ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

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

# Προγραμματισμός Υπολογιστών

## Λύσεις Ασκήσεων Βιβλίου Μαθητή και Τετραδίου Εργασιών Μαθητή

Γ΄ Τάξη ΕΠΑ.Λ.

**ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ
«ΔΙΟΦΑΝΤΟΣ»

# Προγραμματισμός Υπολογιστών

## Γ΄ Τάξη ΕΠΑ.Λ.

## Λύσεις Ασκήσεων Βιβλίου Μαθητή και Τετραδίου Εργασιών Μαθητή

Γ' και Δ' τάξη ημερησίων και εσπερινών ΕΠΑ.Λ.
Τομέας Πληροφορικής

**ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ**

| | |
|---|---|
| ΠΡΟΕΔΡΟΣ: | **Κουζέλης Γεράσιμος,** Καθηγητής ΕΚΠΑ |
| ΕΠΙΣΤΗΜΟΝΙΚΑ ΥΠΕΥΘΥΝΟΣ: | **Τσαπέλας Θεοδόσιος** |
| ΣΥΓΓΡΑΦΙΚΗ ΟΜΑΔΑ | **Αράπογλου Αριστείδης,** Εκπαιδευτικός Πληροφορικής<br>**Βραχνός Ευριπίδης,** Εκπαιδευτικός Πληροφορικής<br>**Κανίδης Ευάγγελος,** Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής<br>**Λέκκα Δήμητρα,** Εκπαιδευτικός Πληροφορικής<br>**Μακρυγιάννης Παναγιώτης,** Εκπαιδευτικός Πληροφορικής<br>**Μπελεσιώτης Βασίλειος,** Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής<br>**Παπαδάκης Σπυρίδων,** Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής<br>**Τζήμας Δημήτριος,** Εκπαιδευτικός Πληροφορικής |
| ΕΠΙΜΕΛΕΙΑ - ΣΥΝΤΟΝΙΣΜΟΣ ΟΜΑΔΑΣ | **Κανίδης Ευάγγελος,** Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής<br>**Μπελεσιώτης Βασίλειος,** Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής |
| ΦΙΛΟΛΟΓΙΚΗ ΕΠΙΜΕΛΕΙΑ | **Ευφροσύνη Δεληγιάννη,** Σχολική Σύμβουλος, ΠΕ02 |
| ΕΠΙΤΡΟΠΗ ΚΡΙΣΗΣ | **Βογιατζής Ιωάννης,** Επίκουρος Καθηγητής, Α.Τ.Ε.Ι. Αθηνών<br>**Εφόπουλος Βασίλειος,** Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής<br>**Κωτσάκης Σταύρος,** Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής |
| ΠΡΟΕΚΤΥΠΩΤΙΚΕΣ ΕΡΓΑΣΙΕΣ | **ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΕΚΔΟΣΕΩΝ «ΔΙΟΦΑΝΤΟΣ»** |

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

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

# Προγραμματισμός Υπολογιστών

## Γ΄ Τάξη ΕΠΑ.Λ.

## Λύσεις Ασκήσεων Βιβλίου Μαθητή και Τετραδίου Εργασιών Μαθητή

Γ' και Δ' τάξη ημερησίων και εσπερινών ΕΠΑ.Λ.
Τομέας Πληροφορικής

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ
«ΔΙΟΦΑΝΤΟΣ»

## Εισαγωγικό σημείωμα

Αγαπητέ μαθητή,
Το τεύχος που κρατάς στα χέρια σου περιέχει ενδεικτικές λύσεις των δραστηριοτήτων του βιβλίου σου «Προγραμματισμός Υπολογιστών - Διδακτικό Υλικό Μαθητή Γ' ΕΠΑΛ», καθώς και του τετραδίου μαθητή. Οι λύσεις που παρουσιάζονται στο τεύχος αυτό είναι ενδεικτικές, χωρίς αυτό να σημαίνει ότι σε περιορίζουν να δημιουργήσεις τις δικές σου τεκμηριωμένες λύσεις. Εξάλλου ένα αλγοριθμικό πρόβλημα, σε αρκετές περιπτώσεις, μπορεί να λυθεί με πολλούς διαφορετικούς τρόπους. Αρκεί η λύση που τελικά βρήκες και προτείνεις, να μπορεί να τεκμηριωθεί, αλλά και να επαληθευτεί ως προς την ορθότητά της. Ένας τρόπος επαλήθευσης είναι, δίνοντας διάφορες χαρακτηριστικές τιμές ως είσοδο στον αλγόριθμο και στη συνέχεια εκτελώντας τον, να δίνει στην έξοδο τα αναμενόμενα αποτελέσματα σε εύλογο χρόνο. Επιπρόσθετα, σε μια σύγχρονη γλώσσα προγραμματισμού οι λύσεις μπορεί να διαφέρουν και ως προς το πώς εκφράζονται. Στη γλώσσα προγραμματισμού Python, ο αλγόριθμος που λύνει ένα πρόβλημα μπορεί να εκφραστεί με διαφορετικούς τρόπους, ακολουθώντας τις δυνατότητες που μας παρέχουν οι συντακτικοί κανόνες της συγκεκριμένης γλώσσας προγραμματισμού.
Στις ενδεικτικές λύσεις που ακολουθούν και στις επόμενες σελίδες, καταβλήθηκε προσπάθεια ώστε ο κώδικας του προγράμματος, που εκφράζει την ενδεικτική λύση, να είναι κατανοητός και ευανάγνωστος χωρίς να εξαντλεί τις δυνατότητες διαφορετικής διατύπωσης μιας δήλωσης ή έκφρασης που μας δίνει η γλώσσα Python. Επιπρόσθετα έχοντας ως σκοπό να αναδειχθεί κυρίως η αλγοριθμική λύση του προβλήματος, η οποία αποτελεί και το βασικό σκοπό του μαθήματος, οι λύσεις που παρουσιάζονται αξιοποιούν ένα πολύ περιορισμένο αριθμό απλών συναρτήσεων και μεθόδων, από το πλούσιο σύνολο που διαθέτει η γλώσσα Python, με τις βασικές ενσωματωμένες βιβλιοθήκες της και το οποίο προσδοκούμε ότι θα έχει κατακτήσει ένας μαθητής της Γ' ΕΠΑΛ με την ολοκλήρωση του Τομέα Πληροφορικής.

Η αξιοποίηση του τεύχους αυτού μπορεί να επιτευχθεί με την ορθή χρήση του. Πριν μελετήσεις την προτεινόμενη ενδεικτική λύση, θα πρέπει να έχεις προσπαθήσει μόνος σου ή σε συνεργασία με μια ομάδα συμμαθητών σου, να βρεις τη δική σου λύση – απάντηση στη δραστηριότητα - άσκηση. Σε καμία περίπτωση δεν προτείνουμε να το χρησιμοποιήσεις για να αποστηθίσεις τις έτοιμες λύσεις ή για να ανατρέξεις γρήγορα στη προτεινόμενη λύση με την πρώτη δυσκολία που θα

αντιμετωπίσεις, προσπαθώντας να λύσεις τις δραστηριότητες του βιβλίου μαθητή και τετραδίου εργασίας. Αυτό θα έμοιαζε σαν να λύνεις ένα σταυρόλεξο σε ένα περιοδικό αντιγράφοντας άμεσα τις λύσεις που σου δίνει στη τελευταία του σελίδα. Μα τότε θα έκανες εξάσκηση στη γραφή και σίγουρα όχι στον προγραμματισμό.
Επομένως, σου προτείνουμε να ανατρέξεις για βοήθεια ή επαλήθευση στο τεύχος αυτό, αφού έχεις πειραματιστεί αρκετή ώρα για να ανακαλύψεις τη λύση του προβλήματος. Μελετώντας στη συνέχεια, αν χρειαστεί, την προτεινόμενη λύση, κατανοώντας τι κάνει ο προτεινόμενος κώδικας, θα κτίσεις σταδιακά τις απαραίτητες γνώσεις για να λύνεις παρόμοια προβλήματα. Η επιμονή στο να βρει κανείς τη λύση ενός αλγοριθμικού προβλήματος είναι βασική διαδικασία για να μάθει να προγραμματίζει τους υπολογιστές. Εξ άλλου, ας μη ξεχνάμε και τη χαρά που αισθανόμαστε ότι τα καταφέραμε, όταν βρούμε τη λύση ενός προβλήματος μετά από κόπο και φωνάζουμε «ΤΟ ΒΡΗΚΑ!».
«ΕΥΡΗΚΑ, ΕΥΡΗΚΑ!», όπως λέγεται ότι αναφώνησε ο αρχαίος Έλληνας Μαθηματικός, Φυσικός, Μηχανικός και εφευρέτης Αρχιμήδης, όταν ανακάλυψε μετά από πολύ χρόνο και κόπο τη γνωστή Αρχή περί της άνωσης.
Αν χρειαστεί να ανατρέξεις στην ενδεικτική λύση, γιατί δυσκολεύεσαι να καταλάβεις πώς πρέπει να ξεκινήσεις, διάβασε μία φορά ακόμη την εκφώνηση και βεβαιώσου - σημείωσε τι ζητά η άσκηση. Μετά, σου προτείνουμε να δεις την αρχή μόνο της προτεινόμενης λύσης και να προσπαθήσεις τη συνέχεια μόνος σου. Εναλλακτικά, μπορείς να δεις την προτεινόμενη λύση και να προσπαθήσεις να την εκτελέσεις στο χαρτί, γραμμή προς γραμμή, με διαφορετικά δεδομένα για να κατανοήσεις καλύτερα τη λειτουργία του προτεινόμενου κώδικα. Προϋπόθεση είναι να έχεις κατανοήσει τι σου ζητάει το πρόβλημα ή η δραστηριότητα και να έχεις υπολογίσει πρώτα, ποια θα είναι τα αποτελέσματα με τα συγκεκριμένα δεδομένα που δοκιμάζεις, ώστε να ξέρεις τι να περιμένεις ως αποτέλεσμα μετά την εκτέλεση του προγράμματος. Στη συνέχεια, αν χρειαστεί μπορεί να πληκτρολογήσεις τον κώδικα στον υπολογιστή, στο προγραμματιστικό περιβάλλον της γλώσσας Python και να επαληθεύσεις τα αποτελέσματα με τη βοήθεια του διερμηνευτή. Η διαδικασία της μελέτης σου μπορεί να θεωρηθεί ολοκληρωμένη, αν μπορείς να λύσεις παρόμοια προβλήματα με μικρές τροποποιήσεις, στον κώδικα του προγράμματος.
Το μάθημα του Προγραμματισμού Υπολογιστών έχει ένα πολύ ιδιαίτερο χαρακτηριστικό: μπορείς μαζί με τον υπολογιστή, με το δικό σου ρυθμό να μαθαίνεις και να επαληθεύεις τις λύσεις σου με το διερμηνευτή της γλώσσας προγραμματισμού. Προσπαθώντας να δημιουργήσεις τον κατάλληλο κώδικα για να λύσεις τα προτεινόμενα προβλήματα του βιβλίου σου, μπορείς να πειραματιστείς με τη γλώσσα

προγραμματισμού Python, να ανακαλύψεις τις δικές σου λύσεις, να τις γράψεις σε κώδικα και να κάνεις τον υπολογιστή να τις εκτελεί, επιστρέφοντάς σου στην οθόνη τα αναμενόμενα αποτελέσματα. Η ανάπτυξη εφαρμογών σε υπολογιστές δεν είναι μια ατομική διεργασία. Πολλές φορές, ακόμα και οι έμπειροι προγραμματιστές υπολογιστών χρειάζεται να συνεργαστούν, να ανταλλάξουν απόψεις και εμπειρίες. Είναι ιδιαίτερα χρήσιμο να «κτίσεις» σε συνεργασία με μια ομάδα συμμαθητών σου τη λύση των προβλημάτων. Όσο μαθαίνεις σταδιακά να προγραμματίζεις τον υπολογιστή, τόσο θα απολαμβάνεις τη μοναδική αίσθηση που σου χαρίζει ο προγραμματισμός υπολογιστών: τη χαρά να *δημιουργείς* άμεσα εκτελέσιμα έργα. Για να το επιτύχεις αυτό χρειάζεται υπομονή, χρόνος και επιμονή. Σημαντική είναι η συνεργασία με τους συμμαθητές σου, αλλά και η καθοδήγηση, όπου χρειαστεί, από τον καθηγητή σου. Με τη σειρά μας ελπίζουμε, όταν διαθέτεις χρόνο, να μην περιοριστείς στις προτεινόμενες δραστηριότητες του Βιβλίου Μαθητή και του Τετραδίου Εργασίας, αλλά να δημιουργείς σταδιακά τα δικά σου έργα, λύνοντας και άλλα παρόμοια προβλήματα με τις δυνατότητες άμεσου πειραματισμού και ανατροφοδότησης, που σου δίνει το σύγχρονο προγραμματιστικό περιβάλλον της γλώσσας Python.

Τα μέλη της συγγραφικής ομάδας

**Σημείωμα**

Το παρόν διδακτικό υλικό σχετίζεται με το Α.Π.Σ. του μαθήματος Προγραμματισμός ΑΠΣ_Τομέα_Πληρ_ΕΠΑΛ (2015) και το σχετικό Ωρολόγιο Πρόγραμμα (ΩΠ_Μαθημάτων_ΕΠΑΛ, 2015).

Η όλη ανάπτυξη και των τεσσάρων βιβλίων του μαθήματος (ΒΜ, ΤΕΜ, ΒΚ, Λύσεις μαθητή), έγινε χωρίς αμοιβή.

Το βιβλίο αυτό, βασίζεται και περιορίζεται στη διδακτέα - εξεταστέα ύλη που καθορίζεται στην ΥΑ Φ6/160716/Δ4 (2016).

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

## Γενικές οδηγίες μαθήματος

### Η φιλοσοφία του μαθήματος

Οι δραστηριότητες που δίνονται τόσο στο Βιβλίο Μαθητή (ΒΜ) όσο και στο Τετράδιο Εργασίας Μαθητή (ΤΕΜ), ακολουθούν μια διερευνητική/ανακαλυπτική προσέγγιση, σύμφωνα με την οποία οι μαθητές κατακτούν τους διδακτικούς στόχους του μαθήματος μέσα από τους αναγκαίους πειραματισμούς με το προγραμματιστικό περιβάλλον.

Σε όλες τις ασκήσεις-δραστηριότητες οι αλγόριθμοι διατυπώνονται στη γλώσσα Python.

Η γλώσσα προγραμματισμού Python χαρακτηρίζεται από την πληθώρα εντολών/συναρτήσεων και βιβλιοθηκών που διαθέτει για την αντιμετώπιση ποικίλων προβλημάτων. Οι ασκήσεις και δραστηριότητες που υπάρχουν στο ΒΜ, στο ΤΕΜ και στο Βιβλίο Καθηγητή (ΒΚ), μπορούν να λυθούν με χρήση των εντολών που περιέχουν τα σχολικά βιβλία Προγραμματισμός Υπολογιστών και Προγραμματισμός της Β' και Γ' τάξης αντίστοιχα. Έτσι, ο μαθητής πρέπει να γνωρίζει και να μπορεί να χρησιμοποιήσει τις παρακάτω συναρτήσεις και μεθόδους οι οποίες αναφέρονται σε αυτά.

Ενδεικτικά:

| | |
|---|---|
| **Γενικές συναρτήσεις:** | len, range, type, id, input, raw_input |
| **Μετατροπές τύπων:** | str, int, float, list, bool |
| **Μαθηματικές συναρτήσεις:** | abs, pow, divmod, math.sqrt |
| **Μέθοδοι λίστας:** | List.append, List.insert, List.pop |
| **Μέθοδοι αρχείου:** | open, File.close, File.read, File.readline |
| **Λειτουργίες λεξικού:** | dict, del |

** Στον πίνακα, όπου **file** είναι ένα αρχείο και **list** μια λίστα.*

**Σκοπός** του μαθήματος είναι η ανάπτυξη της αλγοριθμικής και υπολογιστικής σκέψης του μαθητή με όχημα τη γλώσσα Python και όχι η σε βάθος εκμάθηση της Python με όλες τις δυνατότητες που είναι ενσωματωμένες σε αυτήν ή παρέχονται μέσω των βιβλιοθηκών που μπορεί να βρει και να χρησιμοποιήσει κάποιος. Σε εκπαιδευτικό περιβάλλον, η επίλυση μιας άσκησης και ο τρόπος με τον οποίο, φτάνει ο μαθητής

στη λύση, έχει πολύ συχνά περισσότερη σημασία από το τελικό αποτέλεσμα της λύσης. Παράλληλα, δεν πρέπει να αγνοηθεί ότι κάθε επιστημονικά ορθή λύση είναι αποδεκτή, εφόσον συνάδει με το πλαίσιο της εκφώνησης.

Για το λόγο αυτό, κάθε άσκηση θα πρέπει να οριοθετείται από το πεδίο επίλυσής της, δίνοντας την ανάλογη προσοχή στη λύση που θα δώσουμε. Η οριοθέτηση αυτή μπορεί να γίνεται είτε από την εκφώνηση της άσκησης είτε προφορικά από τον εκπαιδευτικό, αν η άσκηση λύνεται μέσα στην τάξη ως δραστηριότητα. Ενδεικτικά: ο εκπαιδευτικός μπορεί να επεκταθεί στη διδασκαλία του και να επιτρέψει τη χρήση συναρτήσεων και μεθόδων που δεν αναφέρονται στα σχολικά βιβλία της Β' και Γ' τάξης, όπως οι sorted, reverse, max, min, sum, avg, μέθοδος join κ.ά., ή αντίθετα, να απαγορεύσει τη χρήση άλλων συναρτήσεων (εκτός σχολικών βιβλίων) ή ακόμα και κάποιου τελεστή, όπως για παράδειγμα του τελεστή in, ο οποίος θεωρείται γνωστός.

**Σημαντικές Παρατηρήσεις:**

- Η έκδοση της Python που χρησιμοποιείται στο μάθημα είναι η 2 και πιο συγκεκριμένα οποιαδήποτε μεταγενέστερη έκδοση της 2.7.10.
- Αν θέλετε να γράψετε ελληνικούς χαρακτήρες μέσα σε ένα πρόγραμμα -είτε ως σχόλια είτε ως αλφαριθμητικά- θα πρέπει η πρώτη γραμμή του αρχείου python να είναι η παρακάτω:
  `# -*- coding: utf-8 -*-`
- Το περιβάλλον προγραμματισμού που προτείνεται, είναι το IDLE που συνοδεύει την Python 2 και με αυτό συμφωνεί και ο χρωματισμός του κώδικα στο παρόν υλικό. Μπορείτε να χρησιμοποιήσετε όμως και άλλα περιβάλλοντα, όπως είναι για παράδειγμα το PyScripter.
- Αν χρησιμοποιείτε Linux, η Python που διαθέτει, δεν περιέχει το IDLE, οπότε θα πρέπει να αναζητήσετε και εγκαταστήσετε το αρχείο idle-python2.7.

Σημειώνεται ότι πολλές ασκήσεις, ειδικά του Τετραδίου Εργασιών Μαθητή, έχουν παρόμοια δομή με αυτήν που τηρείται στην Τράπεζα Θεμάτων Διαβαθμισμένης Δυσκολίας, για το μάθημα Εισαγωγή στις Αρχές της επιστήμης των Η/Υ Β' ΓΕΛ/ΕΠΑΛ για λόγους συνέχειας και ομοιομορφίας.

**Παρατηρήσεις για τις λύσεις των δραστηριοτήτων** (που ακολουθούν)

Σε όλες τις δραστηριότητες/ασκήσεις αλλά και όπου ζητείται να εισαχθεί από το πληκτρολόγιο μια λίστα αντικειμένων χωρίς να αναφέρεται το πλήθος των στοιχείων που θα εισαχθούν, αλλά και ούτε κάποιο κριτήριο διακοπής της εισαγωγής, μπορείτε να θεωρήσετε ότι η εισαγωγή των δεδομένων σταματάει όταν ο χρήστης δώσει την τιμή None.

