρέχει χωρίς να παράγει κάποιο μήνυμα λάθους, αλλά δεν κάνει αυτό που θα έπρεπε. Αυτή είναι η δυσκολότερη κατηγορία σφαλμάτων, καθώς πρέπει να διατρέξουμε πολλές φορές όλο το πρόγραμμα, γραμμή προς γραμμή, για να καταλάβουμε που έχει γίνει λογικό λάθος. Παράδειγμα: το αποτέλεσμα από τον υπολογισμό μιας σειράς πράξεων δεν είναι το αναμενόμενο, καθώς αποδόθηκε λάθος η σειρά των πράξεων.

Μερικά συνηθισμένα **συντακτικά** λάθη, που χρειάζεται να προσέχουμε:

- Κεφαλαία αντί μικρά γράμματα. Η Python πολλές φορές ξεχωρίζει τα κεφαλαία γράμματα από τα μικρά και δεν τα θεωρεί ως ίδια λέξη. Για παράδειγμα, αν γράψουμε την εντολή Print με κεφαλαίο, δε θα καταλάβει ότι είναι η εντολή print.
- Δεν πρέπει να μπερδεύουμε διπλά εισαγωγικά με μονά. Όταν ανοίγουμε εισαγωγικά, πρέπει να κλείνουμε με τα όμοιά τους (μονά με μονά, διπλά με διπλά).
- Δεν πρέπει να μπερδεύουμε την κάτω _ με τη μεσαία - παύλα.
- Να αποφεύγουμε να χρησιμοποιούμε ελληνικούς χαρακτήρες με λατινικούς στα ονόματα μεταβλητών.
- Όταν ανοίγουμε παρενθέσεις, πρέπει να τις κλείνουμε με το αντίστοιχο σύμβολο (),{},[].
- Ιδιαίτερη προσοχή απαιτείται στα κενά διαστήματα στην αρχή μιας γραμμής, μια και στην Python τα κενά διαστήματα έχουν σημασία.
- Ελέγχουμε την ορθότητα της σύνταξης κάθε εντολής, καθώς συχνά δεν γράφουμε την εντολή σωστά.
- Πρέπει να δίνουμε ιδιαίτερη προσοχή στην περίπτωση που μεταφέρουμε έτοιμο κώδικα από διαφορετικές εκδόσεις της Python, όπως από τη 2 στην 3, διότι μεταξύ των εκδόσεων υπάρχουν διαφορές σε ορισμένες εντολές, ως προς τη σύνταξη.

3.7 Ερωτήσεις

1. Πώς ονομάζεται το προγραμματιστικό περιβάλλον της γλώσσας Python;
2. Τι είναι η μεταβλητή στον προγραμματισμό;
3. Ποιους βασικούς τύπους αριθμών υποστηρίζει η γλώσσα Python;
4. Τι τιμές μπορούν να πάρουν οι λογικοί (boolean) τύποι δεδομένων;
5. Ποια εντολή της γλώσσας Python χρησιμοποιούμε για να εμφανίσουμε ένα μήνυμα στην οθόνη του υπολογιστή;
6. Ποιο σύμβολο χρησιμοποιούμε για να δώσουμε μια τιμή σε μια μεταβλητή;
7. Ποια βασική λειτουργία εκτελεί ο διερμηνευτής της γλώσσας Python;
8. Τι διαφορά έχει ο διερμηνευτής από το μεταγλωττιστή σε μία γλώσσα προγραμματισμού;
9. Ποια είδη σφαλμάτων μπορεί να συναντήσουμε σε ένα πρόγραμμα;

3.8 Αναφορές - Βιβλιογραφία

Downey, A. (2012) «Think Python, How to think like a computer scientist», O' Reilly. http://www.greenteapress.com/thinkPython/. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά στα πλαίσια πτυχιακής εργασίας στο ΤΕΙ Λάρισας και είναι διαθέσιμο στο Διαδίκτυο.

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 International License. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από την Ελληνική κοινότητα του Ubuntu. http://ubuntu-gr.org/ και είναι διαθέσιμο στο Διαδίκτυο.

Αβούρης Ν., Κουκιάς Μ., Παλιουράς Β., Σγάρμπας Κ. (2013) «Εισαγωγή στους υπολογιστές με τη γλώσσα Python», Εκδόσεις Πανεπιστημίου Πατρών, Πάτρα.

Δικτυακός κόμβος υποστήριξης με πλούσιο υλικό πολυμέσων για τη διδασκαλία της γλώσσας Python, http://www.Pythonschool.net/ (τελευταία προσπέλαση 09/07/2015).

Δικτυακός κόμβος υποστήριξης της γλώσσας Python, https://www.Python.org/ (τελευταία προσπέλαση 10/07/2015)

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 Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ. ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π.), Ιανουάριος 2015.

Σχετικό υλικό και χρήσιμοι σύνδεσμοι από την ελεύθερη εγκυκλοπαίδεια Βικιπαίδεια, http://el.wikipedia.org/wiki/Python (τελευταία προσπέλαση 4/07/2015).

4

Αλγοριθμικές δομές

4. Αλγοριθμικές δομές

Στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- επιλέγουμε και να χρησιμοποιούμε βασικές αλγοριθμικές δομές, όπως την ακολουθία, την επιλογή, την επανάληψη.
- υλοποιούμε δεδομένους απλούς αλγόριθμους σε προγράμματα στη γλώσσα προγραμματισμού Python.
- ππεξηγούμε τη λειτουργία τμήματος κώδικα της γλώσσας προγραμματισμού Python.
- ορίζουμε τις δικές μας συναρτήσεις και να τις αξιοποιούμε για την επίλυση προβλημάτων.

Λέξεις κλειδιά

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

Εισαγωγή

Στο κεφάλαιο αυτό θα έχουμε την ευκαιρία να εξοικειωθούμε με τις αλγοριθμικές δομές και να υλοποιήσουμε πιο σύνθετα προγράμματα, χρησιμοποιώντας τις αντίστοιχες δομές στη γλώσσα προγραμματισμού Python. Ένα πρόγραμμα λαμβάνει δεδομένα ως είσοδο, τα επεξεργάζεται σύμφωνα με τις εντολές που περιέχει και στη συνέχεια επιστρέφει τα αποτελέσματα. Μπορεί δηλαδή να παρομοιαστεί με μια συνταγή μαγειρέματος, όπου αρχικά ζητάει τα βασικά υλικά, τα επεξεργάζεται σύμφωνα με τη συνταγή και επιστρέφει το μαγειρεμένο αποτέλεσμα. Έτσι, τα βασικά χαρακτηριστικά ενός προγράμματος είναι:

Είσοδος: Περιλαμβάνει τα δεδομένα που απαιτεί το πρόγραμμα από το περιβάλλον, ώστε να παράγει την επιθυμητή έξοδο. Η είσοδος γίνεται συνήθως από περιφερειακές συσκευές, όπως ποντίκι, πληκτρολόγιο, αναγνώστη barcode, σαρωτή ή κάποιο αποθηκευτικό μέσο, όπως σκληρό δίσκο, usb flash memory, DVD, χρησιμοποιώντας την κατάλληλη εντολή. Στο προηγούμενο κεφάλαιο γνωρίσαμε τη συνάρτηση input(), ώστε να διαβάζει το πρόγραμμα δεδομένα από το πληκτρολόγιο .

Τρόπος Εκτέλεσης: Ο τρόπος με τον οποίο γίνεται η επεξεργασία των δεδομένων. εμπεριέχει μεταξύ άλλων τις ροές εκτέλεσης (ακολουθία, επιλογή, επανάληψη), τα

μοντέλα υπολογισμού, τις συναρτήσεις, τις μεταβλητές. Στο προηγούμενο κεφάλαιο πειραματιστήκαμε με τις μεταβλητές, τις βασικές πράξεις και τις ενσωματωμένες συναρτήσεις, οπότε στο κεφάλαιο αυτό θα επεκταθούμε στις ροές εκτέλεσης, ενώ θα γνωρίσουμε περισσότερα για τις συναρτήσεις.

Έξοδος: Περιλαμβάνει τα τελικά αποτελέσματα που παράγονται και συνήθως εμφανίζονται ή τυπώνονται για το χρήστη, με την έξοδο να πραγματοποιείται συνήθως στην οθόνη ή στον εκτυπωτή. Μπορεί να έχει τη μορφή κειμένου/αριθμού ή και γραφικών, όπως για παράδειγμα ένα πρόγραμμα που δημιουργεί ένα παιχνίδι. Ήδη, γνωρίσαμε στο προηγούμενο κεφάλαιο, τη βασική εντολή εξόδου της γλώσσας προγραμματισμού Python: την εντολή **print**.

Διδακτικές ενότητες

4.1 Αλγοριθμικές δομές - Ροές εκτέλεσης προγράμματος

4.1.1 Ακολουθία

Πρόκειται για μια σειρά από εντολές που εκτελούνται η μία μετά την άλλη, ώστε να δοθεί στην έξοδο ένα επιθυμητό αποτέλεσμα.

Δραστηριότητα. Δομή ακολουθίας

Να γραφεί πρόγραμμα που να επιλύει το νόμο του Ωμ (Ohm) και να υπολογίζει την ένταση του ρεύματος I για τάση V = 220 Volt και R = 25 Ohm.

```
# Πρόγραμμα εύρεσης έντασης του ηλεκτρικού ρεύματος
V = 220
R = 25
I = V / R
print " Η ένταση του ρεύματος είναι:", I, " A"
```

4.1.2 Δομή επιλογής if

Αν θέλουμε να εκτελεστεί μια ακολουθία εντολών, μόνον εφόσον πληρείται μία συγκεκριμένη συνθήκη, τότε χρησιμοποιούμε τη δομή επιλογής if (ΑΝ) με τη συνθήκη την οποία θέλουμε να ελέγξουμε. **Αν η συνθήκη είναι αληθής, τότε** το σύνολο των εντολών που περιέχονται στη δομή if, θα εκτελεστεί, αλλιώς η ροή του προγράμματος θα προσπεράσει τη δομή if και θα συνεχίσει από το τέλος της if.

```
if  <συνθήκη ελέγχου>:
   # γράψε τις εντολές της if εδώ
# εντολές
```

Παράδειγμα

```
# πρόγραμμα εμφάνισης της απόλυτης τιμής ενός ακεραίου αριθμού
a=int(input('Δώσε ένα ακέραιο αριθμό '))
if a<=0:
    a=(-1)*a
print a
```

Αν ανάλογα με την αποτίμηση μιας συνθήκης θέλουμε να εκτελεστούν διαφορετικές εντολές, τότε μπορούμε να χρησιμοποιήσουμε τη **δομή επιλογής if…else (ΑΝ…ΑΛΛΙΩΣ)**. Αν ισχύει η συνθήκη (True), θα εκτελεστεί το μπλοκ εντολών της if, αλλιώς, αν δεν ισχύει (False), θα εκτελεστεί το μπλοκ εντολών της else.

Η εντολή ελέγχου ΑΝ_ΑΛΛΙΩΣ συντάσσεται ως:

```
if <συνθήκη ελέγχου>:
        #γράψε τις εντολές εδώ
else:
        #γράψε τις εντολές εδώ
```

Σημείωση: Οι ομάδες εντολών που θα εκτελεστούν, αν ισχύει η συνθήκη, ορίζονται ως ένα μπλοκ με εσοχή (κενά διαστήματα) βάζοντας τη μία εντολή κάτω από την άλλη. Δεν πρέπει να διαγράψουμε τα κενά αυτά διαστήματα.

Η Python προσφέρει τη δυνατότητα για σύνταξη σύνθετων δομών επιλογής με τη χρήση της εντολής elif. Η σύνταξη είναι ως εξής:

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

Δραστηριότητες ενότητας

Δραστηριότητα 1. Δομή επιλογής if … else…

Να γραφεί πρόγραμμα που να διαβάζει την ηλικία ενός ατόμου. Αν η ηλικία είναι μεγαλύτερη ή ίση των 18 ετών να εμφανίζει μήνυμα ότι επιτρέπεται να ψηφίζει, διαφορετικά να εμφανίζει μήνυμα ότι είναι αρκετά νέος ακόμα.

Ψευδοκώδικας

```
Αλγόριθμος ΗΛΙΚΙΑ
 Εκτύπωσε «Δώσε την ηλικία σου:»
Διάβασε age
Αν age >= 18 Τότε
    Εκτύπωσε «Είσαι αρκετά μεγάλος και μπορείς να ψηφίσεις στις εθνικές εκλογές»
Αλλιώς
    Εκτύπωσε «Είσαι αρκετά μικρός ακόμα για να ψηφίσεις στις εθνικές εκλογές»
Τέλος_Αν
Τέλος ΗΛΙΚΙΑ
```

Πρόγραμμα σε Python

```
age =  input ("Δώσε την ηλικία σου: ")
if age >= 18:
   print"Είσαι αρκετά μεγάλος και μπορείς να ψηφίσεις στις εθνικές εκλογές"
else:
   print "Είσαι αρκετά μικρός ακόμα για να ψηφίσεις στις εθνικές εκλογές"
```

Δραστηριότητα 2. Δομή επιλογής - Εμφωλευμένο if … else …

Α) Μελετήστε το παρακάτω πρόγραμμα.

- Τι πιστεύετε ότι κάνει το πρόγραμμα;
- Τι θα τυπώσει το παρακάτω πρόγραμμα για τους βαθμούς 6, 10, 12, 15, 18, 20;
- Τι θα συμβεί, αν, κατά λάθος, βάλουμε ως βαθμό το 25; Πληκτρολογήστε το πρόγραμμα και επαληθεύστε τις απαντήσεις σας.

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

```
# Εισαγωγή βαθμού από το χρήστη
test = input('Παρακαλώ δώσε το βαθμό που πήρες από το τεστ:')
if test<=20:
# Εμφωλευμένο if
   if test>= 18:
      print 'Μπράβο τα πήγες πολύ καλά'
      # για 18 <= test <= 20
    else:
        if test >= 15:
            print 'Τα πήγες αρκετά καλά!'
            # για 15<= test < 18`
        else:
            if test >= 10:
```

```
            print"Τα πήγες σχετικά καλά."
            # για 10<=τεστ<15
        else:
            print "Θα πρέπει να διαβάσεις ξανά το κεφάλαιο"
            # για τεστ<10
else:
    # για test >20
    print "Ο βαθμός που έδωσες είναι πάνω από είκοσι"
```

Δραστηριότητα 3. Δομή επιλογής - Έλεγχος θερμοστάτη για ψύξη, θέρμανση

Να γραφεί πρόγραμμα που να διαβάζει τη θερμοκρασία ενός δωματίου και, αν η θερμοκρασία είναι ίση ή μεγαλύτερη από 32 βαθμούς, να εμφανίζει μήνυμα για να ανοίξει η ψύξη του κλιματιστικού, αν είναι μικρότερη ή ίση με 16, να εμφανίζει μήνυμα για να ανοίξει η θέρμανση, διαφορετικά να εμφανίζει μήνυμα ότι δεν χρειάζεται να ανοίξει το κλιματιστικό.

```
temperature = int(input("Ποια είναι η θερμοκρασία του δωματίου σε βαθμούς Κελσίου; "))
if temperature >= 32:
   print "Άνοιξε την ψύξη του κλιματιστικού και ρύθμισε τη θερμοκρασία"
elif temperature <= 16:
    print "Άνοιξε τη θέρμανση του κλιματιστικού και ρύθμισε για οικονομία τη θερμοκρασία"
else:
   print "Η θερμοκρασία είναι καλή δε χρειάζεται ακόμη να ανοίξεις το κλιματιστικό"
print "τέλος"
```

Δραστηριότητα 4. Δομή επιλογής

Χωρίς να πληκτρολογήσετε το παρακάτω πρόγραμμα βρείτε τι κάνει και δώστε, με μια πρόταση, ένα χαρακτηριστικό τίτλο.

```
number = input("Δώστε έναν ακέραιο αριθμό: ")
if number < 0:
   print "Ο αριθμός είναι αρνητικός"
elif number > 0:
   print "ο αριθμός είναι θετικός"
else:
   print "ο αριθμός είναι 0"
```

4.1.3 Δομή επανάληψης (for και while)

Συχνά, ορισμένοι υπολογισμοί σε ένα πρόγραμμα είναι αναγκαίο να εκτελούνται περισσότερες από μία φορές. Υπάρχουν δύο τύποι επαναλήψεων:

- Οι προκαθορισμένοι, όπου το πλήθος των επαναλήψεων είναι δεδομένο πριν αρχίσουν οι επαναλήψεις.
- Οι μη προκαθορισμένοι ή απροσδιόριστοι, όπου το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών του σώματος της επανάληψης.

Ένας *βρόχος* είναι μια ακολουθία εντολών οι οποίες δηλώνονται μία φορά, αλλά μπορούν να εκτελεστούν πολλές διαφορετικές φορές. Το τμήμα κώδικα μέσα στο βρόχο θα εκτελείται για έναν καθορισμένο αριθμό επαναλήψεων ή για όσο ισχύει μία συνθήκη. Κατ' αυτό τον τρόπο, έχουμε και το διαχωρισμό σε **for** και **while** βρόχους αντίστοιχα, που υποστηρίζει η γλώσσα Python.

Οι βρόχοι for εκτελούνται για συγκεκριμένο πλήθος επαναλήψεων. Για τη δημιουργία τους χρησιμοποιείται η συνάρτηση **range**().

```
for onoma_metavlitis in range (αρχή, μέχρι, βήμα):
  Εντολή1
  Εντολή2
  …..
  Εντολήν
```

Δραστηριότητα. Εξοικείωση με τη δομή επανάληψης for

Να γραφεί πρόγραμμα που να εκτυπώνει πέντε (5) φορές το μήνυμα «Θέλει αρετή και τόλμη η ελευθερία!»

```
for metritis in range (5):
  print "Θέλει αρετή και τόλμη η ελευθερία!"
```

4.1.3.1 Προσεγγίζοντας τη συνάρτηση range

Η range() είναι μια ενσωματωμένη συνάρτηση της γλώσσας Python, η οποία, ανάμεσα σε άλλα, χρησιμοποιείται για την υπόδειξη του αριθμού των επαναλήψεων που θα εκτελεστούν σε ένα βρόχο.

Η δομή της συνάρτησης range είναι της μορφής (αρχή, μέχρι, βήμα), όπου αρχή, μέχρι, βήμα, ακέραιοι αριθμοί. Οι ενδείξεις της αρχής και του βήματος δεν είναι υποχρεωτικές, αλλά μπορεί να προστεθούν, ώστε η επανάληψη να μοιάζει με τη σύνταξη του ψευδοκώδικα. Αντίθετα η ένδειξη «μέχρι», πρέπει πάντα να αναφέρεται.

