ΚΕΦΑΛΑΙΟ 3

1. Τι είναι τα δεδομένα και που βρίσκονται. (σελ 29)

ΑΠ: Δεδομένα είναι ένα μη αξιολογημένο σύνολο διακριτών στοιχείων, μιας παρατήρησης - αναφοράς, μια συλλογή που αποτυπώνει «τιμές» επί αντικειμένων, προσώπων, γεγονότων κλπ. Τα δεδομένα μπορεί να είναι αποθηκευμένα στην κύρια μνήμη του υπολογιστή, σε αποθηκευτικό μέσο ή η εισαγωγή τους να γίνεται από κάποια μονάδα.

2. Τι ονομάζουμε «τύπο δεδομένων». (σελ 29)

ΑΠ: Οι τύποι δεδομένων προσδιορίζουν τον τρόπο παράστασης των δεδομένων εσωτερικά στον υπολογιστή, καθώς και το είδος της επεξεργασίας τους από αυτόν.

3. Ποιοι τύποι δεδομένων υπάρχουν στην Python. (σελ 29)

ΑΠ: Οι τύπο δεδομένων είναι ο αριθμητικός, ο λογικός και οι συμβολοσειρές ή αλφαριθμητικά.

4. Τι περιλαμβάνει ο αριθμητικός τύπος δεδομένων. Τι τιμές μπορεί να λάβει. (σελ 29)

ΑΠ: Οι αριθμοί είναι 3 τύπων. Είναι οι ακέραιοι, οι αριθμοί κινητής υποδιαστολής και οι μιγαδικοί αριθμοί.

5. Τι τιμές μπορεί να λάβει ο λογικός τύπος δεδομένων. Γιατί χρησιμοποιείται. (σελ 30)

ΑΠ: Ο λογικός τύπος δεδομένων μπορεί να λάβει 2 τιμές. Την True και την False. Σκοπό έχει την καταγραφή του αποτελέσματος ενός ελέγχου.

6. Τι είναι οι συμβολοσειρές (αλφαριθμητικά). (σελ 30 και σελ 127)

ΑΠ: Οι συμβολοσειρές (αλφαριθμητικά) είναι μια ακολουθία από χαρακτήρες που έχουν σταθερό μέγεθος και μη μεταβαλλόμενα περιεχόμενα. Δηλαδή, δεν μπορούμε να προσθέσουμε ή να αφαιρέσουμε χαρακτήρες, ούτε να τροποποιήσουμε τα περιεχόμενα του αλφαριθμητικού. Γι' αυτό λέμε ότι η δομή αυτή ανήκει στις μη μεταβαλλόμενες δομές της Python. Η αρίθμηση των χαρακτήρων σε ένα αλφαριθμητικό ξεκινάει από το 0.

7. Με ποια εντολή μπορούμε να ελέγξουμε τον τύπο δεδομένων. Να γράψετε τη

σύνταξη και τη λειτουργία της. (σελ 30 και φυλλάδιο)

ΑΠ: Η συνάρτηση type χρησιμοποιείται για να εμφανίσουμε στην οθόνη τον τύπο μιας μεταβλητής (ενός δεδομένου). Χρησιμοποιείται γράφοντας την εντολή type και δίπλα το δεδομένο μέσα σε παρενθέσεις, και όταν εκτελείται η εντολή, επιστρέφεται ο τύπος του.

παραδείγματα

>>>text = "Hello World." #Η μεταβλητή text θα πάρει ως τιμή το κείμενο: Hello World

>>>type(text) #Η python αναγνωρίζει ότι η μεταβλητή text είναι τύπου str και θα μας το

εμφανίσει στην οθόνη.

>>>x = 35.2 #Η μεταβλητή x θα πάρει ως τιμή το 35,2

>>>type(x) #Η python αναγνωρίζει ότι η μεταβλητή x είναι τύπου float και θα μας το εμφανίσει στην οθόνη.

8. Τι είναι οι τελεστές. Ποιες κατηγορίες τελεστών υπάρχουν. (σελ 30-32)

ΑΠ: Οι τελεστές είναι σύμβολα ή λέξεις για τη δημιουργία αριθμητικών και λογικών εκφράσεων. Υπάρχουν οι αριθμητικοί, οι σχεσιακοί (συγκριτικοί) και οι λογικοί.

9. Ποιοι είναι οι αριθμητικοί τελεστές. (σελ 32)

ΑΠ: Είναι σύμβολα που χρησιμοποιούμε για να κάνουμε μαθηματικές πράξεις. Αυτοί είναι πρόσθεση (+), αφαίρεση (-), πολλαπλασιασμός (*), ακέραια διαίρεση (/ ή //), ύψωση σε δύναμη (**) και υπόλοιπο ακέραιας διαίρεσης ή mod (%).

10. Ποια είναι η προτεραιότητα (ιεραρχία) μεταξύ αριθμητικών τελεστών. (σελ 31)

ΑΠ: Η προτεραιότητα μεταξύ των αριθμητικών τελεστών έχει ως εξής: *ύψωση σε δύναμη>πολλαπλασιασμός, ακέραια διαίρεση, υπόλοιπο ακέραιας διαίρεσης>πρόσθεση, αφαίρεση*. Εάν είναι ίδια προτεραιότητα στους αριθμητικούς τελεστές, ξεκινάμε τις πράξεις από αριστερά προς τα δεξιά.