Σε κάποιες περιπτώσεις χρησιμοποιούνται συναρτήσεις χωρίς την ανάλυσή τους και αυτό γίνεται διότι έχουν ήδη οριστεί σε προηγούμενες δραστηριότητες και έτσι απλά αντιγράφονται ξανά. Σε μια τέτοια περίπτωση, θα πρέπει να ανατρέξετε στην αντίστοιχη δραστηριότητα για την αναλυτική μορφή της συνάρτησης, διαδικασία που άλλωστε χαρακτηρίζει την *επαναχρησιμοποίηση* κώδικα.

Η χρήση της input για την εισαγωγή δεδομένων, δεν αποκλείει την εισαγωγή αλφαριθμητικών. Για να γίνει όμως, σωστά θα πρέπει αυτά να περικλείονται σε εισαγωγικά (quotes), όπως ακριβώς αναφέρονται και μέσα στον κώδικα. Έτσι, αν για παράδειγμα θέλουμε να δώσουμε το όνομα *Τομέας Πληροφορικής*, θα το δώσουμε ως εξής: *"Τομέας Πληροφορικής"*.

Αν θέλουμε να γράψουμε μια πολύ μεγάλη εντολή σε περισσότερες από μια γραμμές, τότε χρησιμοποιούμε το σύμβολο \ .

## Περιεχόμενα

# Μέρος Ι

## ΚΕΦΑΛΑΙΟ 1 Από το πρόβλημα στην ανάπτυξη αλγορίθμου

Το κεφάλαιο αυτό είναι εκτός διδακτέας και εξεταστέας ύλης και δεν αποτελεί αντικείμενο αυτοτελούς εξέτασης.

## ΚΕΦΑΛΑΙΟ 2 Ανάπτυξη προγράμματος

Το κεφάλαιο αυτό είναι εκτός διδακτέας και εξεταστέας ύλης και δεν αποτελεί αντικείμενο αυτοτελούς εξέτασης.
Οι έννοιες του θεωρούνται γνωστές από τη Β' τάξη και μπορούν να χρησιμοποιηθούν στη λύση δραστηριοτήτων και ασκήσεων χωρίς να αποτελούν αντικείμενο αυτοτελούς εξέτασης.

# ΚΕΦΑΛΑΙΟ 3 Βασικά στοιχεία γλώσσας προγραμματισμού

## Εισαγωγικό σημείωμα

*Το κεφάλαιο αυτό είναι **εντός** διδακτέας-εξεταστέας ύλης. Στο κεφάλαιο αυτό δεν υπάρχουν ασκήσεις- δραστηριότητες στο Βιβλίο Μαθητή.*

## Λύσεις Δραστηριοτήτων Τετραδίου Εργασιών Μαθητή

**Δραστηριότητα 1.**

Η απάντηση στη δραστηριότητα αυτή, αναφέρεται στο Τετράδιο Εργασίας Μαθητή.

**Δραστηριότητα 2.** Εξάσκηση στο περιβάλλον IDLE/Python

Να ανοίξετε το περιβάλλον του IDLE-Shell (>>>) και να κάνετε τις παρακάτω ενέργειες:

• Πληκτρολογήστε help() και πατήστε ENTER. Τι κάνει αυτή η εντολή;

```
Type "copyright", "credits" or "license()" for more information.
>>> help()

Welcome to Python 2.7!  This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/2.7/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics".  Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".
```

• Πληκτρολογήστε: keywords. Ποιο είναι το αποτέλεσμα της εντολής αυτής;

```
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

and                 elif                if                  print
as                  else                import              raise
assert              except              in                  return
break               exec                is                  try
class               finally             lambda              while
continue            for                 not                 with
def                 from                or                  yield
del                 global              pass

help>
```

- Πληκτρολογήστε print. Διαβάστε προσεκτικά τη σύνταξη και τη λειτουργία της εντολής (statement) print και βγείτε από το περιβάλλον της βοήθειας, πατώντας ENTER.

help> print

- Χρησιμοποιήστε την εντολή print για να εμφανίσετε στην οθόνη το μήνυμα: *Καλημέρα φίλοι μου, μόλις γνώρισα το περιβάλλον του διερμηνευτή IDLE.* Παρατήρηση: να μη ξεχάσετε να βάλετε εισαγωγικά, όπου χρειάζονται.

```
>>> print "Καλημέρα φίλοι μου, μόλις γνώρισα το περιβάλλον του διερμηνευτή IDLE"
Καλημέρα φίλοι μου, μόλις γνώρισα το περιβάλλον του διερμηνευτή IDLE
>>>
```

- Έχει διαφορά αν χρησιμοποιήσετε μονά ή διπλά εισαγωγικά; Μπορείτε να ξεκινήσετε με μονά εισαγωγικά και να κλείσετε με διπλά;

Όχι, Όχι

- Παρατηρήστε την οθόνη του IDLE. Γιατί κάποιες εντολές και λέξεις είναι με διαφορετικό χρώμα;

Συζητήστε

- Να γράψετε μια απλή γραμμή σχολίων της αρεσκείας σας, δίνοντας προσοχή να ξεκινάει με #. Τα σχόλια είναι πολύ χρήσιμα για να τεκμηριώσετε τον κώδικα σας.

>>> # αυτό είναι ένα σχόλιο

**Δραστηριότητα 3.** Τύποι δεδομένων (ερωτήσεις αντιστοίχισης)

Σε ποιο τύπο δεδομένων στη γλώσσα προγραμματισμού Python αντιστοιχούν οι τιμές της αριστερής στήλης του παρακάτω πίνακα; Να συνδέσετε κατάλληλα τις τιμές της αριστερής στήλης με το σωστό τύπο δεδομένων της δεξιάς στήλης. Να σημειωθεί ότι περισσότερες από μία επιλογές της στήλης Α αντιστοιχούν σε κάποια από τις επιλογές της στήλης Β.

| **Στήλη Α** (Τιμή) | **Στήλη Β** (Τύπος δεδομένων) |
|---|---|
| 1. -27<br>2. 35.7<br>3. 'False'<br>4. True<br>5. "432.12"<br>6. 'μεταβλητή'<br>7. 12 / 2<br>8. 20 % 3 | Α. int (ακέραια)<br><br>Β. float (κινητής υποδιαστολής)<br><br>Γ. string (συμβολοσειρά)<br><br>Δ. bool (λογική) |

**Απάντηση**

1--Α, 2--Β, 3--Γ, 4--Δ, 5--Γ, 6--Γ, 7--Α, 8--Α

**Δραστηριότητα 4. (Σωστό-Λάθος). Δίνοντας όνομα σε μία μεταβλητή**

Ποιο από τα παρακάτω υποψήφια ονόματα της αριστερής στήλης του πίνακα δεν είναι αποδεκτό ως όνομα μεταβλητής. Σημειώστε με **Λ** (Λάθος) αυτά που πιστεύετε ότι δεν είναι αποδεκτά ονόματα και με **Σ** (Σωστό) εκείνα που πιστεύετε ότι είναι αποδεκτά.

| Υποψήφια ονόματα | Σ/Λ |
|---|---|
| x!b | Λ |
| Metavliti3 | Σ |
| Metavliti+3 | Λ |
| Kila 2 | Λ |
| mikos_1 | Σ |
| 245 | Λ |
| 1onoma | Λ |
| Print | Λ |

Δοκιμάστε στο περιβάλλον της Python να δώσετε ως ονόματα μεταβλητών τα παραπάνω, ελέγχοντας τις απαντήσεις σας. Στη συνέχεια, δικαιολογήστε τις απαντήσεις σας, απαριθμώντας τους βασικούς κανόνες που πρέπει να ακολουθούμε για να δώσουμε ένα όνομα σε μια μεταβλητή.

**Απάντηση**

Η απάντηση βρίσκεται στην παράγραφο 3.2 του βιβλίου μαθητή

**Δραστηριότητα 5.** Εμβάθυνση στις μεταβλητές και τύπους δεδομένων.

Να καταγράψετε τι πιστεύετε ότι θα εμφανιστεί στην οθόνη μετά την εκτέλεση των παρακάτω τμημάτων προγραμμάτων (ξεκινήστε από την αριστερή στήλη). Επαληθεύστε τις απαντήσεις σας εκτελώντας τα προγράμματα μέσα από το περιβάλλον της γλώσσας Python.

**Α.**

```
>>> x = 35
>>> y = 10
>>> x = x / y
>>> print x
…… Τι παρατηρείτε;
```

**Β.**

```
>>> x = 45
>>> y = 10
>>> divmod(x,y)
……
>>> x / y
……
>>> x % y
……
>>> divmod(y,x)
……..
```

**Γ.**

```
>>> a = 0xB
>>> print a
……..
```

**Δ.**

```
>>> x,y,z = 1, 4, "today"
>>> print z, x
……
```

**Ε.**

```
>>> x = 234
>>> y = 456.7
>>> x,y = y,x
>>> print x
……..
>>> print y
……Τι παρατηρείτε;
```

**ΣΤ.**

```
>>> x = 2
>>> x = 2 **3 + 2/3
>>> print x
……
>>>x = 2 **3 + 2/ float(3)
>>>print x
……..
```

**Ζ.**

```
>>>x = 2
>>>x -= 1
>>>x = x -1
>>>print x
……
```

**Βοήθεια - απαντήσεις:**

**Α.** 3 **Β.** (4, 5), 4, 5, (0, 10), **Γ.** 11, **Δ.** today 1
**Ε.** 456.7, 234, **ΣΤ.** 8, 8.66666666667, **Ζ.** 0

**Δραστηριότητα 6.** Πράξεις

Να συμπληρώσετε κατάλληλα τον παρακάτω πίνακα υπολογίζοντας το αποτέλεσμα, με την εφαρμογή της προτεραιότητας των πράξεων.
Στην αριστερή στήλη του πίνακα παρουσιάζεται μια πράξη που πρέπει να εκτελεστεί στον υπολογιστή, χρησιμοποιώντας το προγραμματιστικό περιβάλλον της γλώσσας Python 2.7.
Αρχικά να συμπληρώσετε τη μεσαία στήλη με τα αποτελέσματα που πιστεύετε ότι θα εμφανιστούν στην οθόνη του υπολογιστή μετά την εκτέλεση της πράξης. Στη συνέχεια, να επαληθεύσετε τα στοιχεία που συμπληρώσατε, πληκτρολογώντας και εκτελώντας κάθε πράξη ξεχωριστά, μέσα στο περιβάλλον της γλώσσας Python.

| **Πράξεις** | **Αναμενόμενο χειρόγραφο αποτέλεσμα** | **Αποτέλεσμα στην οθόνη** |
|---|---|---|
| 15 + 2/2 | | |
| 5 * (3 + 2) / 10 | | |
| 15 * 2 / 3 | | |
| 15 * 2 / 4 | | |
| 15*2.0/4 | | |
| 2**3*3**2 | | |
| 8/4%2 | | |
| 11%3-2*2 | | |
| 2*(5%3)+4/(1+3) | | |

**Βοήθεια:** 16, 2, 10, 7, 7.5, 72, 0, -2, 5

**Δραστηριότητα 7.** Βρες το αποτέλεσμα (Διερεύνηση εκφράσεων)
Δίνονται οι παρακάτω εκφράσεις σε Python
**Α)** (x+y)/(x**3+y**2+1)*z
Αν x=2, y=3 και z=1, ποιο αποτέλεσμα θα εμφανιστεί στην οθόνη του υπολογιστή; Επαληθεύστε την απάντησή σας, πληκτρολογώντας διαδοχικά, στο προγραμματιστικό περιβάλλον της Python, τις εκφράσεις και αντικαθιστώντας τις μεταβλητές με τις αντίστοιχες τιμές τους. Δικαιολογήστε το αποτέλεσμα που εμφανίστηκε στην οθόνη.
**Β)** a+b*(a**c+c/2)**2
Αν a=1 και b=2 και c=4, ποιο αποτέλεσμα από τα παρακάτω θα εμφανιστεί στην οθόνη του υπολογιστή:

α) 27 β) 19 γ) 64

**Γ)** (x*y+x+2)**2+3**2
Αν x=2, y=3, ποιο αποτέλεσμα από τα παρακάτω θα εμφανιστεί στην οθόνη του υπολογιστή:

α) 109 β) 81 γ) 36

**Βοήθεια-απαντήσεις:**

**Α)** 5/18 = 0 (ακέραια διαίρεση) Β) 19, Γ) 109

**Δραστηριότητα 8.** Τύποι δεδομένων

Να δώσετε διαδοχικά τις παρακάτω εντολές στο διερμηνευτή της Python και να εξηγήσετε τα αποτελέσματα.

```
>>> a = 2
>>> type(a)
<type 'int'> #ακέραιος
>>> b = 10
>>> p = a*b
>>> print a + b
12                          # 10 +2
>>> print " a + b "
a + b # τα εισαγωγικά δηλώνουν συμβολοσειρά
```

```
>>> type( "python")
<type 'str'> # συμβολοσειρά
>>> type( 3.14 )
<type 'float'>
>>> type( a == 2 )
<type 'bool'> #Λογική σύγκριση
```

**Δραστηριότητα 9.** Πράξεις - Τύποι δεδομένων

Να δώσετε τις παρακάτω εντολές στο διερμηνευτή της Python 2.7 και να εξηγήσετε τη λειτουργία του τελεστή της διαίρεσης.

| | |
|---|---|
| >>> 3 / 2<br>1 # ακέραια διαίρεση<br>>>> 3.0 / 2<br>1.5 # διαίρεση<br>>>> 1 / 2<br>0<br>>>> 1.0 / 2<br>0.5 | >>> type( 1 / 2 )<br><type 'int'><br>>>> type( 1.0 / 2)<br><type 'float'><br>>>> int( 1.0 / 2.0)<br>0<br>>>> float( 1 ) / 2<br>0.5 |

**Δραστηριότητα 10.** Λογικοί τελεστές (Θέμα για συζήτηση) Συζητήστε στην τάξη για τη χρήση των λογικών πράξεων **not**, **or** και **and**, καθώς και για τους τελεστές σύγκρισης στην καθημερινή ζωή και στον προγραμματισμό υπολογιστικών συσκευών. Δώστε παραδείγματα συνδέοντας λογικές εκφράσεις για τον έλεγχο καθημερινών λειτουργιών, όπως για παράδειγμα: ΑΝ (φανάρι πράσινο για τον πεζό) ΚΑΙ (δεν έρχεται κινούμενο όχημα) ΤΟΤΕ διασχίζω τη διάβαση.

Να γίνει συζήτηση

**Δραστηριότητα 11.** Λογικοί τελεστές και πίνακες αλήθειας

Συμπληρώστε τις παρακάτω σχέσεις με το Ψευδής/False/0 ή Αληθής/True/1.

| | |
|---|---|
| 1 != 0 | True |
| 1 != 1 | False |
| 0 != 1 | True |
| 0 != 0 | False |

| | |
|---|---|
| 1 == 0 | False |
| 1 == 1 | True |
| 0 == 1 | False |
| 0 == 0 | True |

**Βοήθεια**

| | |
|---|---|
| != όχι ίσο / διάφορο | == ίσο / ταυτίζεται |

**Δραστηριότητα 12.** Λογικές πράξεις

Να συμπληρώσετε τον παρακάτω πίνακα αληθείας

| P | Q | P and Q | P or Q | not P | not P and not Q |
|---|---|---|---|---|---|
| True | True | True | True | False | False |
| True | False | False | True | False | False (Η λύση δίνεται για βοήθεια) |
| False | True | False | True | True | False |
| False | False | False | False | True | True |

**Δραστηριότητα 13.** Λογικές εκφράσεις

Ποια η τιμή αληθείας για τις παρακάτω προτάσεις (Ψευδής/False/0 - Αληθής/True/1); Συμπληρώστε αρχικά τον πίνακα χειρόγραφα. Στη συνέχεια, δοκιμάστε τις προτάσεις στο περιβάλλον της Python και συγκρίνετε τις απαντήσεις σας με τα αποτελέσματα που θα εμφανιστούν.

| Εκφράσεις | Χειρόγραφος υπολογισμός | Αποτέλεσμα περιβάλλοντος Python |
|---|---|---|
| 1 == 1 and 0 != 1 | | True |
| "test" == 'test' | | True |
| 1 == 1 or 2 != 1 or 5 == 5 | | True |
| False and 1!= 0 | | Η λύση εδώ είναι False |
| not (4 == 4 and 1 != 0) | | False |
| not (5==5 or (1!=0 and 6 != 7)) | | Η λύση εδώ είναι False |

**Δραστηριότητα 14.** Τελεστές (Ερωτήσεις αντιστοίχισης)

Κάντε τις κατάλληλες συνδέσεις. Γράψτε με τη σειρά κάτω από τον πίνακα τους αριθμούς της στήλης Α και δίπλα τους το αντίστοιχο γράμμα της στήλης Β, ώστε να σχηματίζεται η σωστή απάντηση.

| Στήλη Α | Στήλη Β |
|---|---|
| 1. * | α. Σχεσιακός τελεστής |
| 2. False | β. Λογικός τελεστής |
| 3. > | γ. Αριθμητικός τελεστής |
| 4. and | δ. Αλφαριθμητική τιμή |
| 5. length | ε. Λογική τιμή |
| 6. "πλάτος" | στ. Όνομα Μεταβλητής |

**Απάντηση:** 1 -- γ, 2--ε, 3--α, 4--β, 5--στ, 6--δ

**Δραστηριότητα 15.** Η εντολή print (statement)

Να δώσετε τις παρακάτω εντολές στο διερμηνευτή της Python και να εξηγήσετε τα αποτελέσματα. Τι συμπεράσματα βγάζετε για τη λειτουργία της print;

```
>>> print 1,000, 000
1 0 0
>>> print (1, 2, 3)
(1, 2, 3)
>>> print 1, ; print 2, ; print 3;
1 2 3
>>> print 1, 2, 3
1 2 3
>>> print 1;2;3
1
2
3
```

**Δραστηριότητα 16.** Περισσότερα για την εντολή print

Μπορείτε να προβλέψετε τα αποτελέσματα των παρακάτω εντολών πριν τις εκτελέσετε στο διερμηνευτή της Python; Επαληθεύστε τις απαντήσεις σας μέσα από το περιβάλλον της
Python.

| | |
|---|---|
| >>> print “Python ”, 2<br>Python 2<br>>>> print “Python ” + 2<br>Μήνυμα λάθους<br>>>> print “Python ” + str(2)<br>Pyhton2 | >>> print “Monty” + “Python”<br>MontyPython<br>>>> print 3 * “Python”<br>PythonPythonPython<br>>>> print 3 * “Python” * 2<br>PythonPythonPythonPythonPython<br>Python |

**Δραστηριότητα 17.** Μεταβλητές

Να γράψετε σε Python τις αντίστοιχες εντολές που επιτελούν τις παρακάτω λειτουργίες:

- Να οριστεί η μεταβλητή με όνομα name, ώστε να έχει ως τιμή το όνομά σας.
- Μηδενισμός της μεταβλητής number.
- Αύξηση κατά 1 της μεταβλητής number.
- Αύξηση της μεταβλητής number κατά 50%.
- Διπλασιασμός της μεταβλητής number.
- Να οριστεί η μεταβλητή value=456.7 και στη συνέχεια να οριστεί η μεταβλητή value_square, που να περιέχει το τετράγωνο της value.
- Να οριστεί η μεταβλητή με όνομα logic που να περιέχει την τιμή False.

**Απάντηση**

name="Nikos", number=0, number=number+1,
number=number*1.5, number=number*2, value = 456.7,
value_square=value**2, logic=False.