Δραστηριότητα 1

Πειραματιστείτε με τη χρήση της συνάρτησης range στο προγραμματιστικό περιβάλλον της Python για να δείτε τι παράγει η συνάρτηση. Για παράδειγμα:

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(0, 35, 5), παράγει τη λίστα: [0,5,10,15,20,25,30]

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

Δραστηριότητα 2. Δομή Επανάληψης με for

Εξασκηθείτε στη δομή επανάληψης for μέσω των παρακάτω προβλημάτων

α) Να γραφεί αλγόριθμος και το αντίστοιχο πρόγραμμα σε γλώσσα Python που να διαβάζει πέντε τυχαίους ακεραίους αριθμούς και να υπολογίζει το άθροισμά τους.

Ψευδοκώδικας	Πρόγραμμα σε Python
Αλγόριθμος Άθροισμα sum← 0 Για i από 1 μέχρι 5 Διάβασε x sum← sum+x Τέλος_επανάληψης Εκτύπωσε sum Τέλος Άθροισμα	sum = 0 for i in range(5): x = int(input("Δώσε έναν αριθμό: ")) sum = x + sum print 'Το αποτέλεσμα είναι ', sum

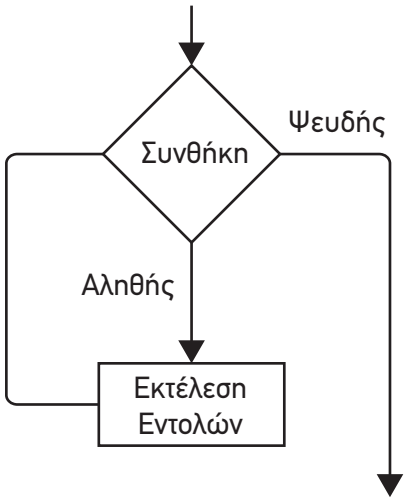
β) Να γραφεί αλγόριθμος και το αντίστοιχο πρόγραμμα σε γλώσσα Python που να υπολογίζει το άθροισμα 1+2+3+...+100

Ψευδοκώδικας	Πρόγραμμα σε Python
Αλγόριθμος αθροίζω sum← 0 Για i από 1 μέχρι 100 sum← sum+i Τέλος_επανάληψης Εκτύπωσε sum Τέλος αθροίζω	# Πρόγραμμα Αθροίζω sum = 0 for i in range(1, 101): sum = sum + i print sum

γ) Να γραφεί αλγόριθμος και το αντίστοιχο πρόγραμμα σε Python που να υπολογίζει την παράσταση: 1+1*2+1*2*3+…+1*2*3*…*v

4.1.4 Δομή επανάληψης με βρόχο while

Ο βρόχος **while** (ή Όσο <συνθήκη> επανάλαβε) χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων, όπου υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου, με τον έλεγχο της συνθήκης να πραγματοποιείται πριν από την εκτέλεση των εντολών του βρόχου.



Αρχική τιμή μεταβλητής
while ονομα_μεταβλητής <συνθήκη>:
  Εντολή1
  Εντολή2
  ….
  Εντολήν

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

Σημείωση2: Θα πρέπει πριν το βρόχο while, αρχικά, να δώσουμε μια τιμή στη μεταβλητή που ελέγχει τη συνθήκη του βρόχου, ώστε αυτός να εκτελεστεί ή όχι.

Δραστηριότητες ενότητας

Δραστηριότητα 1. Δομή Επανάληψης με while

Εξασκηθείτε στη δομή επανάληψης while μέσω των παρακάτω προβλημάτων

α) Να γραφεί, με χρήση του βρόχου while, ο αντίστοιχος αλγόριθμος και το αντίστοιχο πρόγραμμα της δραστηριότητας 2, της προηγούμενης παραγράφου.

Ψευδοκώδικας

```
Αλγόριθμος αθροίζω
sum← 0
i← 1
Όσο i<=100 επανάλαβε
  sum← sum+i
  i← i+1
Τέλος_επανάληψης
Εκτύπωσε sum
Τέλος αθροίζω
```

Πρόγραμμα σε Python

```
# Πρόγραμμα Αθροίζω
sum = 0        # αρχική τιμή στο άθροισμα
i=1            # αρχική τιμή στη μεταβλητή ελέγχου
while i <= 100 :      # έλεγχος της επανάληψης
    sum = sum + i
    i= i + 1          # αύξηση του μετρητή
print sum
```

β) Μια συνήθης εφαρμογή του while βρόχου είναι να ελέγχει την εγκυρότητα των δεδομένων εισόδου από το χρήστη. Συμπλήρωσε κατάλληλα την παρακάτω συνθήκη, έτσι ώστε να εξασφαλίζεται ότι ο χρήστης θα εισάγει τελικά την τιμή 'ναι' ή 'όχι'.

```
choice = raw_input('Σας ενδιαφέρει ο προγραμματισμός? (ναι/όχι)')
while _________:
    choice = raw_input("Παρακαλώ δώστε έγκυρη τιμή ")
```

Δραστηριότητα 2. Δομή Επανάληψης - Δημιουργία quiz

Να γραφεί πρόγραμμα για την εύρεση της απάντησης σε μια ερώτηση πολλαπλής επιλογής, με τέσσερις απαντήσεις 1, 2, 3 και 4. Το πρόγραμμα θα δίνει αρχικά την ερώτηση και τις πιθανές απαντήσεις και θα ζητάει από το χρήστη να πληκτρολογήσει τον αριθμό της απάντησης που θεωρεί σωστή. Αν η απάντηση είναι λάθος θα εμφανίζει μήνυμα λάθους και θα ζητάει ξανά να δοθεί η απάντηση. Η διαδικασία θα συνεχίζεται όσο η απάντηση δεν είναι σωστή. Κάθε αποτυχημένη προσπάθεια θα αποθηκεύεται σε ένα μετρητή. Όταν απαντηθεί σωστά η ερώτηση, θα εμφανίζεται ανάλογο μήνυμα μαζί με τις συνολικές προσπάθειες.

```
# Πρόγραμμα quiz
print "Που έζησε ο συγγραφέας και ποιητής Γιάννης Σκαρίμπας;"
print  "1. ΑΘΗΝΑ"
print "2. ΠΑΤΡΑ "
print "3. ΧΑΛΚΙΔΑ"
print "4. ΚΑΣΤΟΡΙΑ"
metriti = 0
reply = 0
while  reply != 3:
    reply = input ("Βρες τη σωστή απάντηση ")
    if reply != 3:
        print "Πρέπει να ξαναδοκιμάσεις. "

        metriti = metriti + 1

print " ΜΠΡΑΒΟ ΑΠΑΝΤΗΣΕΣ ΣΩΣΤΑ ΜΕ ΤΗΝ", metriti, "η ΠΡΟΣΠΑΘΕΙΑ"
```

Δραστηριότητα 3. Δομή Επανάληψης

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

```
thenum = random.randint(1,100)
print "Έχω σκεφτεί ένα αριθμό από το 1 μέχρι το 100."
print "Μπορείς να τον μαντέψεις;"
guess = 0

while guess != thenum:
```

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

4.2 Συναρτήσεις

Η έννοια των συναρτήσεων αποτελεί ένα από τα πιο σημαντικά δομικά στοιχεία ενός προγράμματος σε όλες τις γλώσσες προγραμματισμού. Οι συναρτήσεις μπορεί να είναι είτε έτοιμες από τη γλώσσα προγραμματισμού, είτε να δημιουργούνται από εμάς. Στην ενότητα αυτή θα προσεγγίσουμε διάφορα χαρακτηριστικά των συναρτήσεων.

4.2.1 Δημιουργώντας τις δικές μας συναρτήσεις

Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Μας επιτρέπουν να δίνουμε ένα όνομα σε ένα σύνολο εντολών και να το εκτελούμε καλώντας το όνομά τους, οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε. Αυτή η διαδικασία ονομάζεται *κλήση* (call) της συνάρτησης. Στα παραπάνω προγράμματα χρησιμοποιήσαμε αρκετές ενσωματωμένες συναρτήσεις, όπως την int() και τη range.

Οι συναρτήσεις ορίζονται χρησιμοποιώντας τη χαρακτηριστική λέξη **def**, από το define που σημαίνει ορίζω. Στη συνέχεια ακολουθεί ένα όνομα που ταυτοποιεί την εκάστοτε συνάρτηση. Μετά προσθέτουμε ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν μερικά ονόματα μεταβλητών και η γραμμή τελειώνει με διπλή τελεία (:).

Ας δούμε ένα απλό παράδειγμα:

```
def tragouda ():
    print 'Υλαγιαλή! Υλαγιαλή! Υλαγιαλή! Μέσα στη νύχτα με πηγαίνουν οι ανέμοι'

# Τέλος της συνάρτησης
tragouda() # κλήση της συνάρτησης
tragouda() # κι άλλη μία κλήση της συνάρτησης
```

Στο παραπάνω παράδειγμα ορίσαμε μια συνάρτηση με το όνομα tragouda σύμφωνα με τη σύνταξη που περιγράφηκε παραπάνω. Η συνάρτηση tragouda δεν έχει παραμέτρους, γι' αυτό δε δηλώνονται καθόλου μεταβλητές ανάμεσα στις παρενθέσεις. Οι παράμετροι στις συναρτήσεις είναι απλά η είσοδος στη συνάρτηση, ώστε να περνάμε διαφορετικές τιμές στη συνάρτηση και να παίρνουμε αντίστοιχα αποτελέσματα. Στη συνέχεια καλούμε την ίδια συνάρτηση δύο φορές, για να δείξουμε ότι δε χρειάζεται πλέον να γράφουμε τον ίδιο κώδικα δύο φορές. Αν θέλουμε μπορούμε να φτιάξουμε άλλη μια συνάρτηση **epanalave_tragouda**(), που να καλεί την tragouda δύο φορές. Σκεφτείτε τι θα εμφανιστεί στην οθόνη όταν εκτελεστούν τα παρακάτω. Δημιουργήστε μια ακόμα συνάρτηση που να καλεί και χρησιμοποιεί κατάλληλα την **epanalave_tragouda**(), ώστε να εμφανίζει τους στίχους 4 φορές.

```
def epanalave_tragouda():
    tragouda()
    tragouda()
# τέλος συνάρτησης
epanalave_tragouda()

def epanalave_4fores():
    epanalave_tragouda()
    epanalave_tragouda()
# τέλος συνάρτησης
epanalave_4fores()
```

4.2.2 Οι εσοχές

Οι εσοχές στην αρχή των εντολών είναι πολύ σημαντικές στην Python. Ο κενός χώρος πριν από μια εντολή και γενικότερα η στοίχιση των εντολών δεν είναι μόνο θέμα αισθητικής, όπως σε άλλες γλώσσες, αλλά θέμα ουσίας που μπορεί να αλλάξει το αποτέλεσμα του προγράμματος, όπως φαίνεται παρακάτω:

```
>>> def print2():
        print "********"
        print "********"
>>> print2()
********
********
```

```
>>> def print2():
        print "********"
      print "********"
********
>>> print2()
********
```

Στη δεύτερη περίπτωση, στη δεξιά στήλη, η δεύτερη εντολή print βρίσκεται έξω από τη συνάρτηση και εκτελείται κανονικά. Η συνάρτηση εμφανίζει μόνο μια γραμμή με αστέρια και όχι δύο, όπως θέλουμε. Δεν υπάρχει δηλαδή κάποια ειδική εντολή που να υποδηλώνει το τέλος του μπλοκ εντολών της συνάρτησης, όπως π.χ. τέλος_συνάρτησης, end_def. Όλα εξαρτώνται από τη στοίχιση των εντολών, **άρα πρέπει να είμαστε ιδιαίτερα προσεκτικοί με τη στοίχιση των εντολών** και την εσοχή πριν από κάθε εντολή, ώστε να εξασφαλίσουμε ότι ανήκει στο σωστό μπλοκ.

4.2.3 Ορισμός Συνάρτησης

Όπως είδαμε ο ορισμός συναρτήσεων στην Python είναι αρκετά απλός:

```
def <όνομα συνάρτησης> ( [ {λίστα παραμέτρων} ] ):
    εντολές
  [ return <αποτέλεσμα> ]
```

Οι αγκύλες [ ] σημαίνουν πως ό,τι περικλείεται μέσα σε αυτές είναι προαιρετικό. Η λίστα των παραμέτρων μπορεί να είναι κενή. Μια συνάρτηση δεν είναι υποχρεωτικό να επιστρέφει κάποια τιμή.

Δραστηριότητα

Ορίστε τις δύο συναρτήσεις, με ονόματα add και times3, όπως φαίνεται παρακάτω και πειραματιστείτε με τα διαφορετικά παραδείγματα κλήσης τους στο περιβάλλον της Python:

```
def add(arg1, arg2):
result = arg1+arg2
return result
def times3(arg):
    ginomeno = 3*arg
    return ginomeno
```

```
>>> add(10, 18)
28
>>> add(10, 18.5)
28.5
>>> times3(10)
30
```

Μπορούμε να συνδυάσουμε την κλήση συναρτήσεων, με το σκεπτικό ότι το αποτέλεσμα της μιας συνάρτησης μπορεί να αποτελέσει τα δεδομένα εισόδου μιας άλλης.

```
>>> times3(2.5)
7.5
>>> times3('python')
'python python python'
>>> times3( times3( 9 ') )
'9 9 9 9 9 9 9 9 9 '
>>> times3(add(add('ab','ba'),' '))
'abba  abba  abba'
```

Παρατηρήστε ότι έχουμε ορίσει μία συνάρτηση, η οποία δέχεται όλους τους τύπους των ορισμάτων και η λειτουργία της αναπροσαρμόζεται δυναμικά ανάλογα με αυτά και αν δοθούν αριθμοί τους προσθέτει, ενώ αν δοθούν αλφαριθμητικά, τα συνενώνει.

4.2.4 Παράμετροι συναρτήσεων

Μια συνάρτηση δέχεται δεδομένα μέσω των **παραμέτρων** και επιστρέφει τα αποτέλεσματα μέσω άλλων ή και των ίδιων παραμέτρων στο πρόγραμμα ή σε άλλη 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 τιμες ονομάζονται *ορίσματα*.

Κάποιες από τις ενσωματωμένες συναρτήσεις, που έχουμε ήδη συναντήσει, δεν απαιτούν ορίσματα, όπως για παράδειγμα, όταν καλούμε την math.pi, όπου δεν έχουμε κάποιο όρισμα. Σε άλλες όμως συναρτήσεις απαιτούνται ένα ή και περισσότερα ορίσματα, όπως στην math.pow όπου απαιτούνται δύο, ένα για τη βάση και ένα για τον εκθέτη.

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

Σημείωση: Η μεταβίβαση παραμέτρων στην Python λειτουργεί με τέτοιο τρόπο, ώστε οποιαδήποτε αλλαγή στις παραμέτρους εντός της συνάρτησης δεν έχει καμία επίδραση στα ορίσματα-μεταβλητές που έχουν οριστεί εκτός της συνάρτησης, όπως φαίνεται παρακάτω:

```
def increment(a, b):
    a = a + 1
    b = b + 1
    return a+b
# τέλος συνάρτησης
z = 1
w = 2
q = increment( z, w )
print z, w, q
1 2 5
```

Τα ορίσματα z, w δεν αλλάζουν τιμή, παρόλο που οι αντίστοιχοι παράμετροι αυξάνονται εντός της συνάρτησης increment, όπως φαίνεται και από την τιμή που επιστρέφει η συνάρτηση και καταχωρείται στην q. Περισσότερα για την εμβέλεια των παραμέτρων θα δούμε στη Γ' τάξη.

4.2.5 Δραστηριότητες ενότητας

Δραστηριότητα 1. Συναρτήσεις

Δημιουργήστε ένα νέο αρχείο στην Python, στο οποίο θα προσθέσετε τους ορισμούς των συναρτήσεων που δίνονται παρακάτω.

Ορίστε τη συνάρτηση printPython3, ώστε να εμφανίζει τη λέξη Python τρεις (3) φορές. Στη συνέχεια ορίστε τη συνάρτηση printPython9, ώστε να εμφανίζει τη λέξη Python εννέα (9) φορές.

```
def printPython3():
    print 'Python'
    print 'Python'
    print 'Python'
```

```
def printPython9():
    print 'Python'
    print 'Python'
    print 'Python'
    print 'Python'
    print 'Python'
    print 'Python'
    print 'Python'
    print 'Python'
    print 'Python'
```

- Πώς μπορούμε να γράψουμε τη συνάρτηση printPython9, χρησιμοποιώντας λιγότερες εντολές;

- Πώς μπορούμε να ορίσουμε μια συνάρτηση η οποία να εμφανίζει 21 φορές τη λέξη Python, με αποκλειστική χρήση των συναρτήσεων printPython3() και printPython9().

Δραστηριότητα 2. Συναρτήσεις (η ανάγκη για γενίκευση)

Δίνονται οι παρακάτω συναρτήσεις σε Python.

```
def printPython3():
    for i in range(3):
        print 'Python'
def printPython12():
    for i in range(12):
        print 'Python'
def printPython100():
    for i in range(100):
        print 'Python'
```

Συμπληρώστε κατάλληλα τον ορισμό της παρακάτω συνάρτησης printPython ώστε να εμφανίζει τη λέξη Python N φορές και στη συνέχεια δώστε τις διπλανές κλήσεις στο διερμηνευτή:

```
def printPython( _____ ):
    for i in range( N ):
        print 'Python'
```

```
>>> printPython(3)
>>> printPython(9)
>>> printPython(21)
>>> printPython(100)
```

Ως προς τι διαφέρει η τελευταία συνάρτηση από όλες τις προηγούμενες;

Ποια είναι η σχέση της με αυτές;

Δραστηριότητα 3. Συναρτήσεις (Δευτεροβάθμια Εξίσωση)

Να ορίσετε συνάρτηση για τον υπολογισμό της διακρίνουσας $\Delta = \beta^2 - 4\alpha\gamma$ της εξίσωσης $\alpha x^2 + \beta x + \gamma = 0$, και στη συνέχεια, να αναπτύξετε πρόγραμμα για την επίλυσή της.

Δραστηριότητα 4. Συναρτήσεις

Ανοίξτε ένα νέο αρχείο στην Python για να προσθέσετε τους ορισμούς των συναρτήσεων που δίνονται παρακάτω. Συνεχίστε καλώντας τις συναρτήσεις όπως:

```
# Δημιουργία της συνάρτησης
def add(arg1, arg2):
    result = arg1+arg2
    return result
def times3(arg):
    result = 3*arg
    return result
# τέλος ορισμού συναρτήσεων
add(10, 18)
add(10, 18.5)
add('Python ', '3.4')
times3(100)
times3(2.5)
times3('Python')
```

Μελετήστε τα αποτελέσματα και καταγράψτε τα συμπεράσματά σας για τις συναρτήσεις, τους αριθμητικούς τύπους και τα αλφαριθμητικά.