**Δραστηριότητα 18.** Αντιμετάθεση τιμών μεταξύ δύο μεταβλητών

Να εξηγήσετε τη λειτουργία των παρακάτω τμημάτων κώδικα σε γλώσσα Python, αρχικά του τμήματος Α' (αριστερή στήλη) και στη συνέχεια του τμήματος Β' (δεξιά στήλη). Τα δύο τμήματα κώδικα επιτελούν την ίδια λειτουργία;

| A | B |
|---|---|
| a = input( “a = ” )<br>b = input( “b = ” )<br>temp = a<br>a = b<br>b = temp<br>print a, b | a = input( “a = ” )<br>b = input( “b = ” )<br>a,b = b,a<br>print a, b |

**Απάντηση:** Το τμήμα Α χρησιμοποιεί μια βοηθητική μεταβλητή temp για τη αντιμετάθεση των τιμών. Το τμήμα Β χρησιμοποιεί μια εσωτερική διαδικασία της Python που επιτρέπει αντιμετάθεση με την εντολή a,b=b,a.

**Δραστηριότητα 19**

Να γράψετε, σε γλώσσα Python, πρόγραμμα που να διαβάζει το μήκος της ακτίνας ενός κύκλου και να τυπώνει τη διάμετρο, το μήκος και το εμβαδόν αυτού του κύκλου. Βοήθεια: Η διάμετρος του κύκλου δίνεται από τον τύπο $d=2\cdot r$, η περίμετρος από τον τύπο $p=2\cdot\pi\cdot r$ (όπου $\pi\approx3,14$) και το εμβαδόν από τον τύπο $E=\pi\cdot r^2$.

**Απάντηση**

```
import math # βιβλιοθήκη με μαθηματικές συναρτήσεις
r = float(input ("Δώστε την ακτίνα του κύκλου : "))
d = 2*r
print "Η διάμετρος του κύκλου με ακτίνα ",r, " είναι: ",d
p=2*math.pi*r
print "Η περίμετρος του κύκλου με ακτίνα " ,r, " είναι: ", p
E=math.pi* r**2
print "Το εμβαδόν του κύκλου με ακτίνα ",r, " είναι: ",E
```

**Δραστηριότητα 20.** Προσθέστε τα κατάλληλα σχόλια

Να συμπληρώσετε ως σχόλια (τεκμηρίωση) την επεξήγηση της ανάθεσης για κάθε πρόταση:

```
a = b = c = 0        # …Όλες οι μεταβλητές παίρνουν την τιμή 0.
a, b = 1, 2.5        # …a =1 και b= 2.5
a, b, c = (1,2,3)    # …a=1, b=2, c=3
a, b = b, a          # …αντιμετάθεση τιμών
```

Δοκιμάστε τις απαντήσεις σας στο περιβάλλον IDLE της Python χρησιμοποιώντας και την εντολή print, ώστε να εμφανιστούν τα αποτελέσματα για κάθε ανάθεση τιμών.

# ΚΕΦΑΛΑΙΟ 4 Αλγοριθμικές δομές

*Το κεφάλαιο αυτό είναι **εντός** διδακτέας-εξεταστέας ύλης. Στο κεφάλαιο αυτό δεν υπάρχουν ασκήσεις - Δραστηριότητες στο Βιβλίο Μαθητή.*

## Λύσεις Δραστηριοτήτων Τετραδίου Εργασιών Μαθητή

**Δραστηριότητα 1.** Αλγοριθμικές δομές - Δομή ακολουθίας - Ροές εκτέλεσης προγράμματος.

Μελετήστε το παρακάτω τμήμα προγράμματος και προσπαθήστε να βρείτε το αποτέλεσμα που θα εμφανιστεί στην οθόνη μετά την εκτέλεσή του. Εκτελέστε το πρόγραμμα στο διερμηνευτή της Python και απαντήστε στα ερωτήματα:

```
# Διερεύνηση x += 1 / x = x + 1
x = 0
print 'x= ', x
x += 1
x = x + 1
x = x - 1
x -= 1
print 'x= ', x
```

- Τι θα εμφανιστεί από την εκτέλεση του προγράμματος;
  **x= 0, x= 0**

- Τι αποτέλεσμα έχουν οι εντολές x+=1 και x-=1; ***Αύξηση της τιμής της μεταβλητής x κατά 1 και μείωση της τιμής της μεταβλητής x κατά 1 αντίστοιχα.***
- Με ποιες εντολές είναι ισοδύναμες οι δύο παραπάνω εντολές;
  ***Με τις x = x + 1 και x = x -1 αντίστοιχα***

**Δραστηριότητα 2.** Δομή ακολουθίας (Υπολογισμός μέσου όρου τριών αριθμών)

Να γραφεί πρόγραμμα σε Python που να διαβάζει τις τιμές τριών αριθμών, να υπολογίζει το μέσο όρο τους και να τον εμφανίζει στην οθόνη.

**Απάντηση**

**Πρόγραμμα σε Python** (έκδοση 2.7.x)

```
# Πρόγραμμα ΜΕΣΟΣ ΟΡΟΣ ΤΡΙΩΝ ΑΡΙΘΜΩΝ
A = input('Δώσε τον πρώτο αριθμό Α: ' )
B = input('Δώσε το δεύτερο αριθμό Β: ' )
C = input('Δώσε τον τρίτο αριθμό Γ: ')
MESOS_OROS = (A+B+C)/3.0
print 'Ο μέσος όρος των αριθμών: ',A,B,C', είναι:',
MESOS_OROS
```

**Διερεύνηση**

Τι διαφορά μπορεί να υπάρξει στο αποτέλεσμα, αν γράψουμε την έκφραση: MESOS_OROS = (A+B+C) / 3 αντί της MESOS_OROS = (A+B+C)/3.0 (με τα A,B,C να έχουν ακέραιες τιμές); (Αφορά την έκδοση Python 2.7.x)

**Απάντηση:** Θα γίνει ακέραια διαίρεση

**Δραστηριότητα 3.** Γεννήτρια αριθμών

Να εκτελέσετε τρεις (3) τουλάχιστον φορές στο διερμηνευτή της Python τις παρακάτω εντολές και στη συνέχεια απαντήστε στα ερωτήματα:

- Τι τύπου είναι οι τιμές που εμφανίζονται (ακέραιες, κινητής υποδιαστολής);
  *Ακέραιες*
- Ποιο το διάστημα των τιμών της μεταβλητής number;
  *Από 1 μέχρι και 5*

```
import random # αρχικά δηλώνουμε ότι θα χρησιμοποιήσουμε τη random
number = random.randint(1, 6) # επιστρέφει έναν τυχαίο αριθμό στο [ 1, 6)
print number
```

**Βοήθεια**

Η random.randint( start, end ) επιστρέφει, με τυχαίο τρόπο, έναν ακέραιο αριθμό μέσα από το αριθμητικό διάστημα [start, end).

**Δραστηριότητα 4.** Δομή επιλογής if–else (Λογικές εκφράσεις-συνθήκες)

Συμπληρώστε στη 2η στήλη του πίνακα, ποια θα είναι τα αποτελέσματα μετά την εκτέλεση του αντίστοιχου κώδικα Python της 1ης στήλης. Να λάβετε υπόψη ότι το αποτέλεσμα εξαρτάται από την τιμή αλήθειας των λογικών εκφράσεων της συνθήκης ελέγχου μέσα στη δομή επιλογής if.

Στη συνέχεια, εκτελέστε κάθε τμήμα του κώδικα στο περιβάλλον της Python συγκρίνοντας τα αποτελέσματα που εμφανίζονται στην οθόνη με τα δικά σας.

| if …else | Αναμενόμενο Αποτέλεσμα (Χειρόγραφα) | Αποτέλεσμα περιβάλλοντος Python |
|---|---|---|
| 1 == 1 and 0 != 1 | | True |
| # " " και ' '<br>if "test" == 'test':<br>  print "True"<br>  else:<br>  print "False-0" | | True |
| # Ελληνικά και Λατινικά<br>if "KALHMERA" == "ΚΑΛΗΜΕΡΑ":<br>  print "True"<br>else:<br>  print "False-0" | | False-0 |
| if (1 == 1 or (2 != 1 or 5 == 5)):<br>  print "True"<br>else:<br>  print "False-0" | | True |

| | | |
|---|---|---|
| if (2!=2 and 1!= 0):<br>  print "True"<br>else:<br>  print "False-0" | | False-0 |
| # true και True<br>if (true):<br>  print "True"<br>else:<br>  print "False-0" | | Error (λόγω του true) |
| # true και True<br>if True:<br>  print "True"<br>else:<br>  print "False-0" | | True |

**Δραστηριότητα 5.** Δομή επιλογής if … else …(Άρτιος ή περιττός)

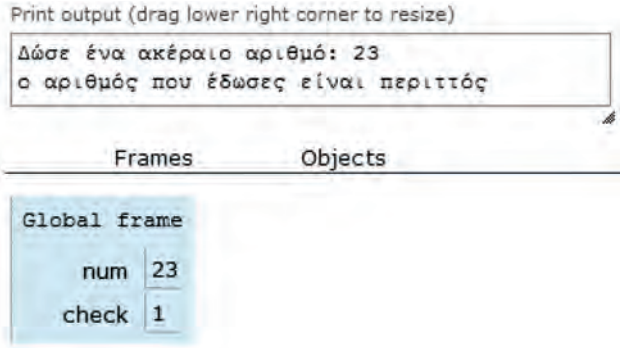
Να γραφεί πρόγραμμα σε Python που να διαβάζει έναν ακέραιο αριθμό και να ελέγχει αν είναι άρτιος ή περιττός. Στη συνέχεια, να εμφανίζει στην οθόνη αντίστοιχο μήνυμα.

Οδηγία: α) Δημιουργήστε το πρόγραμμα πρώτα σε χαρτί και στη συνέχεια, καταχωρήστε και εκτελέστε το στο περιβάλλον της Python, β) μελετήστε τις τυχόν διαφορές με την αρχική σας δημιουργία.

**Απάντηση**

Ένας ακέραιος αριθμός x είναι άρτιος, αν είναι πολλαπλάσιος του 2. Αν, δηλαδή, το υπόλοιπο της διαίρεσης του x δια του 2 είναι 0. Διαφορετικά είναι περιττός.

```
#πρόγραμμα άρτιος ή περιττός
num=int(input('Δώσε ένα ακέραιο αριθμό: '))
check = num % 2
if check == 0:
    print "Ο αριθμός είναι άρτιος"
else:
    print "Ο αριθμός που έδωσες είναι περιττός"
```




***Εικ. 4.1:** Αποτέλεσμα εκτέλεσης κώδικα στο PythonTutor.*

**Δραστηριότητα 6.** Δομή επιλογής if … elif … else … (Συμπλήρωσε κατάλληλα τις στήλες του πίνακα)

Μελετήστε τον κώδικα Python (με εντολές if) στην αριστερή στήλη του πίνακα που ακολουθεί. Στη συνέχεια, στη δεξιά στήλη του ίδιου πίνακα, γράψτε αντίστοιχο πρόγραμμα σε Python, χρησιμοποιώντας τη δομή επιλογής if elif else. Το πρόγραμμα θα πρέπει να επιστρέφει τα ίδια αποτελέσματα στην οθόνη με αυτά του προγράμματος της αριστερής στήλης, για όλους τους δυνατούς συνδυασμούς τιμών των μεταβλητών ar1 και ar2. Στη συνέχεια, δοκιμάστε, με διάφορες χαρακτηριστικές τιμές των ar1 και ar2, τα δύο προγράμματα στο περιβάλλον της Python.

| **Κώδικας με χρήση της απλής if** | **Κώδικας με χρήση της if - elif - else** |
|---|---|
| ar1 = input('Δώσε έναν αριθμό A : ')<br>ar2 = input('Δώσε δεύτερο αριθμό B: ')<br>if ar1 < ar2:<br>  print "A < B"<br>if ar1 > ar2:<br>  print "A > B"<br>if ar1 == ar2:<br>  print " A = B " | ar1 = input('Δώσε έναν αριθμό A : ')<br>ar2 = input('Δώσε δεύτερο αριθμό B: ')<br>if ar1 < ar2:<br>  print "A < B"<br>elif ar1 > ar2:<br>  print "A > B"<br>else:<br>  print " A = B " |

**Δραστηριότητα 7.** Εμφωλευμένη επιλογή (Κλιμακωτή χρέωση)

Μια εταιρεία κινητής τηλεφωνίας ακολουθεί, ανά μήνα, την πολιτική τιμών που φαίνεται στον παρακάτω πίνακα:

| Πάγιο=20€ | |
|---|---|
| Χρόνος τηλεφωνημάτων (δευτερόλεπτα) | Χρονοχρέωση (€/ δευτερόλεπτο) |
| 1-500<br>501-800<br>801 και άνω | 0.02<br>0.009<br>0.007 |

Να γραφεί πρόγραμμα σε Python που:

- Να διαβάζει τη χρονική διάρκεια των τηλεφωνημάτων ενός συνδρομητή σε διάστημα ενός μήνα.
- Να υπολογίζει τη μηνιαία χρέωση του συνδρομητή.
- Να εμφανίζει σχετικό μήνυμα «η συνολική χρέωση του μήνα είναι: » και τη μηνιαία χρέωση του συνδρομητή.

**Σημείωση: Η χρονοχρέωση θεωρείται κλιμακωτή.** Δηλαδή τα πρώτα 500 δευτερόλεπτα χρεώνονται με 0.02 €/δευτερόλεπτο, τα επόμενα 300 δευτερόλεπτα (από 501 -& 800) με 0.009 (€/δευτερόλεπτο) και τα πέραν των 800 με 0.007 (€/δευτερόλεπτο).

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή

**Δραστηριότητα 8.** Δομή Επανάληψης με for (συμπλήρωσης κώδικα). Υπολογισμός γινόμενου 10 διαδοχικών ακεραίων αριθμών.
Να συμπληρώσετε κατάλληλα τα κενά, ώστε το παρακάτω πρόγραμμα σε Python να υπολογίζει το γινόμενο των ακεραίων αριθμών (1,2,3,4,5,6,7,8,9,10). Δηλαδή το γινόμενο Multi=1*2*3*4*5*6*7*8*9*10, ή ισοδύναμα το γινόμενο Multi=(((((((1*2)*3)*4)*5)*6)*7)*8)*9)*10.

**Απάντηση**

```
# Πρόγραμμα γινόμενο ακεραίων αριθμών
Multi = 1
for i in range(2 , 11):
    Multi = Multi* i
print ' Το αποτέλεσμα είναι ', Multi
```

**Δραστηριότητα 9.** Δομή Επανάληψης με for (Βρείτε τι θα εμφανίσει το πρόγραμμα). Δίνεται το παρακάτω πρόγραμμα σε Python.

```
s = 0
for i in range(0, 10, 2):
    s = s + 1
    print i, s
print '--------'
s = 0
for i in range(10, 0, -2):
    s += 1
    print i, s
print s
```

1) Μελετήστε τον κώδικα ανά γραμμή εντολών και σημειώστε χειρόγραφα τα αποτελέσματα που περιμένετε να εμφανιστούν στην οθόνη του υπολογιστή.
2) Καταχωρήστε τον κώδικα στο περιβάλλον της Python και δοκιμάστε τον, συγκρίνοντας τα αποτελέσματα της οθόνης με αυτά του σημειώσατε.

**Απάντηση:**

0 1
2 2
4 3
6 4
8 5
--------
10 1
8 2
6 3
4 4
2 5
5

**Δραστηριότητα 10.** Η συνάρτηση range και η δομή επανάληψης for.

Να εκτελέσετε τις παρακάτω συναρτήσεις (μία, μία ανά στήλη) στο διερμηνευτή και να σχολιάσετε τα αποτελέσματα. Ποια είναι η λειτουργία της ενσωματωμένης στην Python συνάρτησης range();

```
>>> range( 5 )
>>> range( 1, 5 )
>>> range (1,5,1)
>>> range( 0, 5 )
>>> range( 1, 10, 2 )
>>> range( 10, 1, –2 )
```

```
>>> range(–10, –20, –10)
>>> range( 0 )
>>> range( 1 )
>>> type( range( 1 ) )
>>> range(4) == [0,1,2,3]
```

**Απάντηση:**

[0, 1, 2, 3, 4]
[1, 2, 3, 4]
[1, 2, 3, 4]
[0, 1, 2, 3, 4]
[1, 3, 5, 7, 9]
[10, 8, 6, 4, 2]

Error
[ ]
[0]
<type 'list'>
True

Να εκτελέσετε τα παρακάτω προγράμματα Α,Β,Γ στην Python και να σχολιάσετε τα αποτελέσματα

**Α**

```
sum = 0

for i in [1,2,3,4,5,6]:
    print i
    sum = sum + i

print "sum = ", sum
```

**Β**

```
sum = 0

for i in range(1, 7) :
    print i
    sum = sum + i

print "sum = ", sum
```

**Γ**

```
sum = 0
i = 1
while i < 7 :
    print i
    sum = sum + i
    i = i + 1
print "sum = ", sum
```

**Απάντηση:**

| | | |
|---|---|---|
| 1<br>2<br>3<br>4<br>5<br>6<br>sum=21 | 1<br>2<br>3<br>4<br>5<br>6<br>sum=21 | 1<br>2<br>3<br>4<br>5<br>6<br>sum=21 |

**Δραστηριότητα 11.** (Η δομή επανάληψης for με εμφωλευμένες επαναλήψεις). Βρες τι κάνει το πρόγραμμα.

Μελετήστε και σημειώστε στο χαρτί, τι κάνει το παρακάτω πρόγραμμα, εάν δεχθούμε ότι με h, m, s συμβολίζουμε τις ώρες, λεπτά και δευτερόλεπτα αντίστοιχα. Στη συνέχεια καταχωρήστε το στο περιβάλλον της Python, εκτελέστε το και συγκρίνετε τα αποτελέσματα με αυτά που σημειώσατε. Παρατηρήστε ότι μέσα σε μία επανάληψη for, μπορεί να περικλείεται και άλλη (εμφωλευμένη) επανάληψη for.

```
for h in range (0,2):
    for m in range (0,3):
        for s in range (0,4):
            print h,m,s
```

**Απάντηση:** Εμφανίζει

0 0 0
0 0 1
0 0 2
0 0 3
0 1 0
0 1 1
0 1 2
0 1 3
0 2 0
0 2 1

0 2 2
0 2 3
1 0 0
1 0 1
1 0 2
1 0 3
1 1 0
1 1 1
1 1 2
1 1 3
1 2 0
1 2 1
1 2 2
1 2 3

**Δραστηριότητα 12.** Δομή Επανάληψης for (Πολλαπλάσια του 9) Να γραφεί πρόγραμμα που να ελέγχει όλους τους τριψήφιους ακεραίους αριθμούς και να εμφανίζει όσους είναι πολλαπλάσια του 9.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή

**Δραστηριότητα 13.** Δομή Επανάληψης με for.
Μελετήστε το παρακάτω πρόγραμμα Python

```
arxh, telos, bhma = input('Δώσε τρεις τιμές, αρχή, τέλος, βήμα: ')
for i in range(arxh, telos, bhma):
    print i
```

Κατά την εκτέλεση του προγράμματος, ποιες τιμές πρέπει να εισάγουμε από το πληκτρολόγιο στις τρεις μεταβλητές, ώστε η εκτέλεση της εντολής επανάληψης να εμφανίσει διαδοχικά:

1) Όλους τους ακέραιους από το 1 μέχρι και το 100.
2) Τους άρτιους αριθμούς από το 0 έως το 100.
3) Τους περιττούς αριθμούς από το 0 έως το 100.

**Απάντηση**

1, 101, 1
2, 101, 2
1, 100, 2

**Δραστηριότητα 14** Δομές: for και if (Υπολογισμός μέσου όρου, μέγιστου και ελάχιστου).

Ένα τμήμα της Γ' τάξης έχει 25 μαθητές. Να αναπτυχθεί πρόγραμμα σε Python που:
1) Να διαβάζει τους βαθμούς όλων των μαθητών στο μάθημα "Προγραμματισμός Υπολογιστών" και να υπολογίζει το μέσο όρο της βαθμολογίας του τμήματος για το μάθημα αυτό και να τον εμφανίζει στην οθόνη.
2) Να βρίσκει την υψηλότερη και τη χαμηλότερη βαθμολογία και να τις εμφανίζει στην οθόνη με κατάλληλο μήνυμα.

**Οδηγία:** Για την επίλυση να μη χρησιμοποιηθούν οι *ενσωματωμένες* μαθηματικές *συναρτήσεις* max() και min() της Python.
Δημιουργήστε τον κώδικα σε χαρτί και στη συνέχεια καταχωρήστε τον στο περιβάλλον της Python.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή

**Δραστηριότητα 15.** Δομή επανάληψης με while (γινόμενο 200 ακεραίων αριθμών)

Κάντε τις απαραίτητες αλλαγές στο παραπάνω πρόγραμμα του λυμένου παραδείγματος, ώστε:
1) Να υπολογίζει το γινόμενο των αριθμών από το 1 ως το 200.
2) Να υπολογίζει το γινόμενο 10 ακεραίων αριθμών που θα τους διαβάζει από το πληκτρολόγιο.

**Απάντηση 1)**

```
p = 1                  # αρχική τιμή στο γινόμενο
i = 1                  #αρχική τιμή στη μεταβλητή ελέγχου
while i <= 100 :       # έλεγχος της επανάληψης
    p= p* i
    i= i + 1           # αύξηση του μετρητή
print p
```

**Απάντηση 2)**

```
i = 1                  #αρχική τιμή στη μεταβλητή ελέγχου
prod = 1               # αρχική τιμή γινομένου
while i <= 10 :        # έλεγχος της επανάληψης
    p=input("Dose akeraio arithmo")
    prod= prod* p
    i= i + 1           # αύξηση του μετρητή
print prod
```

**Δραστηριότητα 16.** Δομή επανάληψης με while (διερεύνηση προγράμματος).

Δίνεται το παρακάτω τμήμα προγράμματος σε Python με αριθμημένες τις εντολές ανά γραμμή:

| | |
|---|---|
| 1. | x = 20 |
| 2. | s = 0 |
| 3. | while x < 100: |
| 4. | x = x + 10 |
| 5. | s = s + x |
| 6. | print x, s |

Να απαντήσετε στα παρακάτω ερωτήματα:
1) Πόσες φορές θα εκτελεστεί η εντολή στη γραμμή 4;
2) Ποιες είναι όλες οι τιμές που θα πάρει η μεταβλητή x κατά την εκτέλεση του προγράμματος (μαζί με την αρχική);
3) Τι θα εμφανιστεί στην οθόνη στο τέλος του προγράμματος;

**Απάντηση:**

1) 8 φορές
2) 20, 30, 40, 50, 60, 70, 80, 90, 100
3) 100, 520

**Δραστηριότητα 17.** Δομές επανάληψης while / for. Σωστό-Λάθος

Χαρακτηρίστε ως Σωστές ή Λανθασμένες τις παρακάτω προτάσεις, σημειώνοντας την ένδειξη Σ, αν η πρόταση είναι σωστή, ή Λ, αν αυτή είναι λανθασμένη.

| **Προτάσεις** | **Σ/Λ** |
|---|---|
| 1) Η δομή while (Όσο) τερματίζει, όταν η συνθήκη γίνει αληθής. | Λ |
| 2) Μια δομή επανάληψης for μπορεί να εκτελείται απεριόριστα. | Λ |
| 3) Η δομή for χρησιμοποιείται, όταν ο αριθμός των επαναλήψεων δεν είναι προκαθορισμένος. | Λ |
| 4) Η δομή while χρησιμοποιείται, όταν ο αριθμός επαναλήψεων είναι προκαθορισμένος. | Λ |
| 5) Οι εντολές που περιλαμβάνονται μέσα στη δομή while θα εκτελεστούν τουλάχιστον μία φορά. | Λ |

**Δραστηριότητα 18.** Δομές επανάληψης while/for. Από τη δομή επανάληψης for στη δομή επανάληψης while και αντίστροφα.

**Α.** Γράψτε, στη δεξιά στήλη του πίνακα και στα κενά κελιά, τμήμα προγράμματος σε Python, ισοδύναμο με αυτό της αριστερής στήλης, χρησιμοποιώντας τη δομή επανάληψης while, όπως στο πρώτο παράδειγμα της 1ης γραμμής.

| | |
|---|---|
| for i in range (1,10):<br>  print i*i | i=1<br>while i<10:<br>  print i*i<br>  i=i+1 |
| for i in range (10,51,2):<br>  print i*i | i=10<br>while i<51:<br>  print i*i<br>  i = i +2 |
| for i in range (100,51,-2):<br>  print i*i | i=100<br>while i>51:<br>  print i*i<br>  i = i -2 |

**Β.** Γράψτε, στην αριστερή στήλη, τμήμα προγράμματος σε Python ισοδύναμο με αυτό της δεξιάς στήλης, χρησιμοποιώντας τη δομή επανάληψης for.

| | |
|---|---|
| for z in range (2,10,4):<br>  print z | z=2<br>while z<10:<br>  print z<br>  z=z+4 |
| for x in range (3,10,2):<br>  print x | x = 1<br>while x < =10:<br>  x = x + 2<br>  print x |

**Δραστηριότητα 19.** Δομές επανάληψης While / For.

Δίνεται το παρακάτω τμήμα προγράμματος σε Python

```
x=60
while x>0:
    for i in range (2,7,2):
        x=x-10
    print x
```

1) Πόσες φορές θα εκτελεστεί η εντολή x=x-10;
2) Τι θα εμφανιστεί διαδοχικά στην οθόνη μετά την εκτέλεση του προγράμματος;

**Απάντηση:**

1) 6 φορές
2) 30 και 0

**Δραστηριότητα 20.**

Ο παρακάτω κώδικας σε Python σχεδιάζει μισό χριστουγεννιάτικο δέντρο:

```
lines = 1
maxlines = 12
while lines <= maxlines:
        print(lines*'*')
        lines +=1
```

```
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
```

Αλλάξτε κατάλληλα τον κώδικα, ώστε να σχεδιάζει και το άλλο μισό.

**Απάντηση**

Γίνεται προσθήκη καταλλήλων κενών μπροστά από κάθε γραμμή και για να υπάρχει συμμετρία, ο αριθμός των " * " σε κάθε γραμμή είναι περιττός

```
lines = 1
maxlines = 24
while lines <= maxlines:
    k=(maxlines-lines)/2
    print (k*" ")+(lines*'*')
    lines +=2
```

**Δραστηριότητα 21.** Δημιουργία και κλήση συναρτήσεων (Διερεύνηση κώδικα - Τι θα εμφανιστεί στην οθόνη)

Μελετήστε και σημειώστε χειρόγραφα το αποτέλεσμα κάθε προγράμματος. Στη συνέχεια καταχωρήστε τον κάθε κώδικα στο περιβάλλον της Python και συγκρίνετε το αποτέλεσμα με αυτό που σημειώσατε.

```
# Πρόγραμμα 1
print '\n'*1000 #clear screen
def print_1(t2):
    t1 = t2+10
    print t1
t1 = 5
print_1(t1)
```

```
# Πρόγραμμα 2
def print_1(t2):
    print t2
    t2 = t2+10
    print t2
t1 = 5
print_1(t1)
print t1
```

**Απάντηση**

Το πρόγραμμα 1 τυπώνει 1000 κενές γραμμές και μετά τον αριθμό 15. Το πρόγραμμα 2 τυπώνει σε διαφορετικές γραμμές τους αριθμούς 5, 15 και 5

**Δραστηριότητα 22.** Δημιουργία συναρτήσεων (το πρώτο μου υποπρόγραμμα)

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασιών Μαθητή.

**Δραστηριότητα 23.** Δημιουργία και κλήση συναρτήσεων (Μέγιστο τριών αριθμών)

Να γραφεί συνάρτηση σε Python που να υπολογίζει το μέγιστο μεταξύ τριών (3) αριθμών.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασιών Μαθητή.

**Δραστηριότητα 24.** Δημιουργία, κλήση και χαρακτηριστικά συναρτήσεων (ερωτήσεις Σωστού-Λάθους)

Συμπληρώστε τη δεύτερη στήλη του πίνακα με Σωστό (Σ) ή Λάθος (Λ).

| Περιγραφή | Σ/Λ |
|---|---|
| Ορίζουμε επιτυχώς μια συνάρτηση:<br>**1) def ektyposi (x1, x2)**<br>**2) def ektyposi (x1 + x2)**<br>**3) def ektyposi (x1; x2)** | 1 Σ<br>2 Λ<br>3 Λ |
| Ορίζουμε επιτυχώς μια συνάρτηση:<br>**1) def ektyposi (x1; x2):**<br>**2) def ektyposi x1, x2**<br>**3) def ektyposi (x1, x1):** | 1 Λ<br>2 Λ<br>3 Λ |
| Η κλήση μιας συνάρτησης γίνεται με:<br>**1)** call όνομα_συναρτησης()<br>**2)** όνομα_συναρτησης ()<br>**3)** run όνομα_συναρτησης ()<br>**4)** Όλα τα παραπάνω | 1 Λ<br>2 Σ<br>3 Λ<br>4 Λ |

**Δραστηριότητα 25.**

Να γράψετε πρόγραμμα σε Python που θα:
1) διαβάζει ένα ποσό σε ευρώ.
2) μετατρέπει το ποσό που διάβασε σε λίρες Αγγλίας (δεχόμαστε ότι 1€ = 0,85 Λίρες)
3) εμφανίζει το ποσό σε ευρώ και λίρες Αγγλίας.
Επέκταση: Αφού διαβάσει το ποσό, να διαβάζει και την ισοτιμία του ενός ευρώ έναντι της λίρας Αγγλίας και στη συνέχεια, με βάση την ισοτιμία αυτή, να κάνει τη μετατροπή.

**Απάντηση**

```
euro = float( raw_input("ποσό σε ευρώ = ") )
lires = 0.85 * euro
print "Τα ", euro, " ευρώ είναι ", lires, " λίρες. "
```

Επέκταση

```
euro = float( raw_input("ποσό σε ευρώ = ") )
isotimia = float( raw_input("ισοτιμία ευρώ/λίρας = ") )
lires = isotimia * euro
print "Τα ", euro, " ευρώ είναι ", lires, " λίρες. "
```

**Δραστηριότητα 26.**

Να γράψετε πρόγραμμα σε Python που να διαβάζει την ηλικία ενός προσώπου. Στη συνέχεια, αν είναι κάτω των 18 ετών, να εμφανίζει τη λέξη "ΑΝΗΛΙΚΟΣ", αν είναι 18 και άνω, να εμφανίζει "ΕΝΗΛΙΚΟΣ" και τέλος, αν είναι άνω των 70 ετών, να εμφανίζει τη λέξη "ΗΛΙΚΙΩΜΕΝΟΣ".

**Απάντηση**

```
age = int( raw_input(" ηλικία = ") )
if age < 18 :
    print " Ανήλικος "
elif age <= 70 :
    print " Ενήλικος "
else :
    print " Ηλικιωμένος "
```

**Δραστηριότητα 27.**

Η μισθοδοσία υπαλλήλου προβλέπει επίδομα τέκνων, με βάση τον παρακάτω πίνακα:

| Αριθμός παιδιών | Ποσό επιδόματος μισθοδοσίας |
|---|---|
| 0 έως και 2 | 0 ευρώ |
| 3 (τρίτεκνη) | 100 ευρώ συνολικά |
| άνω των 3 (πολύτεκνη) | Το αρχικό επίδομα των τριών παιδιών (100€), συν 20 € για κάθε ένα παιδί πέραν των τριών. |

Να γράψετε πρόγραμμα σε γλώσσα Python το οποίο, για μία οικογένεια θα:
1) διαβάζει τον αριθμό των παιδιών ενός (μόνο) υπαλλήλου
2) εμφανίζει το μήνυμα τρίτεκνη ή πολύτεκνη οικογένεια στην αντίστοιχη περίπτωση
3) υπολογίζει και θα εμφανίζει το ποσό του επιδόματος που αναλογεί στον υπάλληλο.

**Απάντηση**

```
paidia = int( raw_input(" Αριθμός παιδιών = ") )
if paidia < 3 :
    poso = 0
elif paidia == 3 :
    poso = 100
    print " Τρίτεκνη οικογένεια "
else :
    poso = 100 + ( paidia – 3 ) * 20
    print " Πολύτεκνη οικογένεια "
print " Επίδομα = " , poso
```

**Δραστηριότητα 28.**

Ένας σκληρός δίσκος έχει χωρητικότητα 500 MB για αποθήκευση αρχείων. Ο κάτοχός του τον γεμίζει με αρχεία. Θεωρώντας ότι το αποθηκευτικό μέσο είναι αρχικά άδειο, να γράψετε, σε Python, πρόγραμμα που θα διαβάζει το μέγεθος κάθε αρχείου σε

MB, μέχρι το συνολικό μέγεθος να ξεπεράσει τη χωρητικότητά αυτή. Στη συνέχεια, θα εμφανίζει το συνολικό πλήθος των αρχείων που έχουν αποθηκευθεί στο δίσκο.

**Απάντηση**

```
size = int( raw_input(" Μέγεθος αρχείου σε MB = ") )
files = 0
capacity = 500
while size <= capacity :
    capacity = capacity – size
    files = files + 1
    size = int( raw_input(" Μέγεθος αρχείου σε MB = ") )
print " Αρχεία στο δίσκο = " , files
```

**Δραστηριότητα 29.**

Ένα ασανσέρ έχει μέγιστο όριο ασφάλειας τα 500 κιλά. Να γράψετε πρόγραμμα σε Python που θα διαβάζει το βάρος και τη σειρά με την οποία κάθε άτομο εισέρχεται στο ασανσέρ (π.χ. 45 1, 89 2). Το πρόγραμμα θα τερματίζει, όταν το ασανσέρ γεμίσει (σε σχέση με το μέγιστο επιτρεπτό όριο ασφαλείας). Στη συνέχεια, θα εμφανίζει τη σειρά του τελευταίου ατόμου που κατάφερε να μπει στο ασανσέρ.

**Απάντηση**

```
weight = int( raw_input(" Βάρος ατόμου σε κιλά = ") )
pos = int( raw_input(" Σειρά = ") )
persons = 0
capacity = 500
while weight <= capacity :
    pos = int( raw_input(" Σειρά = ") )
    capacity = capacity – weight
    persons = persons + 1
    weight = int( raw_input(" Βάρος ατόμου σε κιλά = ") )
print " Σειρά τελευταίου " , pos
```

**Δραστηριότητα 30.**

Σε ένα πλοίο υπάρχουν εισιτήρια Α' Θέσης (κωδικός 0) προς 50€ και Β' θέσης (κωδικός 1) προς 20€, το ένα. Ο μέγιστος επιτρεπόμενος αριθμός επιβατών είναι 400 άτομα και θεωρούμε ότι τελικά το πλοίο γέμισε για το συγκεκριμένο προορισμό που εξετάζουμε.
Να γράψετε πρόγραμμα σε Python, το οποίο να:
1) διαβάζει την κατηγορία εισιτηρίου (κωδικός 0 ή 1) για κάθε επιβάτη
2) εμφανίζει το πλήθος των επιβατών της Α' θέσης
3) εμφανίζει το συνολικό ποσό που πλήρωσαν όλοι οι επιβάτες.

**Απάντηση**

```
passengers = 0
while passengers < 400 :
    cat = int( raw_input(" Κατηγορία εισιτηρίου (0/1) = ") )
    passengers = passengers + 1
    if cat == 0 :
        k = k + 1

print " Επιβάτες Α’ θέσης : " , k
total = k * 50 + (400 - k)*20
print " Συνολικό ποσό " , total
```

**Δραστηριότητα 31.**

Να συμπληρώσετε τα κενά στο παρακάτω πρόγραμμα, ώστε να:
1) εμφανίζει όλους τους αριθμούς από 1 μέχρι 100
2) εμφανίζει όλους τους αριθμούς από 1 μέχρι 100, αλλά με αντίστροφη σειρά
3) εμφανίζει όλους τους άρτιους αριθμούς από 20 μέχρι 80.

**Απάντηση**

```
for number in range( 1 , 101 , 1 )      # ερώτημα 1
for number in range( 100, 0 , –1 )      # ερώτημα 2
for number in range( 20, 80, 2 )        # ερώτημα 3
```

**Δραστηριότητα 32.**

Να γραφεί πρόγραμμα σε Python το οποίο:
1) Να διαβάζει επαναληπτικά ακέραιους αριθμούς, μέχρις ότου δοθεί ο αριθμός 0.
2) Να εμφανίζει στο τέλος, το πλήθος των θετικών αριθμών.
3) Να υπολογίζει και να εμφανίζει στο τέλος, το άθροισμα όλων των αριθμών που διαβάστηκαν (δόθηκαν από το πληκτρολόγιο).

**Απάντηση**

```
number = int( raw_input(" Επόμενος αριθμός = ") )
positives = 0
sum = 0
while number != 0 :
    sum = sum + number
    if number > 0 :
        positives = positives + 1
    number = int( raw_input(" Επόμενος αριθμός = ") )
print " Πλήθος θετικών αριθμών : " , positives
print " Συνολικό Άθροισμα : " , sum
```

**Δραστηριότητα 33.**

Να γράψετε, σε Python, πρόγραμμα με το οποίο: Να καταχωρούνται επαναληπτικά: ο αριθμός κυκλοφορίας οχήματος και ποσό κλήσης από παρκάρισμα ή άλλη αιτία, με την καταχώρηση να επαναλαμβάνεται μέχρι να δοθεί αριθμός κυκλοφορίας 99.

Στο τέλος να εμφανίζει:
1. Το πλήθος των οχημάτων που καταχωρήθηκαν.
2. Το συνολικό ποσό κλήσεων που θα εισπραχθεί.
3. Τον αριθμό κυκλοφορίας του τελευταίου από τα οχήματα που έλαβαν το μέγιστο ποσό κλήσης, καθώς και το ποσό της κλήσης αυτής.

```
number = int( raw_input(" Επόμενος αριθμός οχήματος = ") )
vehicles = 0
total = 0
max = 0
max_vehicle = number
while number != 99 :
    amount = int( raw_input(" Ποσό κλήσης= ") )
    total = total + amount
    if amount > max :
        max = amount
        max_vehicle = number
    elif amount == max :
        max_vehicle = number
        vehicles = vehicles +1
        number = int( raw_input(" Επόμενος αριθμός οχήματος = ") )
print " Πλήθος οχημάτων : " , vehicles
print " Συνολικό Ποσό : " , total
print " Μέγιστο ποσό : " , max
print " τελευταίο όχημα με μέγιστο ποσό : " , max_vehicle
```

# Μέρος ΙΙ

# ΚΕΦΑΛΑΙΟ 5 Κλασικοί Αλγόριθμοι ΙΙ

## Απαντήσεις στις δραστηριότητες του κεφαλαίου 5 του Βιβλίου Μαθητή

### Δραστηριότητα 1

Να αντιστοιχίσετε τους παρακάτω αλγόριθμους με τις κατάλληλες λειτουργίες:

| Αλγόριθμος | Λειτουργία |
|---|---|
| 1. Ταξινόμηση με επιλογή | Α. Αντιμετάθεση ζευγών |
| 2. Ταξινόμηση ευθείας ανταλλαγής | Β. Τοποθέτηση στοιχείου σε ταξινομημένη λίστα |
| 3. Ταξινόμηση με εισαγωγή | Γ. Εύρεση ελαχίστου |

### Απάντηση

Η αντιστοίχιση έχει ως εξής:
**1Γ.** (Ταξινόμηση με επιλογή - Εύρεση Ελαχίστου)
**2Α.** (Ταξινόμηση ευθείας ανταλλαγής - Αντιμετάθεση ζευγών)
**1Α.** (Ταξινόμηση με εισαγωγή - Τοποθέτηση στοιχείου σε ταξινομημένη λίστα)

### Δραστηριότητα 2

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

### Απάντηση

Αν αλλάξουμε μόνο τον τελεστή σύγκρισης στην findMinPosition από "<" σε ">", ώστε να βρίσκει το μεγαλύτερο αντί το μικρότερο, τότε και η συνάρτηση selectionSortAscending θα ταξινομεί σε φθίνουσα αντί σε αύξουσα.

```
def findMinPosition(start, end, List):
    position = start
    for i in range(start, end):
        if List[ i ] > List[ position ] :
            position = i
    return position

def selectionSortAscending(List):
    position = None
    n = len(List)
    for i in range(0,n):
        position = findMinPosition(i, n, List)
        List[ i ], List[ position ] = List[ position ], List [ i ]
    return List
```

Καταφέραμε να αλλάξουμε τη λειτουργία της selectionSortAscending χωρίς να πειράξουμε τίποτα στον κώδικά της. Αυτό είναι ένα παράδειγμα της ανεξαρτησίας και της ευελιξίας του τμηματικού προγραμματισμού.