4.3 Δραστηριότητες κεφαλαίου

Δραστηριότητα 1

Να γραφεί πρόγραμμα σε γλώσσα Python που να διαβάζει τρεις ακεραίους αριθμούς και να υπολογίζει το μέσο όρο τους. Το αποτέλεσμα να εμφανίζεται στην οθόνη.

Δραστηριότητα 2

Να γραφεί πρόγραμμα σε γλώσσα Python που να διαβάζει το μή-κος της ακτίνας r ενός κύκλου και να υπολογίζει τη διάμετρο (2*r), την περιφέρεια (2*π*r) και το εμβαδό του κύκλου (π*r2). Τα αποτελέσματα να εμφανίζονται στην οθόνη με σχετικό μήνυμα, που να επεξηγεί τι είναι ο κάθε αριθμός που εμφανίζεται, όπως για παράδειγμα το *Εμβαδόν του κύκλου είναι: Χ.*

Δραστηριότητα 3

Να γραφεί πρόγραμμα σε γλώσσα Python που να επιλύει την εξίσωση Ax+B=0, για Α και Β πραγματικούς αριθμούς.

Δραστηριότητα 4

Σε ένα ηλεκτρικό κύκλωμα υπάρχουν δύο αντιστάσεις R1 και R2. Θέλουμε να βρούμε την ολική αντίσταση R του κυκλώματος. Υπάρχουν δύο τρόποι σύνδεσης αυτών των αντιστάσεων: σε σειρά ή παράλληλα. Στην πρώτη περίπτωση, ο τύπος που δίνει την ολική αντίσταση είναι R=R1+R2. Στην παράλληλη σύνδεση, η ολική αντίσταση βρίσκεται από τον τύπο 1/R=(1/R1)+(1/R2).

Να γραφεί πρόγραμμα σε γλώσσα Python που να:

- διαβάζει τις τιμές των αντιστάσεων R1 και R2
- διαβάζει τον τρόπο σύνδεσης, δίνοντας την επιλογή 1 για σύνδεση σε σειρά ή τη 2 για παράλληλη σύνδεση, των αντιστάσεων R1 και R2
- υπολογίζει, εφαρμόζοντας τον κατάλληλο τύπο ανάλογα με τον τρόπο σύνδεσης, την τιμή της ολικής αντίστασης R
- εμφανίζει τις τιμές των αντιστάσεων R1, R2, καθώς και την τιμή της αντίστασης R.

Δραστηριότητα 5

Να γραφεί πρόγραμμα σε γλώσσα Python που να βρίσκει σε ποιον όρο το άθροισμα 1+2+3+4+… γίνεται μεγαλύτερο του 2000.

Δραστηριότητα 6

Να γραφεί πρόγραμμα σε γλώσσα Python που να δέχεται τριψήφιο θετικό ακέραιο και να ελέγχει αν είναι *παλίνδρομος*, δηλαδή αριθμός που παραμένει ο ίδιος, όταν αναστρέψουμε τη σειρά των ψηφίων του, όπως για παράδειγμα ο 131 ή ο 797.

Δραστηριότητα 7

Να γραφεί πρόγραμμα σε γλώσσα Python που υλοποιεί τον υπολογισμό της ακολουθίας Fibonacci.

Δραστηριότητα 8

Να γραφεί πρόγραμμα σε γλώσσα Python που να υλοποιεί τον υπολογισμό της δύναμης χωρίς χρήση της συνάρτησης pow.

Δραστηριότητα 9

Να γραφεί πρόγραμμα σε γλώσσα Python που να εξετάζει και να εμφανίζει πόσοι από τους αριθμούς που υπάρχουν μεταξύ του 50 και του 500 είναι πολλαπλάσια του 3.

Δραστηριότητα 10

Να ορίσετε κατάλληλα μια συνάρτηση *παραγοντικό*, που να υπολογίζει και να τυπώνει το N! ενός θετικού αριθμού N.

N!= 1, αν N= 0

N!=1*2*3*….(N-1)*N, αν N>0

Στη συνέχεια να γραφεί πρόγραμμα που να διαβάζει έναν ακέραιο θετικό αριθμό α, να γίνεται έλεγχος ώσπου να δοθεί αριθμός >=0 και στη συνέχεια, καλώντας τη συνάρτηση *παραγοντικό* να υπολογίζει και να εμφανίζει τον παραγοντικό αριθμό του α.

4.4 Ερωτήσεις

1. Με ποια εντολή επιτυγχάνεται η εκτέλεση ορισμένων εντολών υπό συνθήκες;
2. Πότε χρησιμοποιούμε την εντολή if..else;
3. Να δώσετε τη μορφή του βρόχου for, να εξηγήσετε τη λειτουργία του μέσα από ένα απλό παράδειγμα.
4. Η συνάρτηση range(1, 40, 5) ποια λίστα παράγει;
5. Ποια η διαφορά του βρόχου for από το βρόχο while;
6. Σε ποια περίπτωση μπορεί να μην τερματίζει ποτέ ένας βρόχος while;

7. Ποια η βασική εντολή εισόδου και ποια η βασική εντολή εξόδου στη γλώσσα Python;
8. Με ποια λέξη ορίζουμε μια νέα συνάρτηση;

4.5 Αναφορές - Βιβλιογραφία

Downey, A. (2012) «Think Python, How to think like a computer scientist», O' Reilly. http://www.greenteapress.com/thinkPython/. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά στα πλαίσια πτυχιακής εργασίας στο ΤΕΙ Λάρισας και είναι διαθέσιμο στο Διαδίκτυο.

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 International License. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από την Ελληνική κοινότητα του Ubuntu. http://ubuntu-gr.org/ και είναι διαθέσιμο στο Διαδίκτυο.

Tutorial από το δικτυακό τόπο της γλώσσας Python για τη γνωριμία με τη γλώσσα Python – Δομή Ελέγχου και Δομές Επανάληψης (τελευταία προσπέλαση 03/07/2015),

Αβούρης Ν., Κουκιάς Μ., Παλιουράς Β., Σγάρμπας Κ. (2013) «Εισαγωγή στους υπολογιστές με τη γλώσσα Python», Εκδόσεις Πανεπιστημίου Πατρών, Πάτρα.

Δικτυακός κόμβος υποστήριξης με πλούσιο υλικό πολυμέσων για τη διδασκαλία της γλώσσας Python, http://www.Pythonschool.net/ (τελευταία προσπέλαση 09/07/2015)

Δικτυακός κόμβος υποστήριξης της γλώσσας Python, https://www.Python.org/ (τελευταία προσπέλαση 10/07/2015).

Λεβεντέας, Δ. (2010) «Taspython. Εκμάθηση Python Βήμα, 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 Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ. ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π.), Ιανουάριος 2015.

5

Δομές Δεδομένων Ι

5. Δομές Δεδομένων Ι

Στόχοι

Μετά το τέλος του κεφαλαίου θα μπορούμε να:

- περιγράφουμε τις διαφορές μεταξύ στατικών και δυναμικών δομών
- συγκρίνουμε και να επιλέγουμε την καταλληλότερη δομή ανάλογα με τον τύπο του προβλήματος
- επιλέγουμε και να εφαρμόζουμε λειτουργίες σε δομές δεδομένων, όπως οι συμβολοσειρές και οι λίστες στην προτεινόμενη γλώσσα προγραμματισμού
- χρησιμοποιούμε λίστες για την επίλυση προβλημάτων
- αξιοποιούμε τη δομή της πλειάδας για την επίλυση προβλημάτων
- εφαρμόζουμε λειτουργίες των δομών δεδομένων Λεξικό, Πλειάδα
- χρησιμοποιούμε τους τελεστές επεξεργασίας συμβολοσειρών για την επίλυση προβλημάτων.

Λέξεις κλειδιά

Δομή δεδομένων, Συμβολοσειρά, Λίστα, Λεξικό

Εισαγωγή

Μια δομή δεδομένων είναι ένα σχήμα οργάνωσης των δεδομένων τα οποία χρησιμοποιεί το πρόγραμμά μας, με την επιλογή της κατάλληλης δομής δεδομένων να παίζει σημαντικό ρόλο στην ανάπτυξη του αλγόριθμου για την επίλυση ενός προβλήματος. Οι βασικές δομές δεδομένων της Python που θα εξετάσουμε στο κεφάλαιο αυτό είναι οι **συμβολοσειρές**, οι **λίστες**, οι **πλειάδες**, τα **σύνολα** και τα **λεξικά**.

Η λίστα αποτελεί τη βασική δομή δεδομένων της Python, ενώ η δομή του λεξικού χρησιμεύει, όταν θέλουμε να κάνουμε γρήγορη αναζήτηση και ανάκληση πληροφορίας σχετική με κάποια συμβολοσειρά. Μια άλλη ενσωματωμένη δομή είναι η **πλειάδα** (tuple), η οποία μοιάζει με μια λίστα, αλλά είναι δομή που δεν μπορεί να τροποποιηθεί. Ο συνδυασμός λεξικού και πλειάδας ή λίστας, έχει πολλές εφαρμογές όπως για παράδειγμα στην αναπαράσταση **γράφων**.

Διδακτικές ενότητες

5.1 Στατικές και Δυναμικές Δομές Δεδομένων

Οι δομές δεδομένων, γενικά, χωρίζονται σε δύο βασικές κατηγορίες τις **στατικές** και τις **δυναμικές**, ανάλογα με τη φάση της ανάπτυξης του προγράμματος κατά την οποία καθορίζεται το μέγεθος της δομής. Το μέγεθος μιας στατικής δομής καθορίζεται κατά τη συγγραφή του προγράμματος από τον προγραμματιστή, ώστε να είναι γνωστό κατά τη φάση της μεταγλώττισης. Όταν το πρόγραμμα ξεκινήσει να εκτελείται δεσμεύεται χώρος στη μνήμη για τις στατικές δομές.

Το μέγεθος των στατικών δομών παραμένει σταθερό κατά την εκτέλεση του προγράμματος, αφού δεν μπορούμε να αφαιρέσουμε ούτε να προσθέσουμε αντικείμενα στις δομές αυτές. Έναν πίνακα μπορούμε να τον υλοποιούμε σε κάποιες γλώσσες προγραμματισμού και ως στατική δομή δεδομένων. Στην Python στατικές δομές δεδομένων είναι οι πλειάδες (tuples), όπως θα δούμε παρακάτω.

Από την άλλη οι **δυναμικές** δομές δεδομένων στηρίζονται σε μια λειτουργία που λέγεται **δυναμική εκχώρηση μνήμης**. Κατά τη δυναμική εκχώρηση μνήμης, το πρόγραμμα μπορεί να ζητήσει από το λειτουργικό σύστημα όση μνήμη απαιτείται για τη δημιουργία της δομής δεδομένων, κατά την εκτέλεση του προγράμματος. Οι δυναμικές δομές δεδομένων μπορούν να μεταβάλλουν το μέγεθός τους, προσθέτοντας ή αφαιρώντας αντικείμενα. Η πιο γνωστή δυναμική δομή δεδομένων είναι η λίστα, η οποία είναι και η βασική δομή δεδομένων της Python. Με βασικό δομικό λίθο τη λίστα μπορούμε να υλοποιήσουμε όποια σύνθετη δομή δεδομένων θέλουμε, όπως η στοίβα, η ουρά, το δέντρο κ.λπ. Ουσιαστικά, η λίστα της Python δεν είναι τίποτα παραπάνω από ένας δυναμικός πίνακας, δηλαδή ένας πίνακας του οποίου το μέγεθος μπορεί να αυξομειώνεται κατά την εκτέλεση του προγράμματος. Ενδιαφέρον παρουσιάζει ο τύπος της συμβολοσειράς (str) ο οποίος επιτρέπει τη δυναμική δέσμευση μνήμης, αλλά όχι τη μεταβολή του μεγέθους της.

> Στοιχεία για τις στατικές και δυναμικές δομές μπορείτε να βρείτε στο βιβλίο Προγραμματισμός Υπολογιστών, Σιδερίδης Α. κ.ά., ΟΕΔΒ, Κεφάλαιο 17, Έλεγχος και εκσφαλμάτωση προγράμματος, καθώς και στο Κεφάλαιο 14 και 16 αντίστοιχα.

5.2 Συμβολοσειρές (str)

Η **συμβολοσειρά** είναι μια ακολουθία από χαρακτήρες και μπορεί να αποτελείται από περισσότερες από μία λέξεις, με τις λέξεις να μπορούν να είναι στην Ελληνική Γλώσσα, στην Αγγλική ή σε κάθε γλώσσα που υποστηρίζεται από το πρότυπο Unicode. Μια συμβολοσειρά την ορίζουμε με εισαγωγικά αμφίπλευρα, μονά ή διπλά.

Δραστηριότητα 1. Συμβολοσειρές

Πειραματιστείτε με τις συμβολοσειρές, προσπαθώντας να υλοποιήσετε τις ενέργειες της αριστερής στήλης. Αν χρειαστείτε βοήθεια, συμβουλευτείτε τη δεξιά στήλη.

Λίγη πρακτική εξάσκηση: Ενέργειες προς εκτέλεση στο διερμηνευτή, επιλέγοντας τις κατάλληλες εντολές της γλώσσας Python	Αντίστοιχες εντολές και αποτελέσματα στο διερμηνευτή
Εκχώρηση στη μεταβλητή a της λέξης “Καλημέρα” Εκχώρηση στη μεταβλητή b της φράσης «ηλιόλουστη μέρα» Ορίστε σε μια μεταβλητή z το άθροισμα των a και b. Ορίστε σε μια μεταβλητή d το γινόμενο της a * 2. Να εμφανιστούν στην οθόνη οι τιμές των μεταβλητών a, b, z, d	a = "Καλημέρα " b = "ηλιόλουστη μέρα" z = a + b d = a * 2 **print** a Καλημέρα **print** b ηλιόλουστη μέρα **print** z **print** d Καλημέρα ηλιόλουστη μέρα Καλημέρα Καλημέρα

Να βρεθεί, με τη συνάρτηση len(), το μέγεθος της συμβολοσειράς b	len(b) 15
Να επιλεγεί ένας συγκεκριμένος χαρακτήρας από το b για παράδειγμα b[1]	b[1] λ
Να επιλεγεί ένα τμήμα από το αλφαριθμητικό b[11 : 15]	b[11 : 15] μέρα

Στην Python υπάρχει ο τύπος **str** στον οποίο οι συμβολοσειρές αποτελούν ακολουθίες χαρακτήρων. Η προσπέλαση σε κάθε χαρακτήρα της συμβολοσειράς γίνεται με τον αριθμό της θέσης που βρίσκεται, μέσα σε αγκύλες. Η αρίθμηση στην python ξεκινάει από το 0. Η συνάρτηση len μας επιστρέφει το μήκος της συμβολοσειράς.

```
>>> word = "python"
>>> letter = word[1]
>>> print letter
y
>>> print word[0]
P
```

```
>>> "python"[0]
p
>>> len( word )
6
>>> len( "123" )
3
```

Μπορούμε να πάρουμε ένα τμήμα της συμβολοσειράς με χρήση του τελεστή διαμέρισης (slice operator) ":". Όταν γράφουμε word[αρχή:τέλος], επιστρέφεται το μέρος της συμβολοσειράς που ξεκινάει από το χαρακτήρα στη θέση *αρχή* μέχρι τη θέση *τέλος*, χωρίς να περιλαμβάνει το χαρακτήρα της θέσης *τέλος*. Στη βιβλιογραφία το τμήμα αυτό της συμβολοσειράς αναφέρεται και ως **φέτα** (slice).

```
>>> s = 'Monty Python'
>>> s[0:5]
Monty
>>> s[6:len(s)]
Python
```

```
>>> s[len(s)-1]
'n'
>>> s[-1]
'n'
>>> s[0:len(s)]
Monty Python
```

Όπως φαίνεται παραπάνω, μπορεί να χρησιμοποιηθεί και αρνητική αρίθμηση, για να υποδηλώσουμε ότι ξεκινάμε να μετράμε από το τέλος της συμβολοσειράς. Ένα θέμα που πρέπει να προσέξουμε είναι η χρήση του τελεστή εκχώρησης στις συμβολοσειρές.

```
>>> word = s
#και οι δυο μεταβλητές αναφέρονται στο ίδιο αντικείμενο
```

Παραπάνω, οι μεταβλητές word και s αναφέρονται στην ίδια θέση μνήμης.

Ένας σημαντικός τελεστής που θα συναντήσουμε και αργότερα στις λίστες είναι ο τελεστής **in**, ο οποίος ελέγχει αν ένα αντικείμενο ανήκει σε ένα σύνολο αντικειμένων. Δεδομένου ότι οι συμβολοσειρές μπορούν να θεωρηθούν ως σύνολα χαρακτήρων, μπορούμε να τον χρησιμοποιήσουμε όπως φαίνεται παρακάτω. Επίσης, οι γνωστοί συγκριτικοί τελεστές, οι οποίοι συγκρίνουν με βάση τη λεξικογραφική διάταξη των χαρακτήρων, ισχύουν και στις συμβολοσειρές.

Ο τύπος str έχει και κάποιες εγγενείς μεθόδους, τις οποίες μπορούμε να καλέσουμε με τον συμβολισμό τελεία "." (dot notation). Ουσιαστικά οι συμβολοσειρές είναι αντικείμενα του τύπου str και περιέχουν μια σειρά από μεθόδους. Για να δούμε όλες τις

μεθόδους που υποστηρίζει ο τύπος str δίνουμε στον interpreter: dir(str). Η εντολή dir ισχύει και για οποιοδήποτε άλλο τύπο.

```
>>> word = "monty"
>>> "python".upper()
'PYTHON'
>>> 'a' in 'python'
False
>>> 'y' in 'python'
True
>>> word == 'monty'
True
>>> 'antonis' > 'antonia'
True
>>> 1000 < 2
False
>>> '1000' < '2'
True
```

Χρησιμοποιώντας τη Βοήθεια (Help) του προγραμματιστικού περιβάλλοντος, εφαρμόστε ενδεικτικά και άλλες μεθόδους για την επεξεργασία των συμβολοσειρών. Παραδείγματα μεθόδων: word.capitalize(), word.upper(), word.lower().

Τα αντικείμενα του τύπου str δεν είναι τροποποιήσιμα, είναι όπως λέμε *αμετάβλητα* (immutable). Δηλαδή δεν μπορούμε να αλλάξουμε μέρος της συμβολοσειράς. Για παράδειγμα, η εντολή word[0] = 'p' θα μας επιστρέψει λάθος, αφού δεν μπορεί να εκτελεστεί. Οποιαδήποτε επεξεργασία θέλουμε να κάνουμε σε συμβολοσειρές γίνεται με χρήση του τελεστή: για αποκοπή του μέρους που θέλουμε και του τελεστή + για συνένωση.