**Δραστηριότητα 3**

Να γράψετε μια συνάρτηση σε Python, η οποία θα δέχεται μια λίστα, θα ελέγχει αν τα στοιχεία της είναι σε αύξουσα σειρά και θα επιστρέφει αντίστοιχα True ή False. Προτείνεται να χρησιμοποιήσετε μια λογική μεταβλητή.

**Απάντηση**

Η λογική μεταβλητή ascending παραμένει True όσο δεν βρίσκουμε ένα ζευγάρι στοιχείων για το οποίο δεν ισχύει η σχέση που θέλουμε. Δηλαδή ψάχνουμε ένα ζευγάρι που να είναι σε φθίνουσα σειρά. Αν βρούμε ένα τουλάχιστον, δεν έχει πλέον νόημα να συνεχίσουμε.

```
# με τη χρήση λογικής μεταβλητής
def isAscending(myList):
    ascending = True
    i = 0
    N = len(myList)
    while ascending and i<N-1 :
        if (myList[ i ] > myList[ i+1 ]) :
            ascending = False
        i = i +1
    return ascending
```

```
# χωρίς τη χρήση λογικής μεταβλητής
def isAscending2(myList):
    i = 0
    pred = myList[ 0 ]
    for item in myList :
        if item < pred:
            return False
        else:
            pred = item
    return True
```

**Δραστηριότητα 4** (βελτιωμένη φυσαλίδα)

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

**Υπόδειξη:** Χρησιμοποιήστε μια λογική μεταβλητή η οποία θα αλλάζει τιμή, αν υπάρχουν τουλάχιστον δύο στοιχεία τα οποία δε βρίσκονται στην επιθυμητή σειρά, καθώς η "φυσαλίδα ανεβαίνει στην επιφάνεια".

**Απάντηση**

Χρησιμοποιούμε την ιδέα, όπου η λογική μεταβλητή isSorted μας δίνει την πληροφορία, αν η λίστα είναι ταξινομημένη στο τέλος κάθε περάσματος. Αν γίνει έστω και μία αντιμετάθεση, η isSorted θα γίνει False. Αν δεν γίνει αντιμετάθεση, σημαίνει ότι όλα τα στοιχεία είναι στη σωστή σειρά, άρα ταξινομημένα, οπότε ο αλγόριθμος δεν έχει λόγο να συνεχίσει.

```
# 1ος τρόπος με λογική μεταβλητή
def optimizedBubbleSort( A ):
    N = len( A )
    isSorted = False
    i=1
    while i < N and not isSorted :
        isSorted = True
        for j in range(N-1, i-1, -1):
            if A[ j ] < A[ j-1 ]:
            A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
            isSorted = False
        i = i + 1
```

```
# 2ος τρόπος με βίαιη διακοπή της επανάληψης for...in..
def optimizedBubbleSort2( A ):
    N = len( A )
    for i in range( N ):
        isSorted = True
        for j in range(N-1, i, -1):
            if A[ j ] < A[ j-1 ] :
                A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
                isSorted = False
        if isSorted:
                return
```

**Δραστηριότητα 5**

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

**Απάντηση**

Ο πρώτος, πιο απλός και γρήγορος τρόπος, είναι να μετρήσουμε πόσα είναι τα True. Αν είναι έστω k, οπότε γνωρίζουμε το μέγεθος (len) της λίστας, μπορούμε να βρούμε και τα False ( μέγεθος – k). Στη συνέχεια, θέτουμε True στα πρώτα k στοιχεία και False στα υπόλοιπα.

```
#1ος τρόπος: Μετράμε πόσα είναι τα True :
def count_true_values( booleanList ) :
    true_values = 0
    for item in booleanList :
    if item :                          # ή if item == True :
        true_values = true_values + 1
    return true_values

# 2ος τρόπος (True=1,False=0). Μετράμε πόσα είναι τα True
def count_true_values2( booleanList ) :
    true_values = 0
    for item in booleanList :
        true_values = true_values + item
    return true_values
```

```
# 1ος τρόπος: Μετράμε πόσα είναι τα True και στη συνέχεια
# βάζουμε τόσα True στα πρώτα στοιχεία του πίνακα και στα
# υπόλοιπα False.
def swapBooleanbyCounting( List ):
    N = len( List )
    true_values = count_true_values( List )
    for i in range(true_values):
        List[ i ] = True
    for i in range(true_values, N):
        List[ i ] = False
    return List
```

Στο δεύτερο τρόπο ξεκινάμε με δύο δείκτες στα άκρα της λίστας και αντιμεταθέτουμε τα αντιδιαμετρικά στοιχεία (True, False) μέχρι να συναντήσουμε από αριστερά False ή από δεξιά True. Οι δείκτες κινούνται ταυτόχρονα και αντίθετα. Όταν συναντηθούν, έχουμε αντιμεταθέσει όλες τις τιμές που ήταν στη λάθος πλευρά.

```
def swapBooleanbyComparison( List ):
    N = len( List )
    left = 0
    right = N-1
    while left < right:
        if (List[ right ]==True and List[ left ]==False ) :
            List[ left ],List[ right ] = List[ right ],List[ left ]
            left = left+1
            right = right-1
        elif (List[ right ]==False):
            right = right-1
        else :
            left = left+1
    return List
```

**Δραστηριότητα 6**

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

**Απάντηση**

Αρχικά σχεδιάζουμε μια συνάρτηση για να διαβάζουμε μια λίστα, η οποία θα μας χρειαστεί και σε επόμενες δραστηριότητες. Η συνάρτηση διαβάζει από το πληκτρολόγιο μια λίστα από αντικείμενα, ενώ η ανάγνωση της λίστας σταματάει, όταν δοθεί η τιμή None και ακολούθως επιστρέφει τη λίστα ως τιμή της συνάρτησης. Αν θέλουμε να εισάγουμε αλφαριθμητικά, πρέπει να τα εισάγουμε ανάμεσα σε εισαγωγικά (quotes), για παράδειγμα "Γυμνάσιο". Μπορούμε να διαπιστώσουμε ότι η συνάρτηση δουλεύει για κάθε τύπο, κάτι που αποτελεί ένα από τα ισχυρά πλεονεκτήματα της Python.

```
def readList( ) :
    print "****************************************"
    print "* Για το τέλος δώσε την τιμή None *"
    print "****************************************"
    index = 0
    L = [ ]
    value = input("L[" + str(index) + "] = ")
    while value != None :
        L.append(value)
        index += 1
        value = input("L[" + str(index) + "] = ")
    return L
```

**1ος τρόπος:** Μετράμε πόσα είναι τα True, διπλασιάζουμε το ποσό και το συγκρίνουμε με το πλήθος των στοιχείων της λίστας. Έτσι βρίσκουμε ποια τιμή εμφανίζεται τις λιγότερες φορές. Αυτό θα είναι και το πλήθος των ζευγών (True, False).

```
def program6_Counting( ):
    List = readList( )
    N = len( List )

    # μετράει πόσα είναι τα True (ορίστηκε στην δραστηριότητα 5)
    true_values = count_true_values( List )

    # είναι τα True περισσότερα από τα False;
    if (2*true_values < N):
        minValue = True
        couples = true_values
    else:
        minValue = False
        couples = N – true_values
    for i in range(0, 2*couples, 2):
        List[ i ] = True
        List[ i+1 ] = False
    for i in range(2*couples, N):
        List[ i ] = not minValue
    return List
```

**Δραστηριότητα 7**

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

**Απάντηση**

**1ος τρόπος:** Αφού οι τιμές True, False είναι μισές-μισές, τότε θέτουμε στις πρώτες μισές θέσεις True και στις υπόλοιπες False.

```
    # 1ος τρόπος: θέτουμε τα πρώτα N/2 True
    # και τα επόμενα N/2 False.
def splitBoolean_byCounting( List ):
    mid = len(List) / 2
    for index in range(mid):
        List[ index ] = True
        List[ index + mid ] = False
    return List
```

**2ος τρόπος:** Αρκούν N/2 περάσματα του αλγορίθμου ταξινόμησης.

```
def splitBoolean_bySorting( List ):
    N = len( List )
    mid = N / 2
    for i in range(mid):
        for j in range(N-1, i, -1) :
            if List[ j ] > List[ j-1 ] :                    # True > False
                List[ j-1 ], List[ j ] = List[ j ],List[ j-1 ]
    return List
```

**Δραστηριότητα 8**

Το πρόβλημα της ολλανδικής σημαίας αναφέρεται στην αναδιάταξη μιας λίστας γραμμάτων, η οποία περιέχει μόνο τους χαρακτήρες R, W, B (Red, White, Blue), έτσι ώστε όλα τα R να βρίσκονται πριν από τα W και όλα τα W να βρίσκονται πριν από B. Να τροποποιήσετε έναν από τους αλγόριθμους ταξινόμησης που παρουσιάστηκαν σε αυτήν την ενότητα, ώστε να επιλύει αυτό το πρόβλημα.

**Απάντηση**

Αρκεί να αλλάξουμε τη συνθήκη του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής, έτσι ώστε να ισχύει η διάταξη RWB. Έτσι σχηματίζουμε συνθήκη με όλες τις περιπτώσεις. Το σκεπτικό είναι ότι η αντιμετάθεση μεταξύ δυο στοιχείων θα γίνει, αν δεν ισχύει η σειρά R W B.

```
def dutchFlag( L ):
    N = len( L )
    for i in range( N ):
        for j in range(N-1, i, -1) :
            if (L[ j ]=='W' and L[ j-1 ]=='B') \
                or (L[ j ]=='R' and L[ j-1 ]=='W') \
                or (L[ j ]=='R' and L[ j-1 ]=='B') :
                    L[ j-1 ], L[ j ] = L[ j ], L[ j-1 ]
    return L
```

**Σημείωση:** Αν μια εντολή δε χωράει σε μια γραμμή και θέλουμε να συνεχιστεί στην επόμενη, χρησιμοποιούμε το σύμβολο \ .

**Δραστηριότητα 9**

Να γράψετε μια συνάρτηση σε Python η οποία διαβάζει αριθμούς από το χρήστη και στη συνέχεια τους τοποθετεί σε μια λίστα σε φθίνουσα σειρά, την οποία και επιστρέφει. Κάθε φορά που διαβάζει ένα νέο αριθμό, τον τοποθετεί στη σωστή θέση στην ήδη ταξινομημένη λίστα, ώστε να διατηρείται η φθίνουσα διάταξη των στοιχείων της λίστας. Η εισαγωγή των αριθμών σταματάει, όταν δοθεί η τιμή None. Ποιον αλγόριθμο ταξινόμησης σας θυμίζει η παραπάνω λειτουργία; Σε τι διαφέρει η συνάρτηση που θα αναπτύξετε από τον αλγόριθμο αυτόν;

**Απάντηση**

Θέλουμε έναν αλγόριθμο ταξινόμησης ο οποίος να ταξινομεί σταδιακά ένα μέρος του πίνακα (incremental). Δε θέλουμε κάθε φορά που έρχεται ένα νέο στοιχείο, να εκτελούμε πάλι ταξινόμηση όλου του πίνακα, αλλά να τοποθετούμε το νέο στοιχείο στη σωστή θέση, έτσι ώστε ο πίνακας να παραμένει ταξινομημένος. Αυτή είναι η βασική ιδέα του αλγορίθμου ταξινόμησης με εισαγωγή. Κάθε φορά που διαβάζουμε ένα νέο αριθμό, εκτελούμε αναζήτηση για να βρούμε τη θέση που πρέπει να εισαχθεί και παράλληλα μετακινούμε μια θέση δεξιά, όσα στοιχεία πρέπει να βρίσκονται μετά από αυτόν.

```
def insertionSort( ) :
    print "*Δώσε την τιμή None για να σταματήσεις*"
    number = input("number = ")
    L = [ ]
    while (number is not None) :     # number != None
        L.append(number)
        value = L[ len(L) - 1 ]
        j = len(L) – 1
        while j > 0 and L[ j-1 ] < value :
            L[ j ] = L[ j-1 ]
            j = j-1
        L[ j ] = value   # έξω από τη while
        number = input("number = ")
    return L
```

**Δραστηριότητα 10**

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

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

**Απάντηση**

Αρχικά, το πρόγραμμα διαβάζει δύο λίστες με τη συνάρτηση readList που έχουμε ορίσει σε προηγούμενη δραστηριότητα. Στη συνέχεια, εφαρμόζουμε έναν αλγόριθμο ταξινόμησης, ώστε να μπορεί να χρησιμοποιηθεί η δυαδική αναζήτηση μετά, για τον έλεγχο της ύπαρξης ενός αριθμού της λίστας Β στη λίστα Α. Για την καλύτερη οργάνωση του προγράμματος, αναπτύξαμε κάποιες συναρτήσεις.

Αρχικά ορίζουμε την ταξινόμηση ευθείας ανταλλαγής:

```
# Ταξινομεί με τον αλγόριθμο της ευθείας ανταλλαγής τη λίστα Α
def bubbleSort( A ):
    N = len( A )
    for i in range( N ):
        for j in range(N-1, i, -1):
            if A[ j ] < A[ j-1 ] :
                A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
```

Ορίζουμε τη συνάρτηση που υλοποιεί τη δυαδική αναζήτηση:

```
# Ελέγχει την ύπαρξη ενός στοιχείου key στην ταξινομημένη
# λίστα L υλοποιώντας τον αλγόριθμο της δυαδικής αναζήτησης
def binarySearch( A, key ):
    last = len( A ) - 1
    first = 0
    found = False
    while first <= last and not found:
        mid = (last + first) // 2
        if A[ mid ] == key :
            found = True
        elif A[ mid ] < key :
            first = mid + 1
        else:
            last = mid - 1
    return found
```

Στη συνέχεια παραθέτουμε το πρόγραμμα που χρησιμοποιεί τις παραπάνω συναρτήσεις. Παρατηρήστε ότι αυτό δε δουλεύει μόνο για αριθμούς. Δουλεύει επίσης για αλφαριθμητικά αλλά και για λογικές τιμές. Δοκιμάστε να εκτελέσετε το παραπάνω πρόγραμμα και με άλλους τύπους δεδομένων, όπως με αλφαριθμητικά. Αυτό για να εξοικειωθούν οι μαθητές με την ιδέα ότι στην Python έχουμε τη δυνατότητα να ορίζουμε τη λειτουργία του αλγόριθμου ανεξάρτητα από το είδος των δεδομένων. Δηλαδή δε χρειάζεται να ορίσουμε διαφορετικό αλγόριθμο για αλφαριθμητικά, διαφορετικό για αριθμούς κ.ο.κ.

```
def program_10( ):

    A = readList( )
    B = readList( )
    bubbleSort( A )

    count = 0
    for item in B :
        if binarySearch( A, item ) :

        count = count + 1

    print "Πλήθος κοινών στοιχείων = ", count
```

## Λύσεις Δραστηριοτήτων Τετραδίου Εργασιών Μαθητή

**Δραστηριότητα 1.**

Να γράψετε αλγόριθμο σε Python ο οποίος να διαβάζει για κάθε μαθητή το όνομα και το βαθμό του και να τα καταχωρεί σε δύο λίστες student και grade. Η εισαγωγή των δεδομένων σταματάει,όταν δοθεί αντί για όνομα η τιμή 0. Στη συνέχεια:

1) Να εμφανίζει τα ονόματα και τους βαθμούς των μαθητών σε αλφαβητική σειρά, κάνοντας κατάλληλη ταξινόμηση με βάση τη λίστα student.
2) Να διαβάζει το όνομα ενός μαθητή και να εμφανίζει τον αντίστοιχο βαθμό του, αφού προηγουμένως έχει βρει τη θέση του στη λίστα, αξιοποιώντας το γεγονός ότι τα ονόματα είναι σε αλφαβητική σειρά.
3) Να ταξινομεί τους μαθητές με βάση τους βαθμούς τους και να εμφανίζει τα ονόματα των τριών πρώτων μαθητών. Να χρησιμοποιήσετε τον αλγόριθμο ταξινόμησης που ήδη έχετε υλοποιήσει ως συνάρτηση.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή.

**Δραστηριότητα 2.** Ταξινόμηση ευθείας ανταλλαγής

Δίνεται η παρακάτω λίστα Α με 7 αριθμούς. Να εκτελέσετε τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής για την ταξινόμηση των αριθμών σε αύξουσα σειρά, συμπληρώνοντας παράλληλα τα κενά στον παρακάτω πίνακα, έτσι ώστε να φαίνονται τα στοιχεία της λίστας αμέσως μετά από κάθε πέρασμα του αλγορίθμου.

**Λύση**

| | **A[0]** | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** |
|---|---|---|---|---|---|---|---|
| | 55 | 34 | 5 | 3 | 2 | 1 | 1 |
| **1ο πέρασμα** | 1 | 55 | 34 | 5 | 3 | 2 | 1 |
| **2ο πέρασμα** | 1 | 1 | 55 | 34 | 5 | 3 | 2 |
| **3ο πέρασμα** | 1 | 1 | 2 | 55 | 34 | 5 | 3 |
| **4ο πέρασμα** | 1 | 1 | 2 | 3 | 55 | 34 | 5 |
| **5ο πέρασμα** | 1 | 1 | 2 | 3 | 5 | 55 | 34 |
| **6ο πέρασμα** | 1 | 1 | 2 | 3 | 5 | 34 | 55 |

**Δραστηριότητα 3.** Διερεύνηση του αλγορίθμου ταξινόμησης

Να εκτελέσετε την παρακάτω συνάρτηση συμπληρώνοντας τα κενά μέχρι να καταλήξετε στις κατάλληλες τιμές, ώστε να γίνεται η ταξινόμηση σωστά για τη λίστα A = [ 0, 1, 2, 3, 4, 5]

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή.

**Δραστηριότητα 4.**

Να γράψετε ένα πρόγραμμα, σε Python, το οποίο θα διαβάζει ονόματα, μέχρι να δοθεί ως όνομα το '0' και θα τα εμφανίζει σε αλφαβητική σειρά.
**Επέκταση:** Στη συνέχεια το πρόγραμμά σας θα διαβάζει ένα όνομα και θα εμφανίζει κατάλληλο μήνυμα, αν το όνομα αυτό εμφανίζεται στη λίστα με τα ονόματα ή όχι.

**Απάντηση**

Θα χρησιμοποιήσουμε τις συναρτήσεις που υλοποιήσαμε στην δραστηριότητα 1, bubbleSort για ταξινόμηση και binarySearch για δυαδική αναζήτηση. Αρχικά, αποθηκεύουμε τα ονόματα που διαβάζουμε σε μια λίστα.

```
name = raw_input("όνομα = ")
nameList = [ ]
while name != '0' :
    nameList.append( name )
    name = raw_input("όνομα = ")
bubbleSort( nameList )
name = raw_input("δώσε ένα όνομα: ")
found = binarySearch( nameList, name )
if found >= 0:
    print "Το όνομα υπάρχει στη λίστα"
else :
    print "Το όνομα δεν υπάρχει στη λίστα"
```

**Δραστηριότητα 5.**

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει βαθμούς μαθητών, μέχρι να δοθεί αρνητικός και στη συνέχεια θα εμφανίζει τους τρεις μεγαλύτερους βαθμούς που διάβασε.

**Απάντηση**

Διαβάζουμε τους βαθμούς και τους καταχωρούμε σε μια λίστα. Στη συνέχεια, αφού βρούμε το μεγαλύτερο βαθμό, θέτουμε -1 στη θέση του, έτσι ώστε, στην επόμενη επανάληψη, να βρούμε το δεύτερο μεγαλύτερο κ.ο.κ.

```
grade = input("βαθμός = ")
gradeList = [ ]
while grade >= 0 :
    gradeList.append( grade )
    grade = input("βαθμός = ")
maximum = [-1, -1, -1]
pos = -1
for i in range(3):
    for j in range( len(gradeList) ):
        if gradeList[ j ] > maximum[ i ]:
            maximum[ i ] = gradeList[ j ]
            pos = j

# Διαγράφουμε την τιμή του μέγιστου
gradeList[pos] = -1

print maximum[0], maximum[1], maximum[2]
```

**Δραστηριότητα 6. Αλγόριθμος Δυαδικής αναζήτησης**

Δίνεται παρακάτω η λίστα Α με 14 αριθμούς. Να εκτελέσετε τον αλγόριθμο της Δυαδικής αναζήτησης για τον αριθμό 100 και να γράψετε στο τετράδιό σας τους αριθμούς που θα συγκριθούν με το 100 κατά την εκτέλεση του αλγορίθμου.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | 94 | 96 | 99 |

Πόσες συγκρίσεις χρειάστηκαν μέχρι να διαπιστώσει ο αλγόριθμος ότι το 100 δεν υπάρχει στον πίνακα;

**Απάντηση**

Θα συγκριθούν κατά σειρά οι αριθμοί: 13, 89, 96, 99, δηλαδή θα γίνουν 4 συγκρίσεις.

**Δραστηριότητα 7.** Αλγόριθμος Δυαδικής αναζήτησης

Να συμπληρώσετε τα κενά στο παρακάτω πρόγραμμα σε Python, έτσι ώστε να εκτελεί τον αλγόριθμο της Δυαδικής αναζήτησης ενός στοιχείου key σε μια λίστα array.

**Απάντηση**

```
def binarySearch( array, key ) :
    first = 0
    last = len( array ) – 1
    found = False
    while first <= last and not found :
        mid = ( first + last ) // 2
        if array[ mid ] == key :
            found = True
        elif array[mid] < key :
            first = mid + 1
        else :
            last = mid – 1
    return found
```

**Δραστηριότητα 8.**

Να γράψετε ένα πρόγραμμα το οποίο να διαβάζει τους βαθμούς και τα επίθετα των μαθητών της τάξης σας και στη συνέχεια, να τα εμφανίζει σε φθίνουσα σειρά ως προς τους βαθμούς, έτσι ώστε δηλαδή όσοι έχουν καλύτερους βαθμούς να είναι πρώτοι.

**Απάντηση**

```
N = input("Πόσους μαθητές έχει η τάξη; N = ")
gradeList = [ ]
nameList = [ ]
for i in range( N ):
    name = raw_input("όνομα = ")
    grade = input("βαθμός = ")
    nameList.append(name)
    gradeList.append( grade )
for i in range( N ):
    for j in range( N-1, i, -1 ):
        if gradeList[ j ] > gradeList[ j-1 ] :
            gradeList[j], gradeList[j-1] = gradeList[j-1], gradeList[j]
            nameList[j], nameList[j-1] = nameList[j-1], nameList[j]
for i in range(N):
    print nameList[i]
```

**Δραστηριότητα 9.** Αλγόριθμος ευθείας ανταλλαγής

Να συμπληρώσετε τα κενά στο παρακάτω πρόγραμμα, έτσι ώστε να ταξινομεί τα στοιχεία της λίστας array σε φθίνουσα σειρά, σύμφωνα με τον αλγόριθμο της ευθείας ανταλλαγής.

**Απάντηση**

```
array = [9, -2, 6, 4, -8, 12, 5, 18]
N = len( array )
for i in range( 0 , N , 1):
    for j in range( N–2 , i–1, –1 ):
        if array[ j+1 ] > array[ j ] :
            array [ j ] , array [ j+1 ] = array[ j+1 ] , array[ j ]
```

**Δραστηριότητα 10.** Αλγόριθμος ευθείας ανταλλαγής

Πόσα περάσματα θα χρειαστούν για να ταξινομηθούν οι παρακάτω λίστες με τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής σε αύξουσα σειρά;

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| α) | 55 | 46 | 44 | 34 | 28 | 18 | 14 | 12 | 10 | 8 | 4 |
| β) | 4 | 6 | 8 | 18 | 30 | 50 | 56 | 68 | 70 | 3 | 1 |
| γ) | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 30 | 30 | 30 | 30 |
| δ) | 28 | 28 | 28 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |

**Απάντηση**

**α)** 10, **β)** 2, **γ)** 0 **δ)** 8

# ΚΕΦΑΛΑΙΟ 6 Διαχείριση Αρχείων

## Λύση Δραστηριοτήτων βιβλίου Μαθητή

Οι Δραστηριότητες 1 και 2 αναπτύσσονται στο βιβλίο μαθητή

### Δραστηριότητα 3

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

### Απάντηση

```
# Άνοιγμα αρχείου για ανάγνωση
f = open ('demoFile.txt', 'r')

for line in reversed(open("demoFile.txt").readlines()):
    print line

final = [ ]
for line in f :
    final.append(line)

for i in range(len(final)-1,0,-1):
    print final[i]

final.reverse()
for line in final :
    print line
f.close()
```

**2η λύση**

```
f = open ('demoFile.txt', 'r')
for line in (open('demoFile.txt').readlines()):
    print line
final = [ ]
for line in f :
    final.append(line)
final.reverse()
for line in final :
    print line
fout = open ('output.txt', 'w')
for line in final :
    fout.write (line)

f.close()
fout.close()
```

**Δραστηριότητα 4**

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

**Απάντηση**

```
# Κατασκευή λίστας με τα τετράγωνα των αριθμών από 1 έως 10

my_list = [i**2 for i in range(1,11)]

# Άνοιγμα αρχείου κειμένου για εγγραφή
f = open('output.txt', 'w')

# Εγγραφή των στοιχείων της λίστας στο αρχείο

for item in my_list:
    f.write(str(item) + '\n') # δέχεται string όρισμα n write
f.close()
# Άνοιγμα του αρχείου για ανάγνωση και εκτύπωση των περιεχομένων του
f = open('output.txt', 'r')
print f.readline()
print f.read()
f.close()
```

## Λύσεις Δραστηριοτήτων Τετραδίου Εργασιών Μαθητή

**Δραστηριότητα 1.**

Να γράψετε, σε Python, ένα πρόγραμμα το οποίο:
Θα δημιουργεί μια λίστα με όλους τους άρτιους αριθμούς από το 2 μέχρι το 1000 και θα γράφει αυτούς τους αριθμούς σε ένα αρχείο, έναν σε κάθε γραμμή.
Στη συνέχεια, θα διαβάζει τους αριθμούς από το αρχείο και θα γράφει το άθροισμά τους στο τέλος του αρχείου, χωρίς να διαγράψει τα περιεχόμενα του αρχείου.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Μελέτης Μαθητή

**Δραστηριότητα 2.**

Να γράψετε, σε Python, ένα πρόγραμμα το οποίο θα διαβάζει ένα κείμενο από ένα αρχείο και θα εμφανίζει τις λέξεις κάθε πρότασης, σε αντίστροφη σειρά, δηλαδή την πρώτη λέξη τελευταία και την τελευταία πρώτη.
Δεχόμαστε ότι κάθε πρόταση τελειώνει με τελεία '.' και οι λέξεις είναι χωρισμένες με ένα κενό. Δεν υπάρχουν άλλα σημεία στίξεως.

**Απάντηση**

Θα χρησιμοποιήσουμε μια λίστα την οποία γνωρίζουμε από το μάθημα της Β Λυκείου.

```
def ds3():
    input = open('input.txt', 'r')
    for line in input:
        word_list = [ ]
        word = ""
        for symbol in line:
            if symbol == " " :
                word_list.insert(0,word)
                word = ""
            elif symbol == "." :
                word_list.insert(0,word)
            else :
                word += symbol
    for word in word_list:
        print word,
    print
input.close()
```

**Δραστηριότητα 3.**

Να γράψετε ένα πρόγραμμα το οποίο θα ενώνει δύο αρχεία κειμένου σε ένα, τοποθετώντας τα περιεχόμενα του δεύτερου αρχείου μετά από αυτά του πρώτου.

**Απάντηση**

```
outfile = open('file2.txt', 'a')
infile = open('file1.txt', 'r')
for line in infile:
outfile.write(line)

infile.close()
outfile.close()
```

**Δραστηριότητα 4.**

Να υλοποιήσετε μια συνάρτηση copy(source, destination) η οποία θα δημιουργεί ένα αντίγραφο του αρχείου με όνομα source στο αρχείο με όνομα destination.

**Απάντηση**

```
def copy(source, destination):
    inputfile = open(source,"r")
    outputFile = open(destination,"w")
    for line in inputfile:
        outputFile.write(line)
    inputfile.close()
    outputFile.close()

copy("input.txt", "output.txt")
```

# ΚΕΦΑΛΑΙΟ 7 Προηγμένα στοιχεία γλώσσας προγραμματισμού

Οι λύσεις στις δραστηριότητες του Βιβλίου Μαθητή αναπτύσσονται στο ίδιο το βιβλίο.

## Λύσεις Δραστηριοτήτων Τετραδίου Εργασιών Μαθητή

### Δραστηριότητα 1.

Ανοίγουμε ένα νέο αρχείο στην Python στο οποίο θα προσθέσουμε τους ορισμούς των συναρτήσεων που θα δώσουμε παρακάτω.
Ορίζουμε τη συνάρτηση python3 η οποία εμφανίζει τη λέξη python 3 φορές. Επίσης ορίζουμε και τη συνάρτηση python9 που εμφανίζει τη λέξη python 9 φορές.

```
def python3():
    print "python"
    print "python"
    print "python"
>>> python3()
python
python
python
```

```
def python9():
    print "python"
    print "python"
    print "python"
    print "python"
    print "python"
    print "python"
    print "python"
    print "python"
    print "python"
```

Να ξαναγράψετε τη συνάρτηση python9 χρησιμοποιώντας λιγότερες εντολές.
Να ορίσετε μια συνάρτηση η οποία να εμφανίζει 21 φορές τη λέξη python με αποκλειστική χρήση των δυο παραπάνω συναρτήσεων, χρησιμοποιώντας όσο το δυνατόν λιγότερες εντολές.

**Απάντηση:**

```
def python9():
    python3()
    python3()
    python3()

def python21():
    python9()
    python9()
    python3()
```

**Δραστηριότητα 2.**

Δίνονται οι παρακάτω συναρτήσεις σε Python:

```
def python3():
    for i in range(3):
        print "python"
```

```
def python12():
    for i in range(12):
        print "python"
```

```
def python100():
    for i in range(100):
        print "python"
```

Συμπληρώστε τον ορισμό της παρακάτω συνάρτησης, ώστε να αποτελεί γενίκευση των προηγούμενων και στη συνέχεια δώστε τις διπλανές κλήσεις στο διερμηνευτή:

```
def python(N):
    for i in range(N):
        print "python"
```

```
>>> python(3)
>>> python(9)
>>> python(21)
```

Σε τι διαφέρει η τελευταία συνάρτηση από όλες τις προηγούμενες; Ποια είναι η σχέση της με αυτές;

**Απάντηση**

Η τελευταία συνάρτηση αποτελεί γενίκευση όλων των προηγουμένων. Για N=3 το αποτέλεσμα είναι ίδιο με αυτό της printPython3, για N=9 ίδιο με αυτό της printPython9 κ.ο.κ. Το N είναι η παράμετρος που ορίζει κάθε φορά πόσα μηνύματα θέλουμε να εμφανίσουμε στην οθόνη.

**Δραστηριότητα 3.**

Δώστε τις παρακάτω εντολές στο διερμηνευτή της Python. Τι παρατηρείτε; Τι πιστεύετε ότι κάνει η εντολή return;

```
>>> import math
>>> def root(number):
        return math.sqrt(number)
>>> root( 16 ) ; root( 2 )
```

```
>>> a = root( 16 )
>>> print a
>>> print root(a)
>>> print root(root(16))
```

**Απάντηση**

Επιστρέφει το αποτέλεσμα της συνάρτησης

| | |
|---|---|
| 4.0 | 4.0 |
| 1.4142135623730951 | 2.0 |
| | 2.0 |

**Δραστηριότητα 4.**

Να εντοπίσετε στο παρακάτω πρόγραμμα τα τμήματα κώδικα που πρέπει να γίνουν συναρτήσεις και να το ξαναγράψετε, έτσι ώστε να αποφευχθεί η επανάληψή τους και να μειωθεί ο όγκος του.

```
sum1 = 0
for i in range(100):
    sum1 = sum1 + i
print sum1
sum2 = 0
for j in range(100):
    sum2 = sum2 + j
print sum2 + sum1
sum3 = 0
for k in range(sum1):
    sum3 = sum3 + k
print sum3 + sum2 + sum1
```

**Απάντηση**

Επαναλαμβάνονται οι εντολές επανάληψης for

```
def f1(N):
    sum = 0
    for i in range(N):
        sum = sum + i
    return sum
sum=f1(100) + f1(100) +f1(f1(100))
print sum
```

**Δραστηριότητα 5.**

Να εκτελέστε τα παρακάτω τμήματα κώδικα και να εξηγήσετε τα αποτελέσματα. Ήταν αυτά που αναμένατε; Ποια ερμηνεία δίνετε; Αν διακρίνετε κάποιο πρόβλημα, τι μπορείτε να κάνετε για να το διορθώσετε;

| | |
|---|---|
| `>>> import math` | `>>> myglobal = 496` |
| `>>> def donothing(value):`<br>`  local = value+4`<br>`  print local` | `>>> def foo( value ):`<br>`  myglobal = value + 2`<br>`  print(myglobal)` |
| `>>> donothing( 496 )`<br>`>>> print(local)` | `>>> foo( 8128 )`<br>`>>> print(myglobal)` |

**Απάντηση**

Η value είναι τοπική μεταβλητή και αναγνωρίζεται μόνο μέσα στη συνάρτηση. Οπότε print local στην 6η γραμμή του προγράμματος προκαλεί την εμφάνιση λάθους.
Η myglobal είναι γενική μεταβλητή και η τιμή της μπορεί να αλλάξει μέσα στη συνάρτηση αλλά η αλλαγή αυτή ισχύει μόνο για τη συνάρτηση. Έξω από τη συνάρτηση η μεταβλητή myglobal διατηρεί την αρχική τιμή της.

**Δραστηριότητα 6.**

Να ορίσετε μια συνάρτηση με όνομα count η οποία να δέχεται δύο ορίσματα (sequence και item) και να επιστρέφει πόσες φορές εμφανίζεται το item στη λίστα sequence.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή

**Δραστηριότητα 7.**

Να κάνετε τα παρακάτω βήματα:

1) Να ορίσετε συνάρτηση με όνομα purify η οποία δέχεται μια λίστα αριθμών, απομακρύνει όλους τους περιττούς από τη λίστα και επιστρέφει το αποτέλεσμα. Για παράδειγμα η purify([1,2,3]) επιστρέφει 2.
2) Να αναζητήσετε πληροφορίες για τη λειτουργία των μεθόδων pop και remove των λιστών.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή

**Δραστηριότητα 8.**

Να ορίσετε μια συνάρτηση product, η οποία δέχεται μια λίστα ακεραίων και επιστρέφει το γινόμενο όλων των στοιχείων της λίστας.

Βοήθεια

Για παράδειγμα, η product([4, 5, 5]) επιστρέφει 100.

**Απάντηση**

Αναπτύσσεται στο Τετράδιο Εργασίας Μαθητή

**Δραστηριότητα 9.**

Να γράψετε ένα πρόγραμμα σε Python το οποίο θα διαβάζει μια λίστα αριθμών και θα υπολογίζει το μέσο όρο των θετικών αριθμών. Για την εισαγωγή των αριθμών και τον υπολογισμό του μέσου όρου, να υλοποιήσετε κατάλληλες συναρτήσεις.

**Απάντηση**

Θα χρησιμοποιήσουμε μια λίστα την οποία γνωρίζουμε από το μάθημα της Β Λυκείου.

```
def readList( ) :
    List = [ ]
    number = input("Δώσε έναν αριθμό ή None για τέλος : ")
    while number is not None :
        List.append(number)
        number = input("Δώσε έναν αριθμό ή None για τέλος : ")
    return List

def avgPositive( List ) :
    s = 0.0
    positives = 0
    for number in List:
        if number > 0 :
            s = s + number
            positives = positives + 1

    if positives > 0:
        average = s / positives
    else:
        average = None
    return average

# Το πρόγραμμα
L = readList()
average = avgPositive( L )
print average
```

**Δραστηριότητα 10.**

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει δύο λίστες του ίδιου μεγέθους και θα δημιουργεί μια τρίτη. Σε αυτήν κάθε στοιχείο είναι ο μέσος όρος των αντίστοιχων στοιχείων των δύο λιστών, δηλαδή των στοιχείων που βρίσκονται στις ίδιες θέσεις. Για την εισαγωγή των λιστών και τον υπολογισμό της τρίτης λίστας, να υλοποιήσετε κατάλληλες συναρτήσεις.

```
def readList( ):
    ..........................................
def avgList( A, B ):
    ..........................................
```

**Βοήθεια**

Παρακάτω δίνεται ένα παράδειγμα εκτέλεσης της avgList:

```
>>> A = [ 2, 4, 12, 15 ]
>>> B = [ 4, 8, 15, 18 ]
>>> avgList( A, B )
[ 3.0, 6.0, 13.5, 16.5 ]
```

**Απάντηση**

Αν οι δυο λίστες δεν έχουν το ίδιο μέγεθος επιλέγουμε το μικρότερο μέγεθος έτσι ώστε να ορίζεται ο μέσος όρος για κάθε ζευγάρι.

```
def avgList(A, B):
    size = len(A)
    if len(B) < size:
        size = len(B)

    avg = [ ]
    for i in range(size):
        avg.append( (A[i] + B[i])/2.0 )
    return avg

# Χρησιμοποιούμε την readList από την προηγουμένη
# δραστηριότητα αφού την έχουμε ήδη ορίσει


A = readList()
print
print "Δώσε τη δεύτερη λίστα ", len(A), " αριθμών"
print
B = readList()
print avgList(A,B)
```

# ΚΕΦΑΛΑΙΟ 8 Δομές Δεδομένων ΙΙ

## Λύση Δραστηριοτήτων Βιβλίου Μαθητή

### Δραστηριότητα 1

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα εμφανίζει τα γράμματά της, σε ξεχωριστή γραμμή κάθε γράμμα της.