Ένα εξαιρετικά ενδιαφέρον χαρακτηριστικό της Python είναι η πολυμορφική συμπεριφορά της, η οποία φαίνεται στο παρακάτω παράδειγμα:

```
>>> def times( a, b ):
        return a * b
```

```
>>> times( 2, 3 )
6
>>> times( "python#", 3 )
python#python#python#
```

Θα λέγατε ότι ισχύει η παρακάτω σχέση;

"Python" + "Python" + "Python" = 3 * "Python"

Είναι φανερό λοιπόν ότι όλες οι αποφάσεις λαμβάνονται σε χρόνο εκτέλεσης, δηλαδή την τελευταία στιγμή (lazy evaluation). Αυτό προσδίδει στη γλώσσα τη δυναμική της συμπεριφορά.

Τι πιστεύετε ότι θα εμφανιστεί, αν δώσουμε times("Python", "Java"); Τι συμπέρασμα βγάζετε;

5.3 Δομή δεδομένων Λίστα

Μια λίστα είναι ουσιαστικά μια διατεταγμένη ακολουθία από αντικείμενα τα οποία συνήθως είναι του ίδιου τύπου. Δηλαδή μια λίστα μπορεί να αποτελείται και από αντικείμενα διαφορετικού τύπου, όπως φαίνεται στο παρακάτω παράδειγμα:

```
>>> fruits = ['apple', 'orange']
>>> list2 = [50,60,70,80]
```

Η λίστα στην Python αποτελεί τη βασική δομή δεδομένων της γλώσσας.

Για κάθε αντικείμενο που εισάγεται στη λίστα δίνεται ένας αύξων αριθμός, που χρησιμοποιείται για την αναφορά του στο αντικείμενο. Μπορεί κανείς ανά πάσα στιγμή να προσθέσει, να διαγράψει ή να αλλάξει ένα αντικείμενο σε μία λίστα.

```
daysofweek= ["Δευτέρα","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή","Σάββατο","Κυριακή"]
```

Η προσπέλαση στα στοιχεία της λίστας γίνεται ακριβώς όπως στους πίνακες, στην ψευδογλώσσα ή στις συμβολοσειρές που δείξαμε προηγουμένως. Η λίστα μπορεί και αυτή να θεωρηθεί ως ένα σύνολο από αντικείμενα, οπότε μπορούμε να χρησιμοποιήσουμε τον υπαρξιακό τελεστή ***in*** και τη συνάρτηση ***len***. Ας δούμε μερικά παραδείγματα:

```
>>> list1 = [10,20,30,40]
>>> fruits = ['apple', 'orange']
>>> len(list1)
4
>>> print fruits[0]
Apple
>>> 'apple' in fruits
True
>>> list2 = [50,60,70,80]
>>> list1+ list2
[10, 20, 30, 40, 50, 60, 70, 80]
>>> list1 + fruits
[10, 20, 30, 40, 'apple', 'orange']
>>> [1,0]*4
[1, 0, 1, 0, 1, 0, 1, 0]
```

Αν δεν θέλουμε να θέσουμε κάποια αρχική τιμή στα στοιχεία της λίστας μπορούμε να δώσουμε την τιμή ***None***, που υποδηλώνει την απουσία τιμής, δηλαδή array = [None]*N.

Μπορούμε να χρησιμοποιήσουμε τον τελεστή ":" όπως και στις συμβολοσειρές. Εδώ να σημειώσουμε ότι σε αντίθεση με τις συμβολοσειρές, οι οποίες δεν μπορούν να τροποποιηθούν (immutable), τα στοιχεία των λιστών μπορούν να αλλάξουν.

```
>>> a = [5,8,13,21,34]
>>> a[2]
13
>>> a[2] = 0
>>> print a
[5, 8, 0, 21, 34]
>>> a[1:4]
[8, 0, 21]
>>> len( "123" )
```

```
3
>>> s = 'Monty Python'
>>> s[0:5]
'Monty'
>>> s[6:len(s)]
'Python'
>>> s[len(s)-1]
'n'
>>> s[-1]
'n'
>>> s[0:len(s)]
'Monty Python'
```

Οι λίστες, όπως και οι συμβολοσειρές, έχουν κάποιες εγγενείς λειτουργίες τις οποίες μπορούμε να καλέσουμε και τις οποίες ονομάζουμε **μεθόδους**. Κάποιες από αυτές είναι οι παρακάτω:

append: προσθήκη στοιχείου στο τέλος της λίστας

insert: προσθήκη στοιχείου σε συγκεκριμένη θέση στη λίστα.

pop: αφαίρεση στοιχείου από λίστα. Αν καλέσουμε την pop χωρίς ορίσματα αφαιρεί το τελευταίο στοιχείο της λίστας. Ενώ αν δώσουμε ένα όρισμα όπως τον αριθμό 2 παρακάτω, αφαιρείται το στοιχείου που βρίσκεται στην θέση 2, δηλαδή το 13.

```
>>> a=[5,8,13,21,34]
>>> a.append(55)
>>> print a
[5, 8, 13, 21, 34, 55]
>>> a.pop()
55
```

```
>>> print a
[5, 8, 13, 21, 34]
>>> a.pop(2)
13
>>> print a
[5, 8, 21, 34, 1, 2, 3]
```

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

```
>>> a = [5,8,13,21,34]
>>> b = a[ : ]      # δημιουργεί αντίγραφο
>>> d = a           # δείχνουν στην ίδια λίστα
>>> a.pop()         # οι αλλαγές επηρεάζουν και την a και τη d
34
>>> d.pop()
21
>>> a[0]=a[1]=6
>>> print a
[6, 6, 13]
>>> print d
[6, 6, 13]
>>> print b         # Η λίστα b δεν έχει αλλάξει
[5, 8, 13, 21, 34]
```

Προσοχή! δεν μπορεί να γίνει το ίδιο με τις συμβολοσειρές. Δηλαδή η εντολή word[:] δε δημιουργεί αντίγραφο της συμβολοσειράς word.

Ένας δεύτερος τρόπος για να δημιουργήσουμε αντίγραφο μιας λίστας είναι έμμεσα με χρήση του τελεστή **+** ο οποίος δημιουργεί μια νέα λίστα, ως άθροισμα δύο ήδη υπαρχουσών λιστών. Για να δημιουργήσουμε αντίγραφο μιας λίστας αρκεί να προσθέσουμε την κενή [ ]:

```
>>> a = [5,8,13,21,34]
>>> b = a + [ ]
>>> a[0]=a[1]=a[2]=a[3]=0
>>> print a
[0, 0, 0, 0, 34]


>>> print d
[5, 8, 13, 21, 34]
```

Σημαντική σημείωση

Η εντολή

aList = aList + [item]

κάθε φορά που καλείται δημιουργεί μια νέα λίστα ως προϊόν συνένωσης των δυο λιστών, η οποία αποθηκεύεται σε μια νέα θέση στη μνήμη. Αυτή η λειτουργία έχει υπολογιστικό κόστος σε αντίθεση με την εντολή

aList.append(item)

η οποία προσθέτει το στοιχείο item στο τέλος της λίστας.

Δραστηριότητες (εισαγωγικές) ενότητας. Λίστες

Δραστηριότητα 1

Δίνεται το παρακάτω πρόγραμμα. Τι πιστεύετε ότι θα εμφανιστεί στην οθόνη μετά την εκτέλεσή του. Επαληθεύστε το αποτέλεσμα δοκιμάζοντας το πρόγραμμα στο προγραμματιστικό περιβάλλον της Python. Πώς ερμηνεύετε το αποτέλεσμα;

```
daysofweek=["Δευτέρα","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή","Σάββατο","Κυριακή"]
>>> print "θα έχω πολύ ελεύθερο χρόνο την ", daysofweek[6]
```

Ερώτηση: Ποιο πιστεύετε ότι θα είναι το αποτέλεσμα μετά την εκτέλεση του προγράμματος;

Απάντηση:

.....
.....

Ερμηνεία:

.....
.....

Δραστηριότητα 2

Δίνονται τα παρακάτω προγράμματα. Τι πιστεύετε ότι θα εμφανιστεί στην οθόνη μετά την εκτέλεσή τους. Καταγράψτε τις απαντήσεις σας στο φύλλο εργασίας. Στη συνέχεια επαληθεύστε τις απαντήσεις σας, εκτελώντας τα προγράμματα στο προγραμματιστικό περιβάλλον της Python.

```
x = [21, 23, 25, 27]
y = [5, 6, 7, 8]
z = x + y
print z
.....
print z[1]
.....
z[0] = 45
print z
.....
a = [x,y]
print a
print a [1][2]
.....
```

Δραστηριότητα 3

Παρατηρήστε αυτό που θα εμφανιστεί στην οθόνη. Δοκιμάστε τις ίδιες εντολές χωρίς να ακολουθεί το «,» μετά την εντολή print. Τι παρατηρείτε;

```
>>> for a in [15,27,31]:
....        print a,
```

Απάντηση:

.....

```
daysofweek = ["Δευτέρα","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή","Σάββατο","Κυριακή"]
for item in daysofweek:
        print item,
```

Αποτέλεσμα στη οθόνη:

.....

Δραστηριότητα 4

Μελετήστε το παρακάτω πρόγραμμα. Τι πιστεύετε ότι εμφανίζεται στην οθόνη; Επαληθεύστε τις υποθέσεις σας εκτελώντας το πρόγραμμα στο προγραμματιστικό περιβάλλον. Τι παρατηρείτε;

```
listX = [1, 2, 3]
listY = listX
print 'listX =', listX, 'list =', listY
listX[1] = 5000
print 'listX =', listX, 'list =', listY
listY[2] = 100
print 'listX =', listX, 'list =', listY
```

Καταγράψτε ποια πιστεύετε ότι θα είναι τα διαδοχικά αποτελέσματα στην οθόνη

.....

.....

.....

Επαληθεύστε τις υποθέσεις εκτελώντας το παραπάνω τμήμα κώδικα

Δραστηριότητα 5

Δημιουργήστε μια λίστα με ζώα της θάλασσας. Στη συνέχεια και με τις βασικές μεθόδους, προσπαθήστε να προσθέσετε, να διαγράψετε και να επανεισάγετε στοιχεία.

Σημείωση: Για τη διαχείριση μιας λίστας εφαρμόζονται βασικές μέθοδοι όπως: η append(X) για την προσθήκη ενός στοιχείου στο τέλος της λίστας, η insert (i, x) για την εισαγωγή ενός στοιχείου x στη λίστα πριν το i στοιχείο, η pop(i) για τη διαγραφή του i στοιχείου.

Δραστηριότητα 6

Βρείτε τι κάνει το παρακάτω πρόγραμμα. Επαληθεύστε την υπόθεσή σας, εκτελώντάς το στο προγραμματιστικό περιβάλλον της γλώσσας.

```
sqlist=[ ]
for x in range(1,11):
    sqlist.append(x*x)
print sqlist
```

5.4 Επεξεργασία λιστών

Μπορούμε να δημιουργήσουμε μια λίστα, όπως φαίνεται παρακάτω στη συνάρτηση newList. Η παρακάτω συνάρτηση παίρνει σαν όρισμα έναν αριθμό και επιστρέφει μια λίστα με αυτό το μέγεθος. Τα στοιχεία της λίστας έχουν όλα την τιμή 0. Αν θέλουμε να μη δώσουμε τιμή στα στοιχεία της λίστας, θα χρησιμοποιήσουμε την τιμή *None*, που η Python διαθέτει γι' αυτό το σκοπό.

Στη συνέχεια μπορούμε να προσπελάσουμε τα στοιχεία της λίστας, όπως φαίνεται στις εντολές της δεξιάς στήλης.

```
>>> def  newList(size):
        array = [ ]
        for i in range(0,size):
            array.append( 0 )
        return array
```

```
>>> a = newList(5)
>>> a[1] = a[3] = 1
>>> print a
[0,  1,  0,  1,  0]
```

Στις λίστες στην Python, όπως και στις περισσότερες σύγχρονες γλώσσες προγραμματισμού, η αρίθμηση ξεκινάει από το 0 και όχι από το 1.

Αν "βιαζόμαστε", μπορούμε να δημιουργήσουμε μια νέα λίστα με τον τελεστή του πολλαπλασιασμού: a = 5*[0]. Ωστόσο είναι καλύτερα να αποφύγουμε αυτό το ιδίωμα και να χτίσουμε τη λίστα με διαδοχικές κλήσεις της append.

Αν θέλουμε να διατρέξουμε τα στοιχεία μιας λίστας, δεν είναι ανάγκη να χρησιμοποιήσουμε μια μεταβλητή-μετρητή, όπως σε άλλες γλώσσες, αλλά μπορούμε να χρησιμοποιήσουμε το παρακάτω ιδίωμα της Python (αριστερά):

Python		**Ψευδοκώδικας**
`for item in array :` `  print item`	`N = len(array)` `for i in range(0, N):` `  print array[i]`	**Για** i **από** 0 **μέχρι** N–1 **Εμφάνισε** array[i] **Τέλος_Επανάληψης**

Μπορούμε επίσης να κατασκευάσουμε μια λίστα με στοιχεία λίστες. Αυτό μπορεί να μας φανεί χρήσιμο σε προβλήματα τα οποία μοντελοποιούνται με τη χρήση ενός πίνακα δυο διαστάσεων, όπως για παράδειγμα η εύρεση του συντομότερου μονοπατιού μεταξύ δυο πόλεων. Παρακάτω φαίνεται ένας πίνακας στον οποίο έχουν καταχωρηθεί οι αποστάσεις μεταξύ τριών πόλεων:

A =

	0	1	2
0	0	12	14
1	22	0	28
2	15	16	0

Ο πίνακας σχηματίζεται από γραμμές και στήλες. Για να αναφερθούμε στο δεύτερο στοιχείο της πρώτης γραμμής, το 12, γράφουμε A[0,1], αφού η αρίθμηση στις γραμμές και στις στήλες ξεκινάει από το 0. Η ερμηνεία σε αυτή την περίπτωση είναι ότι για να μεταβούμε από την πόλη 0 στην πόλη 1 χρειαζόμαστε 12 km. Αντίθετα, για να κάνουμε την αντίστροφη διαδρομή, θέλουμε 22 km, αφού A[1,0] = 22.

Ο παραπάνω πίνακας μπορεί να υλοποιηθεί στην Python αν θεωρήσουμε τις γραμμές ως λίστες οι οποίες είναι εμφωλευμένες σε μια μεγάλη κεντρική λίστα, όπως φαίνεται στο παρακάτω παράδειγμα:

```
>>> A = [ [0, 12, 14], [22, 0, 28], [15, 6, 0] ]
>>> print A[0]
[0, 12, 14]
>>> print A[0][2]
14
```

Ένα μεγάλο πλεονέκτημα στο οποίο πρέπει να σταθούμε είναι η δυνατότητα αναφοράς και επεξεργασίας μιας συγκεκριμένης γραμμής του πίνακα, σαν να ήταν ένας ξεχωριστός μονοδιάστασος πίνακας.

Πώς όμως θα δημιουργήσουμε έναν πίνακα δύο διαστάσεων; Προτείνεται πάλι η χρήση της append.

Python	Ψευδοκώδικας
`for i in range(0,3):` `  a.append( [ ] )` `  for j in range(0,3):` `  a[i].append(0)`	Για i από 0 μέχρι 2 Για j από 0 μέχρι 2 a[i][j] ← 0 Τέλος_Επανάληψης Τέλος_Επανάληψης

Ενώ σε κάποιες γλώσσες χρησιμοποιείται ο συμβολισμός a[i,j] για προσπέλαση στο στοιχείο που βρίσκεται στην i-οστή γραμμή και στην j-οστή στήλη, στην Python γράφουμε a[i][j]. Αυτός ο συμβολισμός, μας δίνει τη δυνατότητα να επεξεργαστούμε ξεχωριστά μια γραμμή γράφοντας π.χ. a[2] για τη δεύτερη γραμμή. Το συμβολισμό αυτό είναι καλό να τον χρησιμοποιούμε και σε ψευδοκώδικα, κατά τη μεταφορά των αλγορίθμων σε Python.

Μια λίστα μπορεί να λειτουργήσει σαν **στοίβα**, αν οι μόνες πράξεις που εφαρμόζουμε σε αυτήν είναι η **append** και η **pop**, δηλαδή η προσθήκη (ώθηση) και αφαίρεση (απώθηση) στοιχείων μόνο από το ένα άκρο. Ομοίως η λίστα μπορεί να λειτουργήσει σαν ουρά αν προσθέτουμε από το ένα άκρο και αφαιρούμε από το άλλο. Η στοίβα και η

ουρά είναι πολύ χρήσιμες δομές δεδομένων και θα τις μελετήσουμε στη Γ' Λυκείου.

5.5 Σύνολα

Τα **σύνολα** αποτελούν μία από τις ενσωματωμένες δομές δεδομένων της Python και υλοποιούν τα ομώνυμα μαθηματικά αντικείμενα. Ένα σύνολο είναι μια ομάδα από ***μη διατεταγμένα αντικείμενα***, με κάθε αντικείμενο να εμφανίζεται μία φορά. Τα σύνολα υποστηρίζουν τις γνωστές *πράξεις συνόλων*, δηλαδή την ένωση, την τομή και το συμπλήρωμα. Ας δούμε μερικά παραδείγματα:

```
>>> myset = set( [ 1 ,2, 3, 1, 2, 3, 1, 2, 3, 8])
>>> myset
set( [8,  1,  2,  3] )
>>> myset.intersection( set( [1, 2, 90] ) )
set( [1,  2] )
```

5.6 Πλειάδες

Ένας σύνθετος τύπος του οποίου τα στοιχεία είναι *αμετάβλητα* (immutable) είναι η **πλειάδα** (tuple). Τη χρησιμοποιούμε, όταν θέλουμε να συγκρατήσουμε μαζί πολλαπλά αντικείμενα. Για παράδειγμα, θέλουμε να έχουμε σε μια λίστα τα στοιχεία των υπαλλήλων μιας επιχείρησης, όπως όνομα, επώνυμο, βαθμός, τηλέφωνο, τμήμα, μισθός κ.λπ. Αυτό θα το υλοποιήσουμε με μια λίστα από πλειάδες, όπου κάθε πλειάδα θα αντιστοιχεί σε έναν εργαζόμενο. Η πλειάδα μοιάζει με μια λίστα, όπως φαίνεται στα παραδείγματα παρακάτω:

```
>>> rec = 'a', 'b', 120, 'r'
>>> print rec
('a', 'b', 120, 'r')
>>> rec[2]
120
>>> len( rec )
4
```

```
>>> rec[1:3]
('b',  120)
>>> rec = ('apple' ,) + rec[2:3]
>>> print rec
( 'apple', 120 )
```

Συνήθως οι πλειάδες χρησιμοποιούνται για την αναπαράσταση μιας σειράς χαρακτηριστικών/ιδιοτήτων μιας οντότητας, εφόσον τα χαρακτηριστικά αυτά δεν μεταβάλλονται. Μια σημαντική εφαρμογή των πλειάδων είναι ότι μπορούμε να ορίσουμε συναρτήσεις με ορίσματα μεταβλητού μεγέθους.