### Απάντηση

Διασχίζουμε τη λέξη, γράμμα – γράμμα, με την εντολή for … in …

```
# Εμφανίζει κάθε γράμμα της λέξης σε ξεχωριστή γραμμή
def splitLetters( ):
    word = raw_input("Δώσε μια λέξη : ")
    for letter in word:
        print letter
```

### Δραστηριότητα 2

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

### Απάντηση

Ελέγχει αν ένα γράμμα είναι κεφαλαίο με χρήση του τελεστή in διασχίζοντας τη λέξη γράμμα – γράμμα, με την εντολή for … in … , αφού πρώτα κατασκευάσει ένα string/σύνολο με όλα τα κεφαλαία αγγλικά γράμματα.

```
def countCapitals( ):
    enCapSet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    word = raw_input("Δώσε μια λέξη : ")
    countCapitals = 0
    for letter in word:
        if letter in enCapSet :
            countCapitals += 1
    return countCapitals
print countCapitals( )
```

**Δραστηριότητα 3**

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

```
# αρχικά δίνουμε την υλοποίηση μιας στοίβας
# που θα χρησιμοποιήσουμε και σε επόμενες δραστηριότητες
# Ώθηση
def push(stack, item):
    stack.append(item)
# Απώθηση
def pop(stack):
    return stack.pop()
# Έλεγχος άδειας στοίβας
def isEmpty(stack):
    return len(stack)==0

# Δημιουργία νέας στοίβας
def createStack():
    return [ ]
```

```
# Αντιστρέφει μια λέξη με τη χρήση στοίβας
# Το τελευταίο γράμμα της λέξης είναι το τελευταίο που θα
# εισέλθει στη στοίβα και θα βρίσκεται στην κορυφή της. Όταν
# ξεκινήσουμε να απωθούμε γράμματα από τη στοίβα θα βγει
# πρώτα το τελευταίο, μετά το προτελευταίο κ.ο.κ. Άρα, τα
# γράμματα θα εμφανιστούν σε αντίστροφη σειρά από αυτή με την
# οποία μπήκαν στη στοίβα.
```

```
def reverseWord( ):
    word = raw_input("Δώσε μια λέξη : ")
    reverse = ""
    stack = createStack( )
    for letter in word:
        push( stack, letter )

    while not isEmpty(stack):
        reverse = reverse + pop(stack)

    print reverse
```

**Δραστηριότητα 4**

Να γράψετε μια συνάρτηση *isSubstring(string, substring)* η οποία θα ελέγχει αν η συμβολοσειρά substring περιέχεται στη συμβολοσειρά string και αν ναι, θα επιστρέφει True. Στη συνέχεια, να γράψετε συνάρτηση η οποία να υπολογίζει πόσες φορές εμφανίζεται μια συμβολοσειρά μέσα σε μια άλλη.

```
# Αυτό μπορεί να γίνει με χρήση του τελεστή in πολύ απλά
def isSubstring( string, substring):
    return substring in string

# Ελέγχει αν το substring εμφανίζεται στο string ξεκινώντας από τη θέση start
def isPrefix(string, substring, start):
    i = start
    j = 0
    count = 0
while j < len(substring) and i < len(string) :
    if string[ i ] == substring[ j ] :
        count = count + 1
    i = i + 1
    j = j + 1

    return count == len(substring)
# εκτελούμε για κάθε θέση του string την προηγουμένη συνάρτηση
# και έτσι υπολογίζουμε πόσες φορές εμφανίζεται το substring στο String
def findSubstring( string, substring):
    pos = 0
    count = 0

    while pos < len(string) :
        if isPrefix(string, substring, pos)== True :
            count = count + 1
        pos = pos + 1
    return count
```

**Δραστηριότητα 5**

Να γράψετε, σε Python, πρόγραμμα το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο, μέχρι να δοθεί ο αριθμός 0. Κάθε φορά που θα διαβάζει έναν θετικό αριθμό, θα τον προσθέτει σε μια στοίβα. Όταν διαβάζει έναν αρνητικό αριθμό, θα αφαιρεί τόσους αριθμούς από τη στοίβα, όσο είναι η απόλυτη τιμή του αριθμού.

```
# γίνεται χρήση της στοίβας που έχει οριστεί προηγουμένως
def push(stack, item):
    stack.append(item)
def pop(stack):
    return stack.pop()
def isEmpty(stack):
    return len(stack)==0
def createStack():
    return [ ]

def program85( ):
    stack = createStack()
    number = input("δώσε έναν αριθμό : ")
    while number != 0 :
        if number>0 :
            push(stack, number)
        else:
            i = 0
            while i <= number and not isEmpty(stack) :
                print stack.pop() ,
                i = i + 1
            print
        number = input("δώσε έναν αριθμό : ")
program85()
```

**Δραστηριότητα 6**

Να γράψετε, σε Python, πρόγραμμα το οποίο θα δέχεται ως είσοδο ένα κείμενο και θα εμφανίζει πόσες φορές εμφανίζεται κάθε γράμμα του αγγλικού αλφαβήτου σε αυτό. Να χρησιμοποιήσετε αν επιθυμείτε, ένα λεξικό.

```
# 1ος τρόπος : χωρίς λεξικό
# Επιστρέφει τη θέση ενός γράμματος letter στο αλφάβητο
alphabet
def indexStr(letter):
    capSet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    lowSet = "abcdefghijklmnopqrstuvwxyz"
    for index in range(26):
        if letter == capSet[index] or letter == lowSet[index]:
            return index
    return -1;
def program86( ):
    enCapitalSet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    text = raw_input("Δώσε ένα κείμενο με αγγλικά γράμματα : ")
    # δημιουργία και μηδενισμός μιας λίστας μετρητών
    counter = [ ]
    for i in range(26):
        counter.append(0)
    # αυξάνω τον μετρητή στην αντίστοιχη θέση του γράμματος
    for letter in text:
        index = indexStr(letter)
        if index >= 0 :
            counter[ index ] += 1
    print " Letter Frequency"
    print "-------- -----------"
    for i in range(26):
        print " ", enCapitalSet[i]," ", counter[i]
```

```
# 2ος τρόπος : με λεξικό (μόνο για τον εκπαιδευτικό)
def program86_2( ):
    messg = ("Δώσε ένα κείμενο με αγγλικά μικρά γράμματα: ")
    text = raw_input( messg )
    alphabet = dict( )

    for letter in text:
        if letter in alphabet:
            alphabet[ letter ] += 1
        else:
            alphabet[ letter ] = 1
    print " Letter Frequency"
    print "-------- -----------"
    for letter in alphabet:
        print " ", letter," ", alphabet[letter]
    print "*******************************************************"
```

**Δραστηριότητα 7**

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

```
def maxLength( wordList ) :
    maxLen = 0
    maxWord = ""
    for word in wordList:
        if len(word) > maxLen :
            maxLen = len(word)
            maxWord = word
    return maxWord
```

**Δραστηριότητα 8**

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

```
def num_of_Vowels( word ):
    vowels = "AEIOUYaeiouy"
    count = 0
    for letter in word:
        if letter in vowels:
            count += 1
    return count
def maxVowels( wordList ):
    maxV = 0
    maxWord = ""
    for word in wordList:
        if num_of_Vowels(word) > maxV :
            maxV = num_of_Vowels(word)
            maxWord = word
    return maxWord
```

## Λύσεις Δραστηριοτήτων Τετραδίου Εργασιών Μαθητή

### Δραστηριότητα 1.

Ανοίξτε το διερμηνευτή της Python, δώστε τις παρακάτω εντολές και παρατηρήστε τα αποτελέσματα. Στη συνέχεια, περιγράψτε τη λειτουργία των τελεστών +, in, not in και των συναρτήσεων len(), str(), int().

```
>>> w = "MONTY PYTHON"
>>> w[0] + w[1] + w[2] + w[8]
'MONT'
>>> len( w )
12
>>> 123 + "123"
Εμφάνιση Λάθους
>>> str(123) + "123"
'123123'
>>> 123 + int("123")
246
```

```
>>> "PYTHON" in w
True
>>> vowels = "aeiou"
>>> 'e' in vowels
True
>>> 'p' not in vowels
True
>>> for letter in vowels:
        print letter
a
e
i
o
u
```

**Δραστηριότητα 2.**

Να γράψετε μια συνάρτηση η οποία θα ελέγχει αν μια συμβολοσειρά αποτελεί ηλεκτρονική διεύθυνση αλληλογραφίας ελληνικού ιστότοπου, δηλαδή αν περιέχει το σύμβολο '@', αν δεν περιέχει κενά και έχει κατάληξη '.gr'.

**Απάντηση**

```
def isEmail( email ):
    return '@' in email and ' ' not in email \
    and email[-3] + email[-2] + email[-1] == '.gr'

# 2ος τρόπος
def isEmail2( email ):
    N = len( email )
    ending = email[N-3] + email[N-2] + email[N-1]
    return '@' in email and ' ' not in email \
    and ending == '.gr'

# 3ος τρόπος
def isEmail3( email ):
    if '@' not in email:
        return False
    if ' ' in email :
        return False
    if email[-3] + email[-2] + email[-1] != '.gr' :
        return False
    return True
```

**Δραστηριότητα 3.**

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει λέξεις από το πληκτρολόγιο, θα μετράει και θα εμφανίζει πόσες λέξεις ξεκινούν από το γράμμα Α (κεφαλαίο ή μικρό). Όταν δοθεί λέξη που να τελειώνει σε «Ω» ή «ω», τότε το πρόγραμμα θα τερματίζει.

**Απάντηση**

```
count = 0
word = raw_input( 'Δώσε μια λέξη: ' )
while word[-1] != 'ω' and word[-1] != 'Ω' :
    if word[0] == 'A' or word[0] == 'α' :
        count+=1
    word = raw_input('Δώσε μια λέξη: ')
print count
```

**Δραστηριότητα 4.**

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει λέξεις από το πληκτρολόγιο και θα τις ενώνει σε μια μεγάλη πρόταση την οποία στη συνέχεια, θα εμφανίζει στην οθόνη. Οι λέξεις θα χωρίζονται μεταξύ τους με κενά και η πρόταση θα τελειώνει με τελεία '.' .

**Σημείωση:** Η εισαγωγή των λέξεων τελειώνει όταν δοθεί ο χαρακτήρας τελεία " . ".

**Απάντηση**

```
wordList = [ ]
word = raw_input( 'Δώσε μια λέξη: ' )
while word != '.' :
    wordList.append(word)
    word = raw_input('Δώσε μια λέξη: ')
phrase = wordList[0]
for i in range(1,len(wordList)):
    phrase += " " + wordList[i]
phrase += '.'
print phrase
```

**Δραστηριότητα 5**

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μια λέξη και θα ελέγχει αν είναι παλινδρομική (καρκινική), δηλαδή αν διαβάζεται το ίδιο και αντίστροφα, για παράδειγμα radar, madam.

**Απάντηση**

```
word = raw_input('Δώσε μια λέξη: ')
palindrome = True
N = len(word)
i = 0
while i<N/2 and palindrome :
    if word[i] != word[N-i-1] :
        palindrome = False
    i += 1
print palindrome
```

**Δραστηριότητα 6.**

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

**Απάντηση**

```
x = [21, 23, 25, 27]
y = [5, 6, 7, 8]
z = x + y
print z
# Απάντηση: [21, 23, 25, 27, 5, 6, 7, 8]

print z[ 1 ]
# Απάντηση: 23

z[ 0 ] = 45
print z
# Απάντηση: [45, 23, 25, 27, 5, 6, 7, 8]

a = [ x , y ]
print a
print a[ 1 ][ 2 ]
# Απάντηση: [ [45, 23, 25, 27], [5, 6, 7, 8] ]
                7
```

**Δραστηριότητα 7.**

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.

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

**Απάντηση**

θα έχω πολύ ελεύθερο χρόνο την Κυριακή

**Δραστηριότητα 8**

Πειραματιστείτε και δώστε στη συνέχεια την περιγραφή - τεκμηρίωση των παρακάτω συναρτήσεων.

**Απάντηση**

```
def print_grades(grades):       # εκτύπωση βαθμών
    for grade in grades:
        print grade

def grades_sum(grades):         # άθροισμα βαθμών
    total = 0
    for grade in grades:
        total += grade
    return total

def grades_average(grades):    # μέσος όρος βαθμών
    sum_of_grades = grades_sum(grades)
    average = sum_of_grades / float(len(grades))
    return average
```

**Δραστηριότητα 9.**

Ανοίξτε το διερμηνευτή της Python, δώστε τις παρακάτω εντολές και παρατηρήστε τα αποτελέσματα. Να περιγράψετε τη λειτουργία των τελεστών +, in , των μεθόδων pop και append και των συναρτήσεων len και range .

```
>>> w = "MONTY PYTHON"
>>> fib = [1, 1, 2, 3, 5, 8, 13, 21]
>>> fib = fib + [34]
>>> fib = [0] + fib
>>> print fib
>>> last = fib.pop()
>>> print fib, last
>>> fib.append(55)
>>> print fib
>>> 5 in fib
```

```
>>> len( fib )
>>> fib.pop() ; len( fib )
>>> range( 10 )
>>> range( 0, 10 )
>>> len( range( 10 ) )
>>> range( 1, 10 )
>>> range( 1, 10, 2 )
>>> range( 10, 0, –2 )
>>> 100 not in range( 1, 10 )
```

**Απάντηση**

**+ :** συνένωση λιστών
**in :** έλεγχος ύπαρξης ενός στοιχείου σε μια λίστα
**List.pop():** διαγράφει ένα στοιχείο από το τέλος της λίστας το οποίο και επιστρέφει.
**List.append():** εισάγει ένα στοιχείο στο τέλος της λίστας.
**len:** επιστρέφει το πλήθος των στοιχείων (μέγεθος) μιας λίστας.
**range(αρχή, τέλος, βήμα):** επιστρέφει μια λίστα με αριθμούς ανάλογα με τα όρια που έχουν δοθεί (αρχή, τέλος, βήμα)

**Δραστηριότητα 10.**

Να γράψετε ένα πρόγραμμα σε Python το οποίο θα ζητάει από το χρήστη ένα θετικό ακέραιο αριθμό Ν, θα δημιουργεί μια λίστα με όλους τους αριθμούς από το 1 έως και το Ν και στη συνέχεια, θα εμφανίζει το άθροισμα των αριθμών της λίστας.

**Απάντηση**

```
N = input(" N = ")
numberList = range( 1, N+1 )
sum = 0
for number in numberList :
    sum += number
print sum
```

**Δραστηριότητα 11.**

Το παρακάτω πρόγραμμα δημιουργεί μια λίστα με όλα τα θετικά πολλαπλάσια του 3 που είναι μικρότερα του 1000. Να το εκτελέσετε στο IDLE και να διορθώσετε τυχόν λάθη, ώστε να βγάζει το σωστό αποτέλεσμα.

**Απάντηση**

```
list3 = [ ]
for i in range(3,1000, 3):
    list3 = list3 + [ i ]
```

**Δραστηριότητα 12.**

Να γράψετε, σε Python, ένα πρόγραμμα το οποίο θα διαβάζει από το πληκτρολόγιο αριθμούς και θα τους αποθηκεύει σε μια λίστα. Η εισαγωγή των αριθμών θα σταματάει όταν δοθεί ένας αρνητικός αριθμός. Στη συνέχεια το πρόγραμμα θα εμφανίζει:
1) Πόσοι αριθμοί δόθηκαν.
2) Τους αριθμούς σε αντίστροφη σειρά από αυτή που δόθηκαν.

**Απάντηση**

```
mylist = [ ]
number = input('Δώσε μια λέξη: ')
while number >= 0 :
    mylist = [ number ] + mylist
    number = input('Δώσε μια λέξη: ')
print len(mylist)
for number in mylist:
    print number, " ",
```

**Δραστηριότητα 13.**

Να γράψετε, σε Python, ένα πρόγραμμα το οποίο να διαβάζει από το πληκτρολόγιο 12 ακέραιους αριθμούς και τους αποθηκεύει σε μια λίστα. Στη συνέχεια να υπολογίζει και να εμφανίζει:
1) Το μέσο όρο των 12 αριθμών.
2) Το μεγαλύτερο αριθμό από τους 12.
3) Το πλήθος των θετικών αριθμών.

**Απάντηση**

```
# -*- coding: utf-8 -*-

number_list = []

for i in range(12):
    number = input("δώσε έναν αριθμό = ")
   number_list.append(number)
```

```
# Χρησιμοποιούμε έναν αθροιστή s στον οποίο
# προσθέτουμε κάθε φορά τον επόμενο αριθμό στη
# λίστα.
s = 0
for number in number_list:
    s = s + number
average = s / len(number_list)
print "μέσος όρος = ", average

maximum = number_list[0] #αρχικοποίηση με το 1ο στοιχείο
for i in range(1,len(number_list)):
    if number_list[i] > maximum :
        maximum = number_list[i]
print "μέγιστο = ", maximum

# Κάθε φορά που βρίσκουμε έναν θετικό αυξάνουμε
# κατά 1 το μετρητή positives.
positives = 0
for number in number_list:
    if number > 0 :
        positives = positives + 1
print "θετικοι = ", positives
```

Μπορούμε να διασχίσουμε μια λίστα List με δυο τρόπους:

```
for index in range( len( List ) ):
    print List[ index ]
```

```
for item in List :
    print item
```

**Δραστηριότητα 14.**

Να γράψετε, σε Python, 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) Τη μέγιστη τιμή που έχει στοιχείο της λίστας.
3) Πόσες φορές εμφανίζεται αυτή η μέγιστη τιμή.

**Απάντηση**

```
# -*- coding: utf-8 -*-
number_list = [ ]
Sum = 0
while Sum <= 1000:
    number = input("δώσε έναν αριθμό = ")
    number_list.append(number)
    Sum = Sum + number

print "Το άθροισμα είναι :", Sum
maximum = number_list[0]
for item in number_list:
    if item > maximum :
        maximum = item
print "μέγιστο = ", maximum
# Κάθε φορά που βρίσκουμε έναν αριθμό ίσο με τη μέγιστη
# τιμή αυξάνουμε κατά 1 το μετρητή counter.
counter = 0
for item in number_list:
    if item == maximum :
        counter = counter + 1
print " Η μέγιστη τιμή εμφανίζεται:", counter, "φορές"
```

**Δραστηριότητα 15.**

Να γράψετε, σε Python, 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.

**Απάντηση**

```
mylist = [ ]
number = input('Δώσε μια λέξη: ')
while number != 0 :
    mylist.append( number )
number = input('Δώσε μια λέξη: ')
N = len(mylist)
for i in range(N–1, –1, –1) :
    print mylist[ i ], " ",
```

**Δραστηριότητα 16.**

Να αναπτύξετε, σε Python, μια συνάρτηση reverseList( L ), η οποία επιστρέφει τη δοσμένη λίστα L αντεστραμμένη, όπως φαίνεται στο παρακάτω παράδειγμα

```
>>> reverseList( [ 1, 1, 2, 3 ,5, 8, 13 ,21 ] )
[ 21, 13, 8, 5 ,3, 2, 1 ,1 ]
```

**Απάντηση**

```
# 1ος τρόπος
def reverseList( L ):
    reversed = [ ]
    for item in L :
        reversed = [item] + reversed
    return reversed

# 2ος τρόπος
def reverseList2( L ):
    reversed = [ ]
    for i in range(len(L)-1, -1, -1) :
        reversed.append(L[i])
    return reversed
```

**Δραστηριότητα 17.**

Να γράψετε, σε Python, ένα πρόγραμμα το οποίο, δεδομένης μια λίστας ακέραιων αριθμών, θα διαχωρίζει τους αριθμούς σε δύο νέες λίστες, μία για τους θετικούς και μία για τους αρνητικούς.