Μια άλλη γνωστή εφαρμογή τους είναι η **αντιμετάθεση των τιμών δύο μεταβλητών, χωρίς τη χρήση βοηθητικής μεταβλητής,** με την παρακάτω εντολή:

```
>>> a , b = b, a
```

5.7 Λεξικά

Το λεξικό είναι μια εξαιρετικά χρήσιμη ενσωματωμένη (built-in) δομή της Python. Φανταστείτε το σαν έναν τηλεφωνικό κατάλογο ο οποίος μας δίνει τη δυνατότητα να βρούμε πολύ γρήγορα τα στοιχεία κάποιου, μόνο από το όνομά του. Προγραμματιστικά, φανταστείτε το σαν μια λίστα που έχει ως δείκτες αλφαριθμητικά και όχι ακέραιους αριθμούς. Για παράδειγμα, μπορούμε να γράψουμε version["python"] = 3.4. Όπως σε έναν πίνακα, η θέση κάθε στοιχείου είναι μοναδική και δεν μπορεί να περιέχει ταυτόχρονα δυο τιμές, έτσι και στο λεξικό, η λέξη-κλειδί, που χρησιμοποιούμε μέσα στις αγκύλες, έχει μοναδική τιμή και εμφανίζεται το πολύ μια 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:

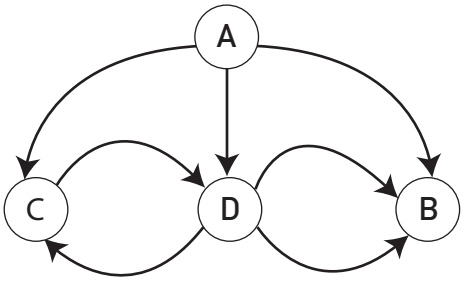
```
>>> dictionary = { 'A':2, 'B':4, 'Ω':8,  'M':16  }
>>> list(dictionary.keys() )
['A',  'B',  'Ω',  'M']
>>> dictionary ['Ω']
8
>>> len(dictionary )
4
>>> del dictionary ['Ω']
>>> del dictionary ['A']
```

```
>>> print dictionary
{'B':4, 'M':16}
>>> dictionary ['Λ'] = 32
>>> print dictionary
{'B':4, 'Λ':32, 'M':16}
>>> 'Λ' in dictionary
True
```

Παραπάνω φαίνονται κάποιες λειτουργίες του λεξικού, όπως η διαγραφή (**del**) και ο έλεγχος ύπαρξης ενός στοιχείου με τον τελεστή **in**.

Θα πρέπει να σημειωθεί ότι κάθε κλειδί έχει μοναδική τιμή, ενώ με την εκχώρηση *Λεξικό[κλειδί]=τιμή* δημιουργείται το ζεύγος {κλειδί : τιμή}. Αν θέλουμε να αντιστοιχήσουμε σε ένα κλειδί περισσότερες από μια τιμές, τότε η τιμή του κλειδιού είναι μια λίστα από τις τιμές αυτές.

Η δομή του γραφήματος συναντάται σε πολλά προβλήματα της Επιστήμης της Πληροφορικής, όπως στις μηχανές αναζήτησης και στην αναπαράσταση του παγκόσμιου ιστού. Παράδειγμα, ο παρακάτω δημοφιλής γράφος, αναπαριστά το πρόβλημα των γεφυρών του Königsberg, ένα από τα πιο γνωστά προβλήματα της Επιστήμης Υπολογιστών, για το οποίο μπορείτε να αναζητήσετε πληροφορίες στο Διαδίκτυο:



Η αναπαράστασή του με λίστα γειτνίασης στην Python υλοποιείται με ένα λεξικό, με κλειδιά τις κορυφές του γράφου και τιμές τη λίστα ακμών κάθε κλειδιού, δηλαδή τη λίστα των γειτονικών κορυφών.

```
graph  = { 'A': ['B', 'C', 'D'],
 'B': ['D']
 'C': ['D']
 'D': ['B', 'C'] }
```

5.8 Δραστηριότητες κεφαλαίου

Δραστηριότητα 1. Βασικές επεξεργασίες λιστών

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο μέχρι το άθροισμά τους να ξεπεράσει το 1000, θα τους αποθηκεύει σε μια λίστα και στη συνέχεια θα υπολογίζει και θα εμφανίζει:

1. το άθροισμά τους
2. το μέσο όρο των στοιχείων της λίστας
3. τη μέγιστη τιμή που έχει στοιχείο της λίστας
4. πόσες φορές εμφανίζεται αυτή η μέγιστη τιμή
5. τη θέση που θα καταλάμβανε ο τελευταίος αριθμός που δόθηκε, αν η λίστα ήταν ταξινομημένη σε αύξουσα σειρά. Θεωρήστε ότι ο τελευταίος αριθμός είναι μοναδικός.

Δραστηριότητα 2

Να δημιουργήσετε τη δομή δεδομένων στοίβα με τη χρήση λίστας, υλοποιώντας τις λειτουργίες **push** (ώθηση), **pop** (απώθηση) για λίστα.

Δραστηριότητα 3

Να δημιουργήσετε τη δομή δεδομένων ουρά με τη χρήση λίστας, υλοποιώντας τις λειτουργίες **enqueue** (εισαγωγή), **dequeue** (εξαγωγή) για λίστα.

Δραστηριότητα 4

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο, μέχρι να δοθεί ο αριθμός 0. Στη συνέχεια με τη χρήση λίστας θα εμφανίζει τους αριθμούς σε αντίστροφη σειρά από αυτή με την οποία τους διάβασε.

Δραστηριότητα 5

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μια λίστα από 100 αριθμούς και θα διαχωρίζει τους αριθμούς σε δυο νέες λίστες, μια για τους θετικούς και μια για τους αρνητικούς. Οι αριθμοί πρέπει να παραμείνουν στη σειρά με την οποία δόθηκαν.

Δραστηριότητα 6

Τα βιβλία αναγνωρίζονται από ένα μοναδικό κωδικό ο οποίος είναι γνωστός ως ISBN. Ωστόσο τα τελευταία χρόνια χρησιμοποιείται και ένας άλλος κωδικός, ο ISSN. Να γράψετε πρόγραμμα το οποίο θα διαβάζει τους κωδικούς βιβλίων μέχρι να δοθεί ο κωδικός "000" και στη συνέχεια θα διαχωρίζει σε δύο λίστες, τα βιβλία με κωδικό ISBN και εκείνα με ISSN.

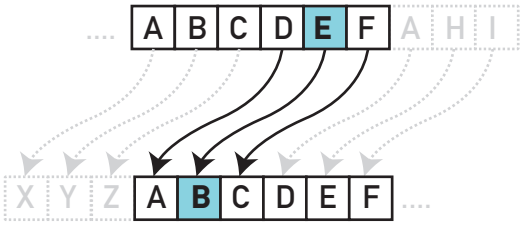
Υπόδειξη: Να επεξεργαστείτε κάθε κωδικό σαν συμβολοσειρά (str) και να ελέγξετε το πρόθεμά του. Για παράδειγμα, σε ένα βιβλίο με ISBN, το ISBN θα καταλαμβάνει τις τέσσερις πρώτες θέσεις της συμβολοσειράς.

Δραστηριότητα 7. Αλγόριθμος Συγχώνευσης

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει δύο λίστες με αριθμούς. Η πρώτη λίστα περιέχει μόνο θετικούς και η δεύτερη μόνο αρνητικούς. Η εισαγωγή δεδομένων για κάθε λίστα τερματίζεται όταν δοθεί ο αριθμός 0, ο οποίος δεν πρέπει να αποθηκευθεί σε κάποια λίστα. Οι αριθμοί κάθε λίστας δίνονται σε αύξουσα σειρά, π.χ. 4, 6, 10, 145 και –100, –98, –10. Στη συνέχεια, το πρόγραμμα θα συγχωνεύει τις δύο λίστες σε μία, έτσι ώστε στην τελική λίστα οι αριθμοί να βρίσκονται σε αύξουσα σειρά κατά απόλυτη τιμή. Αν δύο ετερόσημοι αριθμοί έχουν την ίδια απόλυτη τιμή, υπερισχύουν οι θετικοί., π.χ. 4, 6, 10, –10, 98, –100, 145.

Δραστηριότητα 8. Αλγόριθμος κρυπτογράφησης του Καίσαρα

Με βάση τον αλγόριθμο κρυπτογράφησης του Καίσαρα, όπως φαίνεται στο παρακάτω σχήμα κρυπτογράφησης, δεχόμαστε ότι κάθε γράμμα θα αντιστοιχιστεί στο γράμμα που βρίσκεται τρεις θέσεις πίσω του στο αλφάβητο. Σε αυτή την περίπτωση μπορούμε να ορίσουμε ένα λεξικό.




```
>>> cipher = { 'A':'X', 'B':'Y','C': 'Z','D': 'A', ... , 'X':'U'}
```

Πλέον η κρυπτογράφησή ενός κειμένου text είναι πολύ απλή, όπως φαίνεται και στο παρακάτω τμήμα κώδικα:

```
>>> def encryptCeasar( text, alphabet ):
    cipherText = ""
    for letter in text:
        if letter in alphabet:
            cipherText = cipherText + cipher[letter]
        else:
            cipherText = cipherText + letter

    return cipherText
```

Να σχεδιάσετε μια συνάρτηση που να λαμβάνει τη μετατόπιση-κλειδί και να παράγει το λεξικό αντιστοίχισης των γραμμάτων cipher. Στη συνέχεια, να υλοποιήσετε τις αντίστοιχες συναρτήσεις για την αποκρυπτογράφηση.

Υπόδειξη: Να αναπτύξετε μια συνάρτηση η οποία να δέχεται το λεξικό κρυπτογράφησης και να παράγει το λεξικό αποκρυπτογράφησης, έτσι ώστε να μπορεί να χρησιμοποιηθεί η συνάρτηση encryptCeasar για αποκρυπτογράφηση.

Δραστηριότητα 9. Συχνότητα γραμμάτων

Να γράψετε πρόγραμμα στη γλώσσα Python το οποίο θα δέχεται ως είσοδο ένα κείμενο και θα ελέγχει αν εμφανίζονται όλα τα γράμματα του αλφαβήτου σε αυτό με την ίδια ακριβώς συχνότητα, όπως για παράδειγμα συμβαίνει στη συμβολοσειρά

'abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz', όπου όλα τα γράμματα εμφανίζονται δύο φορές.

Δραστηριότητα 10

Να γραφεί πρόγραμμα που να διαβάζει μια λέξη και να ελέγχει αν η λέξη είναι *καρκινική*, εμφανίζοντας ανάλογο μήνυμα. Δηλαδή εξετάζει αν η λέξη είναι *συμμετρική*, δηλαδή αν μπορεί να διαβαστεί είτε από την αρχή είτε από το τέλος, όπως για παράδειγμα οι λέξεις ANNA ή RADAR.

Δραστηριότητα 11

Βρείτε τι κάνει το παρακάτω πρόγραμμα. Επαληθεύστε την υπόθεση σας εκτελώντας το στο προγραμματιστικό περιβάλλον της γλώσσας.

```
sqlist=[ ]
for x in range(1,11):
    sqlist.append(x*x)
print sqlist
```

Δραστηριότητα 12

Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίστα από λέξεις και θα επιστρέφει τη λέξη που σχηματίζεται από τα πρώτα γράμματα των λέξεων.

Δραστηριότητα 13

Να γράψετε ένα πρόγραμμα σε Python, το οποίο θα διαβάζει από το πληκτρολόγιο μια λίστα από λέξεις μέχρι να δοθεί η λέξη 'end'.

Στη συνέχεια θα εμφανίζει τη λέξη ή τις λέξεις με τα περισσότερα γράμματα (υπόδειξη: Κατασκευάστε μια λίστα με το πλήθος των γραμμάτων κάθε λέξης).

Επίσης, θα εμφανίζει το πλήθος των λέξεων που τελειώνουν με το γράμμα 's'.

5.9 Ερωτήσεις

1. Ποια η διαφορά μεταξύ μιας στατικής δομής δεδομένων και μιας δυναμικής δομής δεδομένων;
2. Ποιες είναι οι βασικές δομές δεδομένων που υποστηρίζει η γλώσσα προγραμματισμού Python;
3. Τι είναι η Λίστα στη γλώσσα προγραμματισμού Python και πώς μπορούμε να την ορίσουμε;
4. Ποιες είναι οι βασικές λειτουργίες-μέθοδοι που μπορούμε να εκτελέσουμε σε μια λίστα;
5. Τι είναι η πλειάδα και πότε τη χρησιμοποιούμε;
6. Τι είναι το λεξικό;

5.10 Αναφορές - Βιβλιογραφία

Downey, A. (2012) «Think Python, How to think like a computer scientist», O' Reilly. http://www.greenteapress.com/thinkPython/. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από στα πλαίσια πτυχιακής εργασίας στο ΤΕΙ Λάρισας και είναι διαθέσιμο στο Διαδίκτυο.

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 International License. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από την Ελληνική κοινότητα του Ubuntu. http://ubuntu-gr.org/ και είναι διαθέσιμο στο Διαδίκτυο.

Δικτυακός κόμβος υποστήριξης με πλούσιο υλικό πολυμέσων για τη διδασκαλία της γλώσσας Python, http://www.Pythonschool.net/ (τελευταία προσπέλαση 09/07/2015).

Δικτυακός κόμβος υποστήριξης της γλώσσας Python, https://www.Python.org/ (τελευταία προσπέλαση 10/07/2015).

Λεβεντέας, Δ. (2010) «Taspython. Εκμάθηση Python Βήμα, 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 Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ. ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π.), Ιανουάριος 2015.

6

Κλασικοί αλγόριθμοι Ι

6. Κλασικοί Αλγόριθμοι Ι

Στόχοι

Οι μαθητές να μπορούν να:

- αναλύουν και να εφαρμόζουν κατάλληλα κλασικούς αλγόριθμους για προβλήματα ταξινόμησης και αναζήτησης
- υλοποιούν σε μια γλώσσα προγραμματισμού κλασικούς αλγόριθμους ταξινόμησης και αναζήτησης.

Εισαγωγή

Ένας αλγόριθμος είναι μια ακολουθία πλήρως καθορισμένων και εκτελέσιμων βημάτων-εντολών, που οδηγεί στην επίλυση ενός προβλήματος, σε πεπερασμένο χρόνο. Στην ενότητα αυτή θα ασχοληθούμε με κάποιους γνωστούς και απλούς αλγόριθμους, όπως ο αλγόριθμος του Μέγιστου Κοινού Διαιρέτη του Ευκλείδη, ο αλγόριθμος της σειριακής αναζήτησης και ο αλγόριθμος της ταξινόμησης με επιλογή.

Λέξεις κλειδιά

Σειριακή αναζήτηση, Ταξινόμηση με επιλογή

Διδακτικές ενότητες

6.1 Υπολογισμός Μέγιστου Κοινού Διαιρέτη

Βασικές έννοιες της επιστήμης της πληροφορικής υπήρχαν πολύ καιρό πριν εμφανιστούν οι υπολογιστές, όσο παράδοξο και αν ακούγεται αυτό. Η έννοια του αλγόριθμου που συνιστά τα θεμέλια και την επιστημονική υπόσταση της Πληροφορικής υπήρχε από την αρχαιότητα. Δύο από τους πιο δημοφιλείς αλγόριθμους που έχουν επινοήσει αρχαίοι Έλληνες, είναι το κόσκινο του Ερατοσθένη για την εύρεση πρώτων αριθμών και ο αλγόριθμος υπολογισμού του Μέγιστου Κοινού Διαιρέτη του Ευκλείδη.

Ο αλγόριθμος του Ευκλείδη βασίζεται στην παρακάτω απλή, αλλά όχι προφανή ιδέα:

ΜΚΔ(α, β) = **ΜΚΔ**(β, α mod β)

Για παράδειγμα, ας προσπαθήσουμε, με επαναληπτική εφαρμογή της παραπάνω σχέσης, να βρούμε το μέγιστο κοινό διαιρέτη των αριθμών 100 και 36:

ΜΚΔ(100, 36) = **ΜΚΔ**(36, 28) = **ΜΚΔ**(28, 8) = **ΜΚΔ**(8, 4) = **ΜΚΔ**(4, 0) = 4

Από τις παραπάνω σχέσεις φαίνεται ότι το κριτήριο διακοπής της επανάληψης θα μπορούσε να είναι το β = 0. Έτσι ο αλγόριθμος σε ψευδοκώδικα και Python δίνεται παρακάτω:

Python	Ψευδοκώδικας
`def gcd( a, b ) :` `  while  b > 0 :` `  a , b = b , a % b` `  return a`	Αλγόριθμος ΜΚΔ Δεδομένα // α, β // Όσο β > 0 Επανάλαβε υ ← α mod β α ← β β ← υ Τέλος_Επανάληψης Αποτελέσματα // α // Τέλος ΜΚΔ

Συνήθως η αναπαράσταση ενός αλγόριθμου σε ψευδοκώδικα είναι αρκετά πιο απλή από την αναπαράστασή του σε οποιαδήποτε γλώσσα προγραμματισμού. Η Python ωστόσο, είναι μια εξαίρεση σε αυτό τον κανόνα. Το επίπεδο πολυπλοκότητάς της είναι τουλάχιστον το ίδιο με αυτό του ψευδοκώδικα, ενώ η χρήση χαρακτηριστικών της ιδιωμάτων, όπως η αντιμετάθεση μεταβλητών με χρήση πλειάδας, απλοποιεί ακόμα περισσότερο τον κώδικα, όπως φαίνεται παραπάνω.

6.2 Σειριακή Αναζήτηση

Πολλές φορές χρειάζεται να αναζητήσουμε κάτι συγκεκριμένο μέσα σε μια συλλογή δεδομένων. Η αναζήτηση γίνεται συνήθως με βάση κάποιο χαρακτηριστικό των δεδομένων, όπως για παράδειγμα το επώνυμο, ο αριθμός ταυτότητας κ.λπ. Η πιο απλή μορφή αναζήτησης είναι η σειριακή ή γραμμική αναζήτηση σύμφωνα με την οποία εκτελούμε εξαντλητική αναζήτηση σε όλα τα στοιχεία της συλλογής, μέχρι να βρούμε αυτό που ψάχνουμε. Έτσι, στη χειρότερη περίπτωση, αν ξεκινήσουμε από το πρώτο στοιχείο και αυτό που ψάχνουμε είναι τελευταίο, θα κάνουμε τόσες συγκρίσεις, όσα

είναι και τα στοιχεία που έχουμε. Μια πρώτη υλοποίηση του αλγόριθμου φαίνεται παρακάτω:

Αλγόριθμος Σειριακής Αναζήτησης (έκδοση 1.0)	
Python	**Ψευδοκώδικας**
def linearSearch(array, key): found = False for item in array : if item == key : found = True return found	Αλγόριθμος Σειριακή_Αναζήτηση Δεδομένα // Συλλογή, ζητούμενο // Βρέθηκε ← Ψευδής Για κάθε στοιχείο της Συλλογής Αν στοιχείο = ζητούμενο Τότε Βρέθηκε ← Αληθής Τέλος_Αν Τέλος_Επανάληψης Αποτελέσματα // Βρέθηκε // Τέλος Σειριακή_Αναζήτηση

Η έκδοση 1.0 του αλγόριθμου επιστρέφει μόνο την πληροφορία αν το στοιχείο υπάρχει στη συλλογή ή όχι. Ωστόσο, πολλές φορές θέλουμε να βρούμε και τη θέση του στοιχείου στη συλλογή, ειδικά αν αυτή υλοποιείται με κάποια δομή δεδομένων που επιτρέπει την αποθήκευση των στοιχείων με κάποια διάταξη. Έτσι μπορούμε να προσθέσουμε και μία μεταβλητή θέση/position. Τώρα όμως η λογική μεταβλητή είναι περιττή, αφού η μεταβλητή θέση μπορεί να παίξει και το ρόλο της λογικής μεταβλητής ως εξής: Αν η θέση παραμείνει –1, τότε σημαίνει ότι το στοιχείο δε βρέθηκε. Αν όμως βρεθεί, τότε θα πάρει την τιμή της θέσης του δείκτη, η οποία θα είναι μη αρνητική, όπως φαίνεται στην επόμενη έκδοση 2.0 του αλγόριθμού μας.

Αλγόριθμος Σειριακής Αναζήτησης (έκδοση 2.0)	
Python	Ψευδοκώδικας
def linearSearch(array, key) : pos = –1 i = 1 while pos < 0 and i <= N : if array[i] == key : pos = i i = i + 1 return pos	Αλγόριθμος Σειριακή_Αναζήτηση Δεδομένα // Συλλογή, ζητούμενο // θέση ← –1 i ← 1 Όσο θέση<0 και i<=N Επανάλαβε Αν Συλλογή[i] = ζητούμενο Τότε θέση ← i Τέλος_Αν i ← i + 1 Τέλος_Επανάληψης Αποτελέσματα // θέση // Τέλος Σειριακή_Αναζήτηση

Μια ακόμα έκδοση του αλγόριθμου, που αποδεικνύει ακόμα μια φορά την απλότητα αλλά και τη μινιμαλιστική έκφραση της Python, είναι η παρακάτω, η οποία εκμεταλλεύεται το γνωστό ιδίωμα της for και τη δυνατότητα βίαιης διακοπής του, χωρίς αυτό να θεωρείται πλέον κακή πρακτική.

Αλγόριθμος Σειριακής Αναζήτησης 3.0

```
def linearSearch( array, key ) :
    for item in array :
        if item == key :
            return True

    return False
```

Μπορείτε να τροποποιήσετε την παραπάνω συνάρτηση, ώστε να επιστρέφει τη θέση του στοιχείου στον πίνακα και όχι μόνο απάντηση True/False στο ερώτημα ύπαρξης.

6.3 Ταξινόμηση με Επιλογή

Το πρόβλημα της ταξινόμησης αναφέρεται στην αναδιάταξη των στοιχείων μιας λίστας, ώστε αυτά να βρεθούν σε αύξουσα ή σε φθίνουσα σειρά, σε σχέση με κάποια σχέση διάταξης. Πολλές φορές χρειάζεται να ταξινομήσουμε λίστες αριθμών, λέξεων ή εγγράφων. Για παράδειγμα, μπορεί να θέλουμε να ταξινομήσουμε τις καρτέλες των μαθητών. Θα πρέπει σε αυτή την περίπτωση να επιλέξουμε ένα χαρακτηριστικό με βάση το οποίο θα γίνει η ταξινόμηση, όπως για παράδειγμα το όνομα, ο βαθμός ή ο αριθμός μητρώου. Το χαρακτηριστικό αυτό λέγεται κλειδί και συνήθως (αλλά όχι πάντα) προσδιορίζει μονοσήμαντα την οντότητα στην οποία αναφέρεται.

Ένας από τους πιο απλούς αλγόριθμους ταξινόμησης είναι ο αλγόριθμος ταξινόμησης με επιλογή (selection sort). Η λογική του αλγόριθμου είναι η παρακάτω:

Βήμα 1. Βρίσκουμε το μικρότερο στοιχείο του πίνακα και το τοποθετούμε στην πρώτη θέση.

Βήμα 2. Βρίσκουμε το αμέσως μικρότερο στοιχείο του πίνακα και το τοποθετούμε στη δεύτερη θέση.

Βήμα 3. Συνεχίζουμε βρίσκοντας κάθε φορά το μικρότερο στοιχείο από τα στοιχεία του πίνακα που απομένουν και το τοποθετούμε στο τέλος των ήδη ταξινομημένων στοιχείων.

Η παραπάνω λογική θα μπορούσε να δοθεί σε ψευδοκώδικα ό-πως φαίνεται παρακάτω:

```
Αλγόριθμος Ταξινόμηση_Επιλογής
Δεδομένα // A, N //
    Για i από 0 μέχρι N–1
        θέση ← θέση_Ελάχιστου(i, N)
        Αντιμετάθεσε A[ i ], A[θέση]
    Τέλος_Επανάληψης
Αποτελέσματα // A //
Τέλος Ταξινόμηση_Επιλογής
```

και ταυτόχρονα να υλοποιηθεί σε Python με την παρακάτω συνάρτηση:

Αλγόριθμος Ταξινόμησης με Επιλογή

```
def selectionSort( array ) :
    N = len( array )
    for i in range(0, N) :
        pos = posMin( i, N, array)
        array[i], array[pos] = array[pos], array[i]
```

Ο αλγόριθμος χρησιμοποιεί έναν άλλο αλγόριθμο για να υπολογί-ζει τη θέση του ελάχιστου στοιχείου του πίνακα.

Αλγόριθμος Εύρεσης της θέσης του Ελάχιστου Στοιχείου	
Python	Ψευδοκώδικας
def posMin(start, size, array) : pos = start for i in range(start, size) : if array[i] < array[pos] : pos = i return pos	Αλγόριθμος θέση_Ελάχιστου Δεδομένα // αρχή, τέλος, Α // θέση ← αρχή Για i από αρχή μέχρι τέλος Αν A[i] < A[θέση] Τότε θέση ← i Τέλος_Αν Τέλος_Επανάληψης Αποτελέσματα // θέση // Τέλος Ταξινόμηση_Επιλογής

Η posMin δέχεται το μέγεθος της λίστας size. Το τελευταίο στοιχείο της λίστας σε αυτή την περίπτωση είναι το size-1.

Παρακάτω φαίνεται το περιεχόμενο ενός πίνακα 7 αριθμών, για κάθε βήμα της εκτέλεσης του αλγόριθμου ταξινόμησης της επιλογής:

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

6.4 Δραστηριότητες κεφαλαίου

Δραστηριότητα 1. Το κόσκινο του Ερατοσθένη

Να αναζητήσετε στο Διαδίκτυο πληροφορίες για το κόσκινο του Ερατοσθένη και στη συνέχεια, αφού περιγράψετε τη λειτουργία του στην τάξη, να υλοποιήσετε τον αλγόριθμο αυτό σε Python.

Δραστηριότητα 2

Να σχεδιάσετε έναν άλλον αλγόριθμο για τον υπολογισμό του Μέγιστου Κοινού Διαιρέτη δύο αριθμών, εκτελώντας εξαντλητική αναζήτηση των πιθανών διαιρετών μέχρι να βρεθεί ο μέγιστος. Έχει σημασία από ποιον αριθμό θα ξεκινήσουμε;

Δραστηριότητα 3

Για δύο αριθμούς α και β, τα διαδοχικά βήματα του αλγόριθμου του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη είναι:

ΜΚΔ(3600, _____) = **ΜΚΔ**(1100, _____) = **ΜΚΔ**(_____,_____)

= **ΜΚΔ**(_____, 100) =

ΜΚΔ(_____,_____) = 100

Να συμπληρώσετε τα κενά στα παραπάνω βήματα του αλγόριθμου.

Δραστηριότητα 4

Να σχεδιάσετε μια συνάρτηση, η οποία θα δέχεται δύο λίστες αριθμών και θα εμφανίζει πόσοι αριθμοί εμφανίζονται και στις δύο λίστες.

Δραστηριότητα 5

Να σχεδιάσετε μια συνάρτηση η οποία θα δέχεται δύο λίστες αριθμών και θα εμφανίζει πόσοι αριθμοί εμφανίζονται μόνο σε μία λίστα.

Δραστηριότητα 6

Να τροποποιήσετε τον αλγόριθμο της σειριακής αναζήτησης, ώστε να υπολογίζει πόσες φορές εμφανίζεται το ζητούμενο στοιχείο στον πίνακα και να εμφανίζει στην οθόνη όλες αυτές τις θέσεις.

Δραστηριότητα 7

Ας υποθέσουμε ότι τα στοιχεία ενός πίνακα, στον οποίο μας ζητείται να εκτελέσουμε αναζήτηση, είναι ακέραιοι αριθμοί ταξινομημένοι σε αύξουσα σειρά. Αν, καθώς σαρώνουμε τον πίνακα, βρεθούμε σε στοιχείο το οποίο είναι μικρότερο από αυτό που ψάχνουμε, αυτό σημαίνει ότι το ζητούμενο θα το είχαμε ήδη συναντήσει, αν υπήρχε. Να τροποποιήσετε τον αλγόριθμο σειριακής αναζήτησης, ώστε, όταν διαπιστώσει ότι το ζητούμενο στοιχείο δεν υπάρχει στον πίνακα, να τερματίζει.

Δραστηριότητα 8

Να τροποποιήσετε τον αλγόριθμο της ταξινόμησης με επιλογή, ώστε να ταξινομεί έναν πίνακα ακεραίων σε φθίνουσα σειρά.

Δραστηριότητα 9

Να τροποποιήσετε τον αλγόριθμο της ταξινόμησης με επιλογή, ώστε σε κάθε βήμα του να βρίσκει το μικρότερο και το μεγαλύτερο, να τοποθετεί το μικρότερο στην αρχή του πίνακα και το μεγαλύτερο στο τέλος.

6.5 Ερωτήσεις

1. Κατονομάστε τρεις γνωστούς αλγόριθμους.
2. Σε ποια ιδέα βασίζεται ο υπολογισμός του Μέγιστου Κοινού Διαιρέτη στον αλγόριθμο του Ευκλείδη;
3. Αν χρησιμοποιήσουμε τη σειριακή αναζήτηση για την εύρεση ενός στοιχείου σε μια συλλογή, με ποιο τρόπο θα γίνει η αναζήτηση;
4. Σε ποια λογική σειρά βημάτων στηρίζεται ο αλγόριθμος ταξινόμησης με επιλογή;

6.6 Αναφορές - Βιβλιογραφία

Δικτυακός κόμβος υποστήριξης της γλώσσας Python, https://www.Python.org/ (τελευταία προσπέλαση 10/07/2015).

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 Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ. ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π.), Ιανουάριος 2015.

7

Διαχείριση αρχείων

7. Διαχείριση Αρχείων

Στόχοι

Μετά το τέλος της ενότητας, το 8ο Κεφάλαιο του Προγράμματος Σπουδών, θα μπορούμε να:

- δημιουργούμε απλά αρχεία
- χειριζόμαστε αρχεία κειμένου, μέσω βασικών εντολών (open, close, read, write) που χρησιμοποιεί η γλώσσα προγραμματισμού Python.

Λέξεις κλειδιά

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

Αρχείο κειμένου: Μια ακολουθία χαρακτήρων η οποία είναι αποθηκευμένη σε ένα μέσο μόνιμης αποθήκευσης, όπως ο σκληρός δίσκος, το DVD ή η μνήμη φλας (flash memory).

Κατάλογος: Μια συλλογή αρχείων ή και καταλόγων, που ονομάζεται επίσης και φάκελος.

Διαδρομή: Μια συμβολοσειρά η οποία προσδιορίζει ένα αρχείο.

Σχετική διαδρομή: Μία διαδρομή η οποία ξεκινάει από τον τρέχοντα κατάλογο.

Απόλυτη διαδρομή: Μια διαδρομή η οποία ξεκινάει από τον ανώτατο κατάλογο στο σύστημα αρχείων.

Διδακτικές ενότητες

7.1 Αρχεία κειμένου

Στα προηγούμενα κεφάλαια δημιουργήσαμε προγράμματα που τα δεδομένα τους, ως είσοδος από το πληκτρολόγιο ή ως έξοδος στην οθόνη, ήταν παροδικά, διαρκούσαν μόνο κατά τη διάρκεια εκτέλεσης του προγράμματος και χάνονταν όταν τελείωνε η εκτέλεση του προγράμματος. Αυτό είχε ως αποτέλεσμα σε κάθε εκτέλεση του προγράμματος, να απαιτείται να δίνουμε δεδομένα από την αρχή.

Ένας απλός τρόπος για να διατηρούν τα προγράμματα τα δεδομένα τους είναι μέσω της ανάγνωσης και εγγραφής αρχείων. Με αυτό τον τρόπο, μπορούμε να αποθηκεύσουμε μόνιμα τα δεδομένα ή να τα ανακτήσουμε, όποτε θέλουμε, ξεπερνώντας το εμπόδιο της προσωρινής μόνο αποθήκευσής τους στην κύρια μνήμη του υπολογιστή για το χρονικό διάστημα που διαρκεί η εκτέλεση του προγράμματος.

Ένα αρχείο κειμένου είναι μια ακολουθία χαρακτήρων αποθηκευμένη σε ένα μέσο μόνιμης αποθήκευσης όπως ο σκληρός δίσκος, το DVD_ROM, ή η μνήμη φλας (flash memory). Το αρχείο αυτό είναι απλό κείμενο και μπορούμε να το ανοίξουμε και να έχουμε πρόσβαση στα περιεχόμενά του χρησιμοποιώντας τις κατάλληλες εντολές της γλώσσας Python.

Προτού μπορέσουμε να εκτελέσουμε οποιαδήποτε λειτουργία σε ένα αρχείο, πρέπει πρώτα να το ανοίξουμε, ώστε να ενημερώσουμε το λειτουργικό σύστημα του υπολογιστή ότι πρόκειται να το χρησιμοποιήσουμε. Για να ανοίξουμε ένα αρχείο χρησιμοποιούμε την ενσωματωμένη συνάρτηση open() με όρισμα το όνομα του αρχείου. Η συνάρτηση μάς επιστρέφει ένα αντικείμενο του αρχείου που μπορούμε να χρησιμοποιήσουμε για να το προσπελάσουμε και να εκτελέσουμε διάφορες λειτουργίες σε αυτό.

X = open ('όνομα_αρχείου', 'τρόπος ανοίγματος')

Η συνάρτηση open() είναι ενσωματωμένη στην Python και δε χρειάζεται να φορτώσουμε κάποια βιβλιοθήκη. Δέχεται δυο ορίσματα:

Το όνομα του αρχείου ("words.txt") είναι, για παράδειγμα, το όνομα με το οποίο αναγνωρίζει το λειτουργικό σύστημα το αρχείο μας.

Ένα ειδικό σύμβολο (σημαία) που καθορίζει την κατάσταση ανοίγματος του αρχείου. Αν πρόκειται να το χρησιμοποιήσουμε για εγγραφή ή για ανάγνωση ("w" για εγγραφή, "r" για ανάγνωση).

Οι τιμές της παραμέτρου κατάστασης ανοίγματος φαίνονται στον παρακάτω πίνακα. Αν δε συμπληρωθούν, τότε θεωρείται ότι το αρχείο ανοίγει για ανάγνωση.

Παράδειγμα

```
>>> keimeno = open('words.txt')
```

στο αντικείμενο keimeno αντιστοιχεί το αρχείο words.txt. Στη συνέχεια μπορούμε να χρησιμοποιήσουμε το keimeno για να εκτελέσουμε λειτουργία ανάγνωσης στο αρχείο.

Επειδή τα πραγματικά ονόματα των αρχείων συχνά είναι μεγάλα και δύσχρηστα, συνηθίζουμε να χρησιμοποιούμε μέσα στον κώδικά μας απλούστερα ως προς την απομνημόνευση ονόματα. Μόλις εκτελεστεί η open(), δημιουργείται το αρχείο και μπορούμε να το δούμε από το λειτουργικό σύστημα.

Ορίσματα κατάστασης ανοίγματος	**Λειτουργία**
"r"	Ανάγνωση
"w"	Εγγραφή (Διαγραφή προηγούμενων περιεχομένων, αν υπάρχουν)
"a"	Προσθήκη (Διατήρηση προηγούμενων περιεχομένων)
"b"	Αρχείο Δυαδικής μορφής
"+"	Προσθήκη δεδομένων στο τέλος του αρχείου "r+": άνοιγμα αρχείου και για ανάγνωση και για εγγραφή.

Η γλώσσα Python υποστηρίζει κάποιες εγγενείς λειτουργίες για να επεξεργαστούμε τα αρχεία, τις οποίες ονομάζουμε **μεθόδους**. Μπορούμε να ανοίξουμε και να χρησιμοποιήσουμε αρχεία χρησιμοποιώντας τις αντίστοιχες μεθόδους για ανάγνωση read(), **readline**() ή για εγγραφή **write**() στο αρχείο. Η δυνατότητα να διαβάζουμε ή να γράφουμε στο αρχείο εξαρτάται από την κατάσταση ανοίγματος που έχουμε καθορίσει κατά το άνοιγμά του, με τη συνάρτηση open().

Τα σύμβολα κατάστασης ανοίγματος, που περιγράφονται στον παραπάνω πίνακα, ορίζονται ως δεύτερα ορίσματα μετά το όνομα του αρχείου. Κάθε φορά μπορεί να επι-

λεχθεί μόνο ένα, ανάλογα με τη λειτουργία που επιθυμούμε να εκτελέσουμε. Αν δεν επιλεχθεί κανένα, το αρχείο ανοίγει ως προεπιλογή μόνο για ανάγνωση (υπονοείται το όρισμα "r").

Όταν ολοκληρώσουμε με τις λειτουργίες που θέλουμε να εκτελεστούν στο αρχείο, καλούμε τη συνάρτηση **close()**, ώστε να δηλώσει ότι τελειώσαμε με τη χρήση του.