**Απάντηση**

```
def splitbySign(L):
    positives = [ ]
    negatives = [ ]
    for number in L:
        if number > 0 :
            positives.append(number)
        elif number < 0 :
            negatives.append(number)
    print positives
    print negatives

# 2ος τρόπος
def splitbySign2(L):
    positives = [ ]
    negatives = [ ]
    for number in L:
        if number > 0 :
            positives = positives + [number]
        elif number < 0 :
            negatives = negatives + [number]
    print positives
    print negatives
```

**Δραστηριότητα 18.**

Στα επόμενα παραδείγματα προσπαθήστε πρώτα να μαντέψετε τι θα εμφανιστεί μετά την print() και έπειτα εκτελέστε τον κώδικα στο προγραμματιστικό περιβάλλον IDLE Python.

```
alist = ['a','b','c','d']
ch = ''
for i in alist:
ch += i
print ch
# Τυπώνει τον αριθμό των στοιχείων της λίστας

list1 = ['a','b','c']
i = 0
s1=' '
for ch in list1:
    i +=2
    s1 = s1+i*ch
print s1
# Τυπώνει μια συμβολοσειρά που περιέχει δύο φορές το πρώτο
# στοιχείο της λίστας, 4 φορές το δεύτερο και 6 φορές το τρίτο
#στοιχείο της λίστας
```

**Δραστηριότητα 19.** Στοίβα - Ουρά

Να υλοποιήσετε τις παρακάτω λειτουργίες της δομής δεδομένων "Στοίβα" συμπληρώνοντας κατάλληλα τις συναρτήσεις. Θυμίζουμε ότι στη στοίβα η *ώθηση* και η *απώθηση* γίνονται από το ένα άκρο της λίστας. Μπορείτε να χρησιμοποιήσετε τις μεθόδους append, pop της λίστας και τη συνάρτηση len.

**Απάντηση**

```
def push(stack, item) :
    stack.append( item )

def pop(stack) :
    return stack.pop()
```

```
def isEmpty(stack) :
    return len(stack) == 0

def createStack( ) :
    return [ ]
```

**Δραστηριότητα 21.**

Να γράψετε, σε Python, ένα πρόγραμμα, το οποίο θα διαβάζει αριθμούς μέχρι να δοθεί το 0 και στη συνέχεια θα εμφανίζει τους αριθμούς στην αντίστροφη σειρά από αυτή που τους διάβασε, με τη χρήση στοίβας.

**Απάντηση**

```
number = input(" Δώσε έναν αριθμό ")
stack = createStack()
while number != 0 :
    push(stack, number)
    number = input(" Δώσε τον επόμενο αριθμό ")
while not isEmpty( stack ) :
    print pop( stack )
```

**Δραστηριότητα 22.**

Να δώσετε την υλοποίηση της δομής δεδομένων ουρά.

```
def enqueue(queue, item) :
    queue.append( item )

def dequeue(queue) :
    return queue.pop(0)
```

```
def isEmpty(queue) :
    return len(queue) == 0

def createQueue( ) :
    return [ ]
```

Στη συνέχεια να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς μέχρι να δοθεί η τιμή None και θα τους αποθηκεύει σε μια ουρά. Μετά, θα τους εξάγει από την ουρά και θα τους αποθηκεύει σε μια στοίβα. Αφού τοποθετηθούν όλοι στη στοίβα, θα τους απωθεί και θα τους εμφανίζει έναν – έναν στην οθόνη. Τι παρατηρείτε; Τι καταφέρατε με τον παραπάνω αλγόριθμο;

**Απάντηση**

```
number = input(" Δώσε έναν αριθμό ")
queue = createQueue()
while number is not None :
    enqueue(queue, number)
    number = input(" Δώσε τον επόμενο αριθμό ")
stack = createStack()
while not isEmpty(queue) :
    push( stack, dequeue( queue ) )
while not isEmpty( stack ) :
    print pop( stack )
```

## ΚΕΦΑΛΑΙΟ 9 Εφαρμογές σε γλώσσα προγραμματισμού με χρήση API

Το κεφάλαιο αυτό είναι εκτός διδακτέας - εξεταστέας ύλης.

## ΚΕΦΑΛΑΙΟ 10 Βάσεις δεδομένων

Το κεφάλαιο αυτό είναι εκτός διδακτέας - εξεταστέας ύλης.

# ΚΕΦΑΛΑΙΟ 11 Αντικειμενοστρεφής Προγραμματισμός

## Λύσεις Δραστηριοτήτων Βιβλίου Μαθητή

**Δραστηριότητα εμπέδωσης** παραγράφου 11.2

Δίνεται η παρακάτω κλάση:

```
class Car:
        def __init__ (self, make):
            self.make=make
            self.speed = 60
        def speed_up(self,speed):
            self.speed = speed
            print "I am driving at a speed", self.speed, "km/h"
        def turn(self):
            print "I am turning … "
```

1. Ποιος είναι ο κατασκευαστής (constructor) της κλάσης;
2. Να καταγράψετε τις ιδιότητες της κλάσης και τις μεθόδους της, καθώς και τις ενέργειες που πραγματοποιούν.
3. Να προσθέσετε τις ιδιότητες color και year που θα αντιπροσωπεύουν αντίστοιχα το χρώμα και το έτος κυκλοφορίας του αυτοκινήτου και θα αρχικοποιούνται στον κατασκευαστή.
4. Να αλλάξετε τη μέθοδο turn, έτσι ώστε να δέχεται ως παράμετρο μια συμβολοσειρά που ορίζει αν το αυτοκίνητο θα στρίψει αριστερά ή δεξιά.
5. Να δημιουργήσετε τα παρακάτω στιγμιότυπα της κλάσης:
   I. Αντικείμενο με όνομα convertible και μάρκα "bmw", χρώμα "μαύρο" και έτος κυκλοφορίας "2013".
   II. Αντικείμενο με όνομα sedan και μάρκα "toyota", χρώμα "κόκκινο" και έτος κυκλοφορίας "2009".
6. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο convertible να στρίψει δεξιά.
7. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο sedan να τρέχει με 90 χιλ/ώρα.

**Απαντήσεις**

1. Ο κατασκευαστής (constructor) της κλάσης είναι η μέθοδος __init____ (self, make), ο οποίος δημιουργεί το αντικείμενο και αρχικοποιεί τις ιδιότητές του.

2. Ιδιότητες της κλάσης **Car**: make, speed
   Μέθοδοι: 1) speed_up(self,speed): δίνει τιμή στην ιδιότητα speed του αντικειμένου και εμφανίζει στην οθόνη το μήνυμα «I am driving at a speed» και στη συνέχεια, εμφανίζει την τιμή της ιδιότητας του αντικειμένου ακολουθούμενη από το μήνυμα «km/h»
   2) turn(self): εμφανίζει στην οθόνη το μήνυμα «I am turning …»

3. Ο κατασκευαστής της κλάσης αλλάζει ως εξής:

```
def __init____ (self, make, color,year):
        self.make=make
        self.speed = 60
        self.color = color
        self.year = year
```

4.
```
def turn(self, direction):
        print "I am turning … ", direction
```

5. I. convertible = Car("bmw","black",2013)
   II. sedan = Car("Toyota","red",2009)

6. convertible.turn("right")

7. sedan.speed_up(90)

**Δραστηριότητα εμπέδωσης παραγράφου 11.4**

- Ποιες είναι οι ιδιότητες της κλάσης Student;
- Δημιουργήστε στιγμιότυπο της κλάσης Student με όνομα "Γιάννης" και τάξη "Β".
- Πώς θα εμφανίσετε τις ιδιότητες του στιγμιότυπου που δημιουργήσατε στο προηγούμενο ερώτημα;
- Να ορίσετε υποκλάση Teacher της κλάσης Person η οποία, εκτός από το όνομα, θα έχει επιπλέον τις ιδιότητες κλάδος (field) και χρόνια υπηρεσίας (years).
- Δημιουργήστε τα παρακάτω στιγμιότυπα της κλάσης Teacher:
  - Στιγμιότυπο με όνομα "Αναστασίου", κλάδο "ΠΕ02" και χρόνια υπηρεσίας "11".
  - Στιγμιότυπο με όνομα "Παπαχρήστου", κλάδο "ΠΕ03" και χρόνια υπηρεσίας "16".

**Απάντηση**

- Ιδιότητες της κλάσης Student: _name (που την κληρονομεί από την κλάση Person) και classatt
- student1 = Student("Γιάννης","Β")
- print student1._name
  print student1.classatt
- class Teacher (Person):
  ```
      def __init__ (self, name,field,years):
          super (Teacher, self). _init_ (name)
          self.field = field
          self.years = years
  ```
- teacher1= Teacher("Αναστασίου ","ΠΕ02",11)
  teacher2 = Teacher("Παπαχρήστου ", "ΠΕ03", 16)

**Δραστηριότητες** (Παραγράφου 11.5)

**Δραστηριότητα 1**

Να ορίσετε κλάση με όνομα Akeraios η οποία θα έχει την ιδιότητα timi και τις παρακάτω μεθόδους:

I. Anathese_timi(timi), η οποία θα αναθέτει τιμή στην ιδιότητα του αντικειμένου.
II. Emfanise_timi(), η οποία θα εμφανίζει την τιμή της ιδιότητας του αντικειμένου.

Στη συνέχεια, να δημιουργήσετε στιγμιότυπο της κλάσης Akeraios με όνομα Artios και να χρησιμοποιήσετε τις παραπάνω μεθόδους για να δώσετε την τιμή 14 στην ιδιότητα του αντικειμένου και να την εμφανίσετε.

**Ενδεικτική λύση**

```
class Akeraios :
        def __init__ (self):
                self.timi = 0
        def Anethese_timi(self, timi):
                self.timi = timi
        def Emfanise_timi(self):
                print self.timi

Artios = Akeraios()

Artios. Anethese_timi(14)
Artios. Emfanise_timi()
```

**Δραστηριότητα 2**

Να ορίσετε κλάση με όνομα Student η οποία θα έχει τρεις ιδιότητες: μία για τον αριθμό μητρώου, μία για το ονοματεπώνυμο και μία τους βαθμούς σε 8 μαθήματα (πίνακας).

Επίσης, να ορίσετε τις παρακάτω μεθόδους της κλάσης:

- Μια μέθοδο για την ανάθεση τιμών στις ιδιότητες ενός αντικειμένου.
- Μια μέθοδο για την εμφάνιση των τιμών των ιδιοτήτων ενός αντικειμένου.
- Μια μέθοδο για τον υπολογισμό και την επιστροφή του μέσου όρου βαθμολογίας στα 8 μαθήματα.

Στη συνέχεια, αφού ορίσετε ένα στιγμιότυπο της κλάσης Student με όνομα Chris, να χρησιμοποιήσετε τις παραπάνω μεθόδους για να δώσετε τιμή στις ιδιότητες του αντικειμένου, να εμφανίσετε τις ιδιότητες του αντικειμένου και να υπολογίσετε το μέσο όρο βαθμολογίας του.

**Ενδεικτική λύση**

```
class Student:
        def __init__ (self, AM, name,grades):
                self.AM=AM
                self.name = name
                self.grades = grades[:8]
        def Emfanise_times(self):
                print "AM: ", self.AM
                print "Name: ", self.name
                print "grades: ", self.grades
            def Ypol_MO(self):
                sum=0
                for i in range(len(self.grades)):
                    sum = sum + self.grades[i]
                return sum/len(self.grades)

    chris = Student( 209,"Chris",[14,16,18,19,12,20,17,20])
    chris. Emfanise_times()
    print chris.Ypol_MO()
```

## Λύσεις Δραστηριοτήτων Τετραδίου Εργασιών Μαθητή

### Δραστηριότητα 1

Να υλοποιήσετε τη δομή δεδομένων "Στοίβα" ως μια κλάση, αξιοποιώντας τις τεχνικές του αντικειμενοστρεφούς προγραμματισμού.

### Απάντηση

```
class Stack :
    def __init__ (self ) :
        self.items = [ ]
    def push(self, item) :
        self.items.append( item )
    def pop(self) :
        return self.items.pop()
def isEmpty(self) :
        return self.items == [ ]
```

Να υλοποιήσετε την εφαρμογή *Αντιστροφή αριθμών* -που περιγράφεται στην παράγραφο 8.3 του βιβλίου Μαθητή- χρησιμοποιώντας την αντικειμενοστρεφή έκδοση της στοίβας που υλοποιήσατε. Ποιες διαφορές παρατηρείτε ανάμεσα στις δυο υλοποιήσεις; Ποια προτιμάτε εσείς και γιατί; Ποια μέθοδος της κλάσης Stack έχει υποκαταστήσει τη συνάρτηση createStack;

```
stack = Stack()
number = int( raw_input("number = ") )
while number != 0 :
    stack.push( number )
    number = int( raw_input("number = ") )

while not stack.isEmpty() :
    number = stack.pop()
    print number
```

Η βασική διαφορά έγκειται στο γεγονός ότι το αντικείμενο στοίβα δεν είναι παράμετρος στην κλήση αλλά χρησιμοποιείται ο συμβολισμός με την τελεία. Π.χ. αντί για pop( stack ) γράφουμε stack.pop() .
Στη θέση της createStack υπάρχει πλέον ο κατασκευαστής της στοίβας με το όνομά της (Stack).

**Δραστηριότητα 2.**

Να ορίσετε την κλάση Student η οποία περιέχει τα παρακάτω δεδομένα:
- το όνομα του μαθητή (name)
- μια λίστα με τους βαθμούς του μαθητή (grades)
- μια μεταβλητή κλάσης η οποία μετράει πόσα αντικείμενα υπάρχουν κάθε στιγμή (students_count) και τις παρακάτω μεθόδους:
- κατασκευαστής (constructor) αντικειμένων της κλάσης μαθητής ( _init_ )
- εμφανίζει μήνυμα "καλημέρα" και το όνομα του μαθητή
- υπολογίζει και επιστρέφει το μέσο όρο των βαθμών του μαθητή (average)
- εμφανίζει το περιεχόμενο της μεταβλητής *πλήθος_μαθητών*, δηλαδή είναι μέθοδος κλάσης και μπορεί να κληθεί και με το όνομα της κλάσης.

Στη συνέχεια να γράψετε ένα πρόγραμμα το οποίο:
1) Δημιουργεί 5 αντικείμενα student με στοιχεία συμμαθητών σας.
2) Αποθηκεύει τα αντικείμενα αυτά σε μια λίστα.
3) Διατρέχει τη λίστα και εμφανίζει τα ονόματα όσων έχουν μέσο όρο πάνω από 18.
4) Εμφανίζει το όνομα του μαθητή με το μεγαλύτερο μέσο όρο.

**Απάντηση**

```
class Student :

    students_count = 0

    def __init__ ( self, name, grades ):
        self.name = name
        self.grades = [] + grades # θέλουμε αντίγραφο της λίστας
        Student.students_count += 1

    def printInfo( self ):
        print "Καλημέρα ", self.name
```

```
    def average( self ):
        s = 0.0
        for grade in self.grades :
            s += grade
        return s / len(self.grades)

    @classmethod
    def print_students():
        print Student.students_count

students = []
students.append( Student("Ευγενία", [18,17,18,20]) )
students.append( Student("Ναταλία", [13,15,16,13]) )
students.append( Student("Αλέξανδρος", [18,17,18,20]) )
students.append( Student("Δημήτρης", [9,12,13,11]) )
students.append( Student("Δήμητρα", [16,17,19,20]) )

max_grade = 0.0
for student in students:
    grade = student.average()
    if grade > 18 :
        print student.name
    if grade > max_grade :
        max_grade = grade
        max_name = student.name
print
print "Ο μαθητής με το μεγαλύτερο βαθμό είναι ο/η ",
max_name

class Stack :
        return self.items.pop()

def isEmpty( self ):
        return self.items == [ ]
```

**Δραστηριότητα 3.**

Να σχεδιάσετε και να υλοποιήσετε μια ιεραρχία κλάσεων, όπου οι κλάσεις Τετράγωνο (Square), Κύκλος (Circle) και Ορθογώνιο (Rectangle) κληρονομούν από την κλάση Σχήμα (Schema). Να ορίσετε όλα τα δεδομένα και τις μεθόδους που θεωρείτε απαραίτητα για κάθε κλάση. Κάθε κλάση πρέπει να έχει και μια μέθοδο Εμβαδό (getArea) που θα υπολογίζει και θα επιστρέφει το εμβαδόν του αντίστοιχου σχήματος.

**Απάντηση**

```
import math
class Schema :

    def area( self ):
        pass

class Square( Schema ) :


    def __init__ (self, length):
        self.side = length


    def getArea( self ):
        return self.side * self.side

class Circle( Schema ) :

    def __init__ (self, length):
        self.radius = length

    def getArea( self ):
        return math.pi * self.radius * self.radius

class Rectangle( Schema ) :

    def __init__ (self, xsize, ysize):
        self.length = xsize
        self.width = ysize
    def getArea( self ):
        return self.length * self.width
```

**Δραστηριότητα 4.**

Δίνεται το παρακάτω πρόγραμμα σε Python:

```
class Teacher:

    count = 0

    def __init__ (self, name, subject):
        self.name = name
        self.subject = subject
        Teacher.count += 1

    def displayCount(self):
        print "All Teachers ", Teacher.count

    def displayTeacher(self):
        print "Name : ", self.name, ", Subject: ", self.subject

godel = Teacher("Turing", "Logic")
dijkstra = Employee("Dijkstra", "Computer Science")
godel.displayTeacher()
dijkstra.displayTeacher( )
Teacher.displayCount( )
```

Να καταγράψετε:
1) τον κατασκευαστή της κλάσης
2) τις ιδιότητες της κλάσης
3) τις μεθόδους της κλάσης
4) τα στιγμιότυπα κάθε κλάσης.

**Απάντηση**

1) Το ρόλο του κατασκευαστή της κλάσης τον παίζει η _init_
2) Οι ιδιότητες της κλάσης είναι subject, name για κάθε αντικείμενο και count για όλη την κλάση.

3) Οι μέθοδοι της κλάσης είναι οι displayCount, displayTeacher.
4) Τα στιγμιότυπα της κλάσης Teacher είναι το godel και της κλάσης Employee το dijkstra.

Να σημειωθεί ότι το γεγονός πως η κλάση Employee δεν ορίζεται σε αυτό το αρχείο, δεν επηρεάζει καθόλου την απάντηση των ερωτημάτων.

**Δραστηριότητα 5.**

Δίνεται το παρακάτω πρόγραμμα:

```
class Customer:
    #Ένας πελάτης με καταθετικό λογαριασμό σε μια τράπεζα.
    # Οι πελάτες έχουν τις παρακάτω ιδιότητες:
    # name: όνομα του πελάτη
    # balance: το υπόλοιπο του λογαριασμού του πελάτη.

    def __init__ ( self, name, balance = 0.0 ):
        # επιστρέφει ένα αντικείμενο της κλάσης Customer
        # με όνομά name και αρχικό υπόλοιπο 0.0
        self.name = name
        self.balance = balance

    def withdraw(self, amount):
        # Επιστρέφει το νέο υπόλοιπο του λογαριασμού μετά την
        # ανάληψη του ποσού *amount*

        self.balance -= amount
        return self.balance

    def deposit(self, amount):
        # Επιστρέφει το νέο υπόλοιπο του λογαριασμού μετά την
        # κατάθεση του ποσού *amount*
        self.balance += amount
        return self.balance
```

Να γράψετε εντολές που να κάνουν τα εξής:

1) Να δημιουργούν δύο αντικείμενα της κλάσης Customer, John και Mary.
2) Να γίνεται κατάθεση του ποσού 1400 ευρώ στο λογαριασμό του John.
3) Να γίνεται ανάληψη 300 ευρώ από το λογαριασμό του John.
4) Να γίνεται ανάληψη 400 ευρώ από το λογαριασμό της Mary.
5) Εκτελέστε το ολοκληρωμένο πρόγραμμα στο περιβάλλον της Python και μελετήστε τα αποτελέσματα.

**Απάντηση**

```
John = Customer("John")
Mary = Customer("Mary")
John.deposit(1400)
John.withdraw(300)
Mary.withdraw(400)
print John.balance
print Mary.balance
```

# ΚΕΦΑΛΑΙΟ 12 Εισαγωγή στην Υπολογιστική Σκέψη

Το κεφάλαιο αυτό είναι εκτός διδακτέας - εξεταστέας ύλης

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: 0-24-0603
ISBN 978-960-06-5890-3