7.1.1 Εγγραφή και ανάγνωση αρχείου

Για να γράψουμε σε ένα αρχείο, πρέπει να το ανοίξουμε σε κατάσταση 'w', ορίζοντάς το ως δεύτερο όρισμα:

```
>>> keimeno = open('output.txt', 'w')
```

ανοίγουμε το αρχείο output.txt σε κατάσταση εγγραφής "w" και το αντιστοιχούμε στο αντικείμενο με όνομα keimeno

Αν το αρχείο υπάρχει ήδη και το ανοίξουμε σε κατάσταση εγγραφής, τότε διαγράφονται τα παλιά δεδομένα και το αρχείο είναι άδειο, οπότε πρέπει να είμαστε προσεκτικοί. Αν το αρχείο δεν υπάρχει, τότε δημιουργείται ένα νέο.

Η μέθοδος write() γράφει δεδομένα στο αρχείο.

```
>>> line1 = "Χαίρε Κόσμε ! \n"
#θέτουμε στη μεταβλητή line1 τη συμβολοσειρά 'Χαίρε Κόσμε !'
>>> keimeno.write(line1)
```

Γράφουμε το περιεχόμενο της μεταβλητής line1 στο αρχείο. Με το \n που θέσαμε στη συμβολοσειρά write αλλάζουμε γραμμή.

Το αντικείμενο αρχείου keimeno παρακολουθεί τη θέση που βρίσκεται και όταν ξανακαλέσουμε τη write, θα προσθέσει τα νέα δεδομένα μετά τη θέση αυτή.

```
>>> line2 = "Pythonistas.\n"
>>> keimeno.write(line2)
```

Όταν τελειώσουμε με το γράψιμο, θα πρέπει να κλείσουμε το αρχείο.

```
>>> keimeno.close()
```

Το όρισμα της write πρέπει να είναι μια συμβολοσειρά. Επομένως, αν θέλουμε να βάλουμε άλλες τιμές σε ένα αρχείο, πρέπει να τις μετατρέψουμε πρώτα σε συμβολοσειρές. Ο ευκολότερος τρόπος για να το κάνουμε αυτό είναι με την str:

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

Μπορούμε να χρησιμοποιήσουμε μια επανάληψη for για να κάνουμε διαδοχικές εγγραφές σε ένα αρχείο.

Ανάγνωση περιεχομένων ενός αρχείου

```
>>> keimeno = open('output.txt', 'r')
```

Τώρα η open() ανοίγει το αρχείο με παράμετρο "r", που σημαίνει ότι επιτρέπει μόνο την ανάγνωσή του.

Οι πιο διαδεδομένες μέθοδοι για διάβασμα είναι η read() και η readline(). Η πρώτη διαβάζει όλο το αρχείο και επιστρέφει ό,τι διαβάσει σε μια μεταβλητή, ενώ η δεύτερη διαβάζει χαρακτήρες από ένα αρχείο μέχρι να συναντήσει μια νέα γραμμή και επιστρέφει το αποτέλεσμα σε μια λίστα:

```
>>> keimeno.readline()
```

Δραστηριότητες ενότητας

Δραστηριότητα 1

Το παρακάτω πρόγραμμα αρχικά δημιουργεί ένα αρχείο με όνομα song.txt. Στη συνέχεια, γράφει μέσα σε ένα αρχείο την πρώτη στροφή από ένα αγαπημένο σας τραγούδι ανά στίχο και όταν ολοκληρώσει, εμφανίζει τα περιεχόμενο.

- Μελετήστε τον παρακάτω κώδικα και συμπληρώστε για κάθε εντολή το ανάλογο σχόλιο για το τι ακριβώς κάνει.
- Σημειώστε ποιο τμήμα του κώδικα χρησιμοποιείται για να γίνει η εγγραφή στο αρχείο και ποιο για να εμφανιστεί το περιεχόμενο στην οθόνη.
- Για ποιο λόγο χρησιμοποιούμε την επανάληψη for i in range(telos);

```
song=open("song.txt", "w")
telos=int(input("Δώσε τον αριθμό των στίχων"))
```

```
# γράφουμε ένα μήνυμα στο αρχείο.
for i in range(telos):
   print "Δώσε το",i+1, "ο στίχο"
   x = raw_input(" " )
   x=x + '\n'
song.write(x)

song.close()
song=open("song.txt", "r")
for line in song:
   print line
 song.close()
```

Δραστηριότητα 2

Δημιουργία ενός αρχείου με το όνομα "hmerologio_2015.txt", που περιέχει το ημερολόγιο του 2015 και εμφάνιση του περιεχόμενου του στην οθόνη με χρήση της ενσωματωμένης βιβλιοθήκης calendar.

Α) Μελετήστε τον κώδικα του προγράμματος που ακολουθεί. Πληκτρολογήστε τον κώδικα στο περιβάλλον IDLE της Python. Στη συνέχεια, εκτελέστε τον για να δείτε τι θα εμφανιστεί στην οθόνη. Βρείτε το αρχείο κειμένου "hmerologio_2015.txt" στο φάκελο που αποθηκεύετε τα προγράμματά σας και ανοίξτε το για να δείτε τα περιεχόμενά του. Τι παρατηρείτε; Συζητήστε στην τάξη πώς λειτουργεί το πρόγραμμα.

Β) Στη συνέχεια, κάντε τις απαραίτητες τροποποιήσεις, ώστε να αποθηκευτεί στο αρχείο "hmerologio_2016", το ημερολόγιο του 2016 και στη συνέχεια να εμφανιστεί στην οθόνη.

```
import calendar # εισάγουμε τη βιβλιοθήκη calendar
hmerologio=open("hmerologio_2017.txt", "w")
# ανοίγουμε το αρχείο hmerologio_2017 για εγγραφή


hmerologio.write ("Το παρακάτω ημερολόγιο έχει δημιουργηθεί με τη γλώσσα Python:\n\n") # γράφουμε ένα μήνυμα στο αρχείο.


a=calendar.TextCalendar(calendar.SUNDAY).formatyear(2017,2,1,1,2)  # θέτουμε στη μεταβλητή a όλο το ημερολόγιο του 2017.

hmerologio.write(a)
# γράφουμε το περιεχόμενο της μεταβλητής a στο αρχείο

hmerologio.write("\n\nPython: Κάνει τα πάντα...\n... αρκεί να ξέρεις πώς.\n( Μελέτησε το εγχειρίδιο χρήσης !)")
# γράφουμε στο αρχείο ένα πρόσθετο μήνυμα
hmerologio.close() # κλείνουμε το αρχείο.

# εμφάνιση του περιεχομένου του αρχείου
hmerologio=open("hmerologio_2017.txt","r")
# ανοίγουμε το αρχείο για ανάγνωση
contents=hmerologio.read()
print contents

# Εμφανίζουμε τα περιεχόμενο (το ημερολόγιο 2017) στην οθόνη.
hmerologio.close()
```

7.2 Ονόματα αρχείων και διαδρομές

Τα αρχεία είναι οργανωμένα σε καταλόγους / φακέλους, με κάθε πρόγραμμα που εκτελείται να έχει έναν κατάλογο, ο *τρέχων* κατάλογος, ο οποίος είναι ο προεπιλεγμένος κατάλογος για τις περισσότερες λειτουργίες. Για παράδειγμα, όταν ανοίγετε ένα αρχείο για διάβασμα, η Python το ψάχνει σε αυτό τον προεπιλεγμένο κατάλογο.

Η μονάδα λογισμικού os παρέχει συναρτήσεις για να μπορούμε να δουλέψουμε με αρχεία και καταλόγους (το "os" σημαίνει "operating system"). Η os.getcwd επιστρέφει το όνομα του τρέχοντος καταλόγου:

```
>>> import os
>>> cwd = os.getcwd()
>>> print cwd
```

Η cwd σημαίνει "current working directory". Το αποτέλεσμα σε αυτό το παράδειγμα είναι C:\Python27.

7.3 Δραστηριότητες κεφαλαίου

Δραστηριότητα

Τι πιστεύετε ότι θα συμβεί, όταν θα εκτελεστούν οι παρακάτω κώδικες. Τεκμηριώστε την άποψή σας.

A]

```
my_list = [i**2 for i in range(1,11)]
# Δημιουργεί μία λίστα τετραγώνων των αριθμών 1 - 10

f = open("output.txt", "w")

for item in my_list:
    f.write(str(item) + "\n")

f.close()
```

B)

```
f = open("bg.txt","w")
f.closed
# False
f.close()
f.closed
# True
```

C)

```
>>> logfile = open('test.log', 'w')
>>> logfile.write('test succeeded')
>>> logfile.close()
>>> print file('test.log').read()
test succeeded
>>> logfile = open('test.log', 'a')
>>> logfile.write('line 2')
>>> logfile.close()
>>> print file('test.log').read()
test succeededline 2
```

D)

```
g = open(' new_file ', 'w')
g.write('A new file begins' )
g.write(' ... today!\n')
g.close( )
```

7.4 Ερωτήσεις

1. Ποιες βασικές λειτουργίες-μεθόδους υποστηρίζει η γλώσσα προγραμματισμού Python, ώστε να μπορούμε να επεξεργαστούμε ένα αρχείο κειμένου;
2. Με ποια συνάρτηση ανοίγουμε ένα αρχείο κειμένου για ανάγνωση;
3. Πώς ανοίγουμε ένα αρχείο κειμένου για εγγραφή;
4. Με ποια συνάρτηση κλείνουμε ένα αρχείο;
5. Με ποια συνάρτηση γράφουμε περιεχόμενο σε ένα αρχείο;

7.5 Αναφορές - Βιβλιογραφία

Berkeley, (2015), http://python.berkeley.edu/.

Downey, A. (2012) «Think Python, How to think like a computer scientist», O' Reilly. http://www.greenteapress.com/thinkPython/. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από στα πλαίσια πτυχιακής εργασίας στο ΤΕΙ Λάρισας και είναι διαθέσιμο στο Διαδίκτυο.

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 International License. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από την Ελληνική κοινότητα του Ubuntu. http://ubuntu-gr.org/ και είναι διαθέσιμο στο Διαδίκτυο.

Αβούρης Ν., Κουκιάς Μ., Παλιουράς Β., Σγάρμπας Κ. (2013) «Εισαγωγή στους υπολογιστές με τη γλώσσα Python», Εκδόσεις Πανεπιστημίου Πατρών, Πάτρα.

Δικτυακός κόμβος υποστήριξης της γλώσσας Python, https://www.Python.org/ (τελευταία προσπέλαση 10/07/2015).

Λεβεντέας, Δ. (2010) «Taspython. Εκμάθηση Python Βήμα, 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 Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ. ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π.), Ιανουάριος 2015.

8

Εφαρμογές σε γλώσσα προγραμματισμού με χρήση API

8. Εφαρμογές σε γλώσσα προγραμματισμού με χρήση API

Στόχοι

Με τη μελέτη του κεφαλαίου, 13ο Κεφάλαιο Προγράμματος Σπουδών, 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.

Λέξεις κλειδιά

Διεπαφές Προγραμματισμού Εφαρμογών, Επικοινωνία Ανθρώπου–Υπολογιστή, σχεδιασμού της διεπαφής, βιβλιοθήκη Tkinter

Διδακτικές ενότητες

8.1 Διεπαφές Προγραμματισμού Εφαρμογών

Ένα πλεονέκτημα της γλώσσας προγραμματισμού Python είναι η συνεισφορά της μεγάλης κοινότητας προγραμματιστών που στηρίζει τη γλώσσα, με συλλογές έτοιμου λογισμικού για πάρα πολλές εφαρμογές. Τις συλλογές αυτές μπορείτε να τις φανταστείτε ως εργαλειοθήκες με έτοιμα εργαλεία, τα οποία μπορούμε να χρησιμοποιήσουμε μέσα στον κώδικά μας, αφού οι περισσότερες εργαλειοθήκες της Python ανήκουν στην κατηγορία του ελεύθερου λογισμικού. Οι συλλογές αυτές λογισμικού είναι γνωστές ως **Διεπαφές Προγραμματισμού Εφαρμογών – Application Programming Interfaces** (**APIs**) ή και ως βιβλιοθήκες λογισμικού.

Κάποια πολύ δημοφιλή APIs για Python είναι: η βιβλιοθήκη οπτικοποίησης δεδομένων Matplotlib, η SeaBorn που αποτελεί επέκταση του Matplotlib με γραφικές αναπαραστάσεις υψηλότερης ποιότητας και η ggplot που είναι βασισμένη στο σύστημα ggplot2 της γλώσσας προγραμματισμού R.

Στο κεφάλαιο αυτό θα ασχοληθούμε με τη βιβλιοθήκη Tkinter, η οποία είναι η πρότυπη βιβλιοθήκη της Python για την υλοποίηση γραφικών διεπαφών (GUI).

Επίσης με τη βοήθεια κατάλληλων APIs μπορούμε να αναπτύξουμε σχετικά εύκολα και γρήγορα εφαρμογές για την ανάσυρση και ανάκτηση πληροφοριών από τον παγκόσμιο ιστό, όπως κάνει μια ιστό-αράχνη (web crawler/spider) μιας μηχανής αναζήτησης.

Ωστόσο, ο βασικός σκοπός αυτού του κεφαλαίου δεν είναι να μάθετε ένα συγκεκριμένο API, ούτε φυσικά να αποστηθίσετε κάποιες βασικές συναρτήσεις των παραπάνω βιβλιοθηκών. Ο στόχος είναι να εξοικειωθούμε με την ιδέα ότι κατά την ανάπτυξη μιας εφαρμογής και αφού αναλύσουμε τις βασικές λειτουργίες της, θα πρέπει πρώτα να αναζητήσουμε αν έχουν ήδη υλοποιηθεί από άλλους προγραμματιστές και είναι μέρη κάποιου API, πριν προσπαθήσουμε να τις υλοποιήσουμε. Γενικά, στην ανάπτυξη εφαρμογών είναι καλή πρακτική, να χρησιμοποιούμε κατάλληλα ήδη γνωστά συγγενή με το θέμα τμήματα άλλων εφαρμογών, αποφεύγοντας να τα δημιουργούμε πάλι και εμείς, εκτός αν αυτό είναι απολύτως απαραίτητο από τη φύση της εφαρμογής.

8.2 Επικοινωνία ανθρώπου-υπολογιστή και διεπαφή χρήστη

Η Επικοινωνία Ανθρώπου – Υπολογιστή (Ε.Α.Υ.) - (Human Computer Interaction – HCI) γνωστή και ως *επικοινωνία ανθρώπου-μηχανής*, είναι το επιστημονικό πεδίο της πληροφορικής που μελετά την αλληλεπίδραση μεταξύ ανθρώπων (χρηστών) και υπολογιστών. Θεωρείται ως το σημείο τομής μεταξύ της Πληροφορικής, της γνωστικής ψυχολογίας, της κοινωνικής ψυχολογίας, της γλωσσολογίας, του βιομηχανικού σχεδιασμού και ακόμα περισσότερων ίσως γνωστικών πεδίων. Η αλληλεπίδραση μεταξύ χρηστών και υπολογιστών γίνεται στο επίπεδο της διεπαφής χρήστη (user interface), μέσω κατάλληλου λογισμικού και υλικού. Η επικοινωνία ανθρώπου υπολογιστή είναι πολύ ευρύτερος τομέας από τη διεπαφή χρήστη, αφού η διεπαφή χρήστη καλύπτει μόνον τα στοιχεία του υπολογιστή με τα οποία ο άνθρωπος έχει άμεση επαφή.

Ο τομέας της Επικοινωνίας Ανθρώπου-Υπολογιστή ασχολείται με το σχεδιασμό, την υλοποίηση, και την αξιολόγηση διαδραστικών υπολογιστικών συστημάτων προορισμένων για ανθρώπινη χρήση και τη μελέτη σημαντικών φαινομένων γύρω από αυτά (πηγή Association for Computer Machinery).

Για παράδειγμα, οι χαρακτήρες και τα αντικείμενα που εμφανίζονται από το λογισμικό στην οθόνη ενός ηλεκτρονικού υπολογιστή, αλλά και η είσοδος που δίνουν οι χρήστες μέσω περιφερειακών συσκευών, όπως το πληκτρολόγιο και το ποντίκι, αποτελούν αντικείμενο έρευνας της αλληλεπίδρασης ανθρώπου-υπολογιστή.

Ο βασικός στόχος της Ε.Α.Υ. είναι η αυξημένη ευχρηστία του συστήματος (Usability). Με τον όρο ευχρηστία ενός συστήματος, σύμφωνα με το διεθνές πρότυπο ISO 9241, περιγράφουμε την ικανότητα ενός συστήματος, να λειτουργεί αποτελεσματικά και αποδοτικά, ενώ παρέχει υποκειμενική ικανοποίηση στους χρήστες του. Η ευχρηστία του συστήματος επιτυγχάνεται, όταν η επικοινωνία ενός χρήστη με ένα σύστημα υπολογιστή γίνεται μέσω ενός περιβάλλοντος που χαρακτηρίζεται από:

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

Παραδείγματα - Εφαρμογές

Όταν σε μια εφαρμογή τα πλήκτρα ΑΠΟΘΗΚΕΥΣΗ (SAVE) και ΔΙΑΓΡΑΦΗ (DELETE) είναι πολύ κοντά, τότε πολύ εύκολα μπορεί να πάει κανείς με το ποντίκι στη λάθος επιλογή. Ακόμη και όταν υπάρχει πλαίσιο διαλόγου που ζητάει επιβεβαίωση της πράξης υπάρχει πρόβλημα, γιατί αυτά τα δύο πλαίσια είναι παρόμοια και εμφανίζονται και στις δύο επιλογές.

Η χρηστικότητα μιας διεπαφής σχετίζεται άμεσα με την πληροφορία που προσφέρει στο χρήστη συνειδητά ή και υποσυνείδητα. Ως πείραμα, κοιτάξτε τη γραφική απεικόνιση στους διακόπτες που ρυθμίζουν τα μάτια της ηλεκτρικής κουζίνας. Πολλές φορές, παρά την προσπάθεια του κατασκευαστή να απεικονίσει ποιος διακόπτης ανοίγει το αντίστοιχο μάτι, μπερδευόμαστε. Τα πράγματα θα ήταν πιο εύκολα, αν τα μάτια της κουζίνας τα τοποθετούσαμε σε διάταξη ενός πλάγιου παραλληλόγραμμου με το κάθε μάτι να βρίσκεται στην κάθε γωνία. Με αυτό το τρόπο, η πάνω σειρά των ματιών θα ήταν λίγο πιο αριστερά από την κάτω και η αντιστοίχιση των διακοπτών θα ήταν άμεση, με τον πρώτο διακόπτη να αντιστοιχεί στο μάτι στην πάνω αριστερή γωνία, το δεύτερο διακόπτη να αντιστοιχεί στο μάτι στην κάτω αριστερή γωνία, τον τρίτο διακόπτη στο

μάτι στην πάνω δεξιά γωνία και το τέταρτο διακόπτη στο μάτι στην κάτω δεξιά γωνία.

Άλλο ενδιαφέρον παράδειγμα είναι οι πόρτες που συναντάμε πολλές φορές σε καταστήματα. Πότε πρέπει να τις τραβήξουμε και πότε να τις σπρώξουμε; Το μήνυμα «ωθήσατε» παρότι μας πληροφορεί δεν μας οδηγεί στο να κάνουμε αυτόματα την ενέργεια. Θα ήταν πιο εύκολο αν, στην περίπτωση που έπρεπε να ωθήσουμε τη πόρτα, υπήρχε μια μικρή βιδωμένη μεταλλική πλάκα και όχι λαβή, ενώ στην περίπτωση που έπρεπε να την τραβήξουμε, αν υπήρχε μόνο λαβή.

Η αντίληψη κάποιας πληροφορίας από τη μεριά του χρήστη επηρεάζεται σημαντικά από τη μεταφορά που χρησιμοποιείται για την οπτική της αναπαράσταση, δηλαδή τη συγκεκριμένη απεικόνιση της συνήθως άγνωστης πληροφορίας, σε κάποιο οπτικό αντικείμενο, συνήθως γνώριμο, ώστε ο χρήστης να μπορεί εύκολα να καταλάβει το νόημα της πληροφορίας. Ως παράδειγμα γνώριμης μεταφοράς, σκεφθείτε τη χρήση του κόκκινου και του μπλε για ζεστό και κρύο στις βρύσες.

8.2.1 Γενικές Αρχές σχεδίασης διεπαφής

Κατά τη διάρκεια του σχεδιασμού της διεπαφής, ο σχεδιαστής θα πρέπει να μη ξεχνά ότι φτιάχνει ένα σύστημα το οποίο απευθύνεται κατ' αρχήν σε ανθρώπους μη σχετικούς και όχι σε έμπειρους χρήστες. Κάτω από αυτό το πρίσμα η σχεδιαζόμενη διεπαφή θα πρέπει να λαμβάνει υπόψη τις ανθρώπινες ιδιαιτερότητες. Για παράδειγμα, η επιλογή των χρωμάτων, των ετικετών, της τοποθέτησης των πλήκτρων και των λειτουργιών δε θα πρέπει να αντιβαίνουν στα χαρακτηριστικά του ανθρώπου. Αν η εφαρμογή απευθύνεται σε άτομα με ειδικές ανάγκες, τότε θα πρέπει να ληφθούν υπόψη τα ιδιαίτερα χαρακτηριστικά τους. Για παράδειγμα, εφαρμογές για άτομα με προβλήματα όρασης θα πρέπει να συμπεριλάβουν πολύ μεγαλύτερα εικονίδια, ηχητικά μηνύματα ανατροφοδότησης κ.ά.

Ο σκοπός της σχεδίασης μιας διεπαφής είναι η δημιουργία όσο το δυνατόν πιο εύχρηστων συστημάτων. Στο σχεδιασμό μιας διεπαφής θα πρέπει να ακολουθούνται ορισμένοι κανόνες. Διάφοροι επιστήμονες έχουν προτείνει ένα σύνολο κανόνων που θα πρέπει να τηρούνται. Οι βασικοί κανόνες του Shneiderman (Shneiderman and Plaisant, 2005) για το σχεδιασμό μιας διεπαφής είναι:

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 έμπειρων χρηστών:
   1) Συντμήσεις.
   2) Ειδικά πλήκτρα.
   3) Μακροεντολές.
3. Συνεχής ανατροφοδότηση της κατάστασης (informative feedback) του συστήματος. Για παράδειγμα, το σύστημα θα πρέπει να δείχνει την πορεία της τρέχουσας εργασίας, όπως: τι ποσοστό αρχείου έχει αντιγραφεί.
4. Οι "διάλογοι" χρήστη - υπολογιστή πρέπει να ολοκληρώνονται σε λίγα βήματα.
5. Πρόβλεψη για σφάλματα των χρηστών και χειρισμός σφαλμάτων.
6. Δυνατότητα αναίρεσης μιας ή περισσοτέρων ενεργειών με ευκολία.
7. Ο έλεγχος της αλληλεπίδρασης θα πρέπει να είναι από την πλευρά του χρήστη και όχι του συστήματος.
8. Το φορτίο βραχύχρονης μνήμης του χρήστη θα πρέπει να ελαχιστοποιηθεί (κανόνας των 7±2).

Θέματα για συζήτηση στην τάξη

- Ποια εικονίδια πιστεύετε ότι είναι τα πιο κατάλληλα να χρησιμοποιήσουμε σε μια εφαρμογή για να δηλώσουμε στο χρήστη ότι με αυτά μπορεί αντίστοιχα: να αποθηκεύσει την εργασία του, να την ανοίξει και να τη διαβάσει, να την εκτυπώσει;
- Αν σχεδιάζατε ένα παιχνίδι, ποιο εικονίδιο θα χρησιμοποιούσατε για τη λήψη βοήθειας;
- Η διάταξη QWERTY του πληκτρολογίου, πώς έχει προκύψει και σε ποιο βαθμό πιστεύετε ότι μας βοηθάει στην πληκτρολόγηση;

Εισαγωγική Δραστηριότητα

Σχετικά με το σχεδιασμό διεπαφών έχουν διατυπωθεί διάφορες προτάσεις όπως (Norman 1990):

- Είναι απαράδεκτο να χρειάζεται κάποιος πτυχίο Πληροφορικής για να δουλέψει αποδοτικά με έναν υπολογιστή.
- Οποιαδήποτε συσκευή, κατασκευή ή υπολογιστής μας αποτρέπει με κάποιο μήνυμα από το να κάνουμε κάτι, είναι κακοσχεδιασμένη.
- Οι άνθρωποι υπερέχουν των υπολογιστών σε συγκεκριμένους τομείς, ενώ οι υπολογιστές των ανθρώπων σε κάποιους άλλους.
- Σε μια καλοσχεδιασμένη διεπαφή, δε χρειάζεται κάποια προσπάθεια για να καταλάβουμε τι και πώς μπορούμε να το κάνουμε, καθώς και τι δεν μπορούμε να κάνουμε.
- Όταν μια διεπαφή και γενικά ένα εργαλείο κάνει καλά τη δουλειά του, μας γίνεται αόρατο. Ένα εργαλείο γίνεται αντιληπτό μόνο, όταν δεν κάνει καλά τη δουλειά του.

Συζητήστε στην τάξη τις προτάσεις αυτές. Βρείτε και περιγράψτε για κάθε πρόταση παραδείγματα από τη δική σας εμπειρία.

8.3 Η βιβλιοθήκη Tkinter για ανάπτυξη γραφικών διεπαφών GUI στην Python

Το παρακάτω πρόγραμμα αποτελεί την πρώτη μας εφαρμογή με γραφική διεπαφή στην Python.

```
import Tkinter as tk
import tkMessageBox

def showInformationBox():
   tkMessageBox.showinfo("Info","Nice Done")

def showOkCancelBox():
   tkMessageBox.askokcancel("okCancel","Remove all System Files?")

def showWarningBox():
   tkMessageBox.showwarning("warning","This is a warning")
```

```
def showErrorBox():
    tkMessageBox.showerror("error","Critical System Error #00000")
root = tk.Tk()
root.title("Computer Science")

message = tk.Label( root, text = "Introduction to Computer Science" )
message.pack()
message2 = tk.Label( root, text = "This is my first program" )
message2.pack()

buttonA = tk.Button( root, text = "Click here for Information", command = showInformationBox)
buttonA.pack()

buttonB = tk.Button( root, text = "Click here for Verification", command = showOkCancelBox)
buttonB.pack()

buttonC = tk.Button( root, text = "Click here for Warning", command = showWarningBox)
buttonC.pack()

buttonD = tk.Button( root, text = "Click here for Error Message", command = showErrorBox)
buttonD.pack()

root.mainloop()
```

Με τις εντολές

```
root = tk.Tk()
root.title("Computer Science")
```

δημιουργούμε ένα παράθυρο με τίτλο *Computer Science*., το οποίο δεν έχουμε όμως εμφανίσει ακόμα. Στο αντικείμενο αυτό προσθέτουμε άλλα αντικείμενα, όπως κείμενο (Label) και κάποια κουμπιά τα οποία, όταν πατήσουμε, εμφανίζονται συγκεκριμένα μηνύματα σε παράθυρα. Για παράδειγμα, η εντολή:

```
buttonA = tk.Button( root, text = "Click here for Information", com-mand = showInformationBox)
```

δημιουργεί ένα κουμπί το οποίο είναι αποθηκευμένο στη μεταβλητή-αντικείμενο buttonA. Το αντικείμενο αυτό έχει πάνω του το κείμενο *Click here for Information*, συνδέεται με το αντικείμενο – παράθυρο root και όταν πατηθεί εκτελείται η συνάρτηση *showInformationBox* την οποία έχουμε ορίσει παραπάνω. Οι συναρτήσεις που έχουμε αντιστοιχίσει στα κουμπιά εμφανίζουν διάφορα είδη παραθύρων, τα οποία μπορεί να έχουν ενημερωτικό σκοπό, να αναφέρουν κάποιο σοβαρό λάθος ή να ζητάνε επιβεβαίωση για κάποια ενέργεια.

Με την εντολή pack τοποθετείται το κουμπί εντός του παραθύρου, ενώ με την εντολή root.mainLoop() εμφανίζεται το παράθυρο.

Περισσότερες πληροφορίες μπορείτε να βρείτε στη βοήθεια του Tkinter που είναι διαθέσιμη στην ιστοσελίδα της Python.

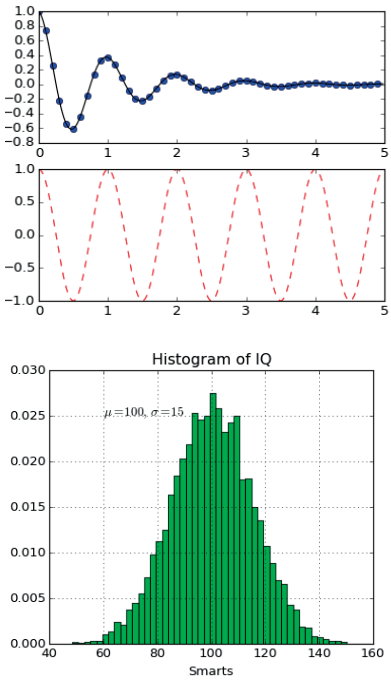
8.4 Δραστηριότητες κεφαλαίου

Δραστηριότητα 1

Εργαζόμενοι σε ομάδες, να πειραματιστείτε με έτοιμα παραδείγματα ανάκτησης και εξαγωγής πληροφορίας από ιστοσελίδες με τη βιβλιοθήκη Beautiful Soup http://www.crummy.com/software/BeautifulSoup/

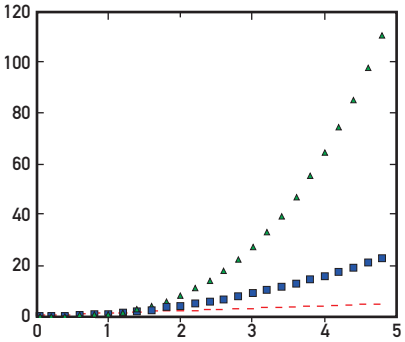
Δραστηριότητα 2

Να πειραματιστείτε με έτοιμα παραδείγματα οπτικοποίησης δεδομένων της βιβλιοθήκης Matplotlib από τη διεύθυνση http://matplotlib.org/examples. Πρώτα, να επιλέξετε μερικά παραδείγματα, σε συνεννόηση με τον καθηγητή σας, έτσι ώστε να χρησιμοποιήσετε αρκετές διαφορετικές κατηγορίες οπτικοποιήσεων, π.χ. γραφικές παραστάσεις συναρτήσεων, ιστογράμματα.



Δραστηριότητα 3

Μετά την εξοικείωσή σας με τη χρήση της βιβλιοθήκης Matplotlib, να αναπτύξετε, σε Python, ένα πρόγραμμα το οποίο, με χρήση αυτής της βιβλιοθήκης, να εμφανίζει, στο ίδιο διάγραμμα, διάφορες κλάσεις πολυπλοκότητας, ώστε να είναι εύκολη η σύγκριση της ασυμπτωτικής συμπεριφοράς τους διαγραμματικά, όπως φαίνεται στο παρακάτω παράδειγμα:



Δραστηριότητα 4. Εφαρμογή στα οικονομικά

Εξαγωγή οικονομικών πληροφοριών για διακύμανση μετοχών σε πραγματικό χρόνο από ιστοσελίδες χρηματιστηριακών ειδησεογραφικών ιστοσελίδων, με χρήση της βιβλιοθήκης εξαγωγής πληροφοριών από ιστοσελίδες Beautiful Soup. Μπορείτε να μελετήσετε την έτοιμη μικροεφαρμογή σε Python, Yahoo Finance Scraper.

http://www.Pythoncentral.io/Python-beautiful-soup-example-yahoo-finance-scraper/

Μπορείτε να χρησιμοποιήσετε και άλλο ελεύθερο λογισμικό για επεξεργασία ιστοσελίδων οικονομικού/χρηματιστηριακού ενδιαφέροντος, όπως η εφαρμογή scrapy, ή η βιβλιοθήκη της Python lxml όπως φαίνεται στο παρακάτω παράδειγμα

http://docs.Python-guide.org/en/latest/scenarios/scrape/

Δραστηριότητα 5. Δημιουργία εκπαιδευτικών ηλεκτρονικών παιχνιδιών

Αν σας ενδιαφέρει η δημιουργία εκπαιδευτικών ηλεκτρονικών παιχνιδιών, μπορείτε να βρείτε ενδιαφέρουσες βιβλιοθήκες λογισμικού για τη δημιουργία παιχνιδιών στη γλώσσα Python. Επισκεφτείτε, για παράδειγμα, το δικτυακό τόπο http://www.pygame.org, για να πληροφορηθείτε σχετικά με το project αυτό και για τις βιβλιοθήκες που μπορεί να χρησιμοποιήσει κανείς για να δημιουργήσει τα δικά του παιχνίδια. Δείτε μερικά παραδείγματα έτοιμων δημιουργιών. Αναζητήστε παρόμοια projects και βιβλιοθήκες στο Διαδίκτυο και παρουσιάστε τις πληροφορίες που βρήκατε στην τάξη.

8.5 Ερωτήσεις

1. Τι είναι η επικοινωνία ανθρώπου υπολογιστή;
2. Τι είναι η διεπαφή χρήστη;
3. Τι είναι η ευχρηστία συστήματος και ποια βασικά χαρακτηριστικά έχει ένα σύστημα με αυξημένη ευχρηστία;
4. Ποιοι είναι οι βασικοί κανόνες για το σχεδιασμό μιας διεπαφής;
5. Τι είναι οι διεπαφές προγραμματισμού εφαρμογών – Application Programming Interfaces (APIs). Αναφέρατε δύο δημοφιλή παραδείγματα για τη γλώσσα Python.

8.6 Αναφορές - Βιβλιογραφία

Ben Shneiderman and Catherine Plaisant. Designing the User In-terface. Pearson Education, Boston, 4th edition, 2005

Donald A. Norman. The Design of Everyday Things. Double-day/Currency, New York, NY, 1990. First published as *The Psychology of Everyday Things*.

Levitin, Anany (2007), The Design & Analysis of Algorithms. Pear-son Education.

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 International License. Το βιβλίο έχει μεταφραστεί και στα Ελληνικά από την Ελληνική κοινότητα του Ubuntu. http://ubuntu-gr.org/ και είναι διαθέσιμο στο Διαδίκτυο.

Αβούρης, Ν. (2000) «Εισαγωγή στην επικοινωνία ανθρώπου υπολογιστή», εκδ. Δίαυλος.

Σημειώσεις επικοινωνίας Ανθρώπου – Μηχανής (2005), Καθηγητής Γιάννης Ιωαννίδης, Δρ. Γιώργος Λέπουρας, διαθέσιμο στο http://cgi.di.uoa.gr/~ys08/EAM_simeiwseis.pdf (Πρόσβαση Ιούνιος 2015)

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 Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ. ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π.), Ιανουάριος 2015

Πηγές υλικού:

Ευρετήριο διεπαφών προγραμματισμού εφαρμογών της Python, https://pypi.Python.org/pypi

Βιβλιοθήκες προγραμματισμού για οπτικοποίηση δεδομένων http://matplotlib.org/users/pyplot_tutorial.html

http://stanford.edu/~mwaskom/software/seaborn/

https://pypi.Python.org/pypi/ggplot

http://bokeh.pydata.org/

Βιβλίο – Python for Data Analysis

http://dl.e-book-free.com/2013/07/Python_for_data_analysis.pdf

Python Scientific Lecture Notes

http://scipy-lectures.github.io/

Βιβλιοθήκη οπτικοποίησης της Google με Python

https://code.google.com/p/google-visualization-Python/

Παράδειγμα Python Beautiful Soup Example: Yahoo Finance Scraper

http://www.Pythoncentral.io/Python-beautiful-soup-example-yahoo-finance-scraper/

Η βιβλιοθήκη εξαγωγής πληροφοριών από ιστοσελίδες Beautiful Soup

http://www.crummy.com/software/BeautifulSoup/

http://www.gregreda.com/2013/03/03/web-scraping-101-with-Python/

Εξαγωγή πληροφοριών από ιστοσελίδες με άλλες βιβλιοθήκες

http://scrapy.org/

Ανάπτυξη Γραφικής Διεπαφής με τη βιβλιοθήκη Tkinter της Python

http://www.Python-course.eu/Python_tkinter.php

https://wiki.Python.org/moin/TkInter

Ευρετήριο όρων

Βάσει του ν. 3966/2011 τα διδακτικά βιβλία του Δημοτικού, του Γυμνασίου, του Λυκείου, των ΕΠΑ.Λ. και των ΕΠΑ.Σ. τυπώνονται από το ΙΤΥΕ - ΔΙΟΦΑΝΤΟΣ και διανέμονται δωρεάν στα Δημόσια Σχολεία. Τα βιβλία μπορεί να διατίθενται προς πώληση, όταν φέρουν στη δεξιά κάτω γωνία του εμπροσθόφυλλου ένδειξη «ΔΙΑΤΙΘΕΤΑΙ ΜΕ ΤΙΜΗ ΠΩΛΗΣΗΣ». Κάθε αντίτυπο που διατίθεται προς πώληση και δεν φέρει την παραπάνω ένδειξη θεωρείται κλεψίτυπο και ο παραβάτης διώκεται σύμφωνα με τις διατάξεις του άρθρου 7 του νόμου 1129 της 15/21 Μαρτίου 1946 (ΦΕΚ 1946,108, Α').

Κωδικός βιβλίου: 0-24-0615
ISBN 978-960-06-5141-6