11. Πώς μπορούμε να αλλάξουμε την ιεραρχία των τελεστών. (σελ 31)

ΑΠ: Για να αλλάξουμε την ιεραρχία, χρησιμοποιούμε παρενθέσεις.

12. Τι είναι οι σχεσιακοί τελεστές. Ποιοι σχεσιακοί τελεστές υπάρχουν στην Python. (σελ 31)

ΑΠ: Οι σχεσιακοί τελεστές χρησιμοποιούνται για την σύγκριση δύο τιμών ή μεταβλητών, με αποτέλεσμα μιας σύγκρισης να είναι True ή False. Οι σχεσιακοί τελεστές που χρησιμοποιούμε είναι οι: Μικρότερο (<), Μικρότερο ή ίσο από (<=), Μεγαλύτερο (>), Μεγαλύτερο ή ίσο από (>=), Ίσο με (==) και Διάφορο από (!=).

13. Πως γίνεται η εφαρμογή των σχεσιακών τελεστών σε συμβολοσειρές. (Φυλλάδιο και σελ 128)

ΑΠ: Για τη σύγκριση μεταξύ συμβολοσειρών η λειτουργία των τελεστών στηρίζεται στη λεξικογραφική διάταξη των χαρακτήρων, με την εξής σειρά:

1. Αριθμοί (ως συμβολοσειρά)

2. Κεφαλαίοι λατινικοί χαρακτήρες

3. Μικροί λατινικοί χαρακτήρες

4. Κεφαλαίοι ελληνικοί χαρακτήρες

5. Μικροί ελληνικοί χαρακτήρες

Ισχύει δηλαδή:

‘0’ < ’1’ <…< ’9’ < ’A’ < ’B’ < ’C <…< ’Z’ < ’a’ < ’b’ <…< ’z’ < ’Α’ <’ Β’ < ’Γ’ <…< ’Ω’ < ’α’ < ’β’ < …<’ω’

Η σύγκριση γίνεται συγκρίνοντας το πρώτο γράμμα μιας λέξης με το πρώτο γράμμα της άλλης. Αν είναι διαφορετικά, τότε το αποτέλεσμα της σύγκρισης θα είναι True ή False σύμφωνα με την παραπάνω διάταξη χαρακτήρων. Αν τα δυο γράμματα είναι ίσα τότε συγκρίνονται τα δεύτερα, τα τρίτα κοκ μέχρι να βρεθούν διαφορετικά γράμματα.

14. Ποιοι είναι οι λογικοί τελεστές και ποια πράξη μπορούμε να κάνουμε με τον καθένα. (σελ 32)

ΑΠ: Οι λογικοί τελεστές είναι ο not (ΟΧΙ) και κάνουμε πράξη άρνησης, and (ΚΑΙ) και κάνουμε πράξη σύζευξης, or (Ή) και κάνουμε πράξη διάζευξης.

15. Ποιο μπορεί να είναι το αποτέλεσμα μιας λογικής πράξης. (σελ 32)

ΑΠ: Το αποτέλεσμα μιας λογικής πράξης είναι True ή False.

16. Ποια είναι η προτεραιότητα μεταξύ λογικών τελεστών. (σελ 32)

ΑΠ: Η προτεραιότητα μεταξύ των λογικών πράξεων είναι *not>and>or*

17. Να περιγράψετε τη λειτουργία του λογικού τελεστή and. (Φυλλάδιο)

ΑΠ: Ο λογικός τελεστής AND εκτελεί την πράξη της σύζευξης 2 πράξεων. Δέχεται 2 λογικές εκφράσεις και παράγει αποτέλεσμα ΠΑΝΤΑ FALSE, όταν τουλάχιστον μια από τις 2 εκφράσεις είναι FALSE.

ή

Ο λογικός τελεστής AND εκτελεί την πράξη της σύζευξης 2 πράξεων. Δέχεται 2 λογικές εκφράσεις και παράγει αποτέλεσμα TRUE, ΜΟΝΟ όταν και οι 2 εκφράσεις είναι TRUE.

18. Να περιγράψετε τη λειτουργία του λογικού τελεστή or. (Φυλλάδιο)

ΑΠ: Ο λογικός τελεστής OR εκτελεί την πράξη της διάζευξης 2 πράξεων. Δέχεται 2 λογικές εκφράσεις και παράγει αποτέλεσμα ΠΑΝΤΑ TRUE, όταν τουλάχιστον μια από τις 2 εκφράσεις είναι TRUE.

ή

Ο λογικός τελεστής OR εκτελεί την πράξη της διάζευξης 2 πράξεων. Δέχεται 2 λογικές εκφράσεις και παράγει αποτέλεσμα FALSE, ΜΟΝΟ όταν και οι 2 εκφράσεις είναι FALSE.

19. Να περιγράψετε τη λειτουργία του λογικού τελεστή not. (Φυλλάδιο)

ΑΠ: Ο λογικός τελεστής NOT εκτελεί την πράξη της άρνησης 1 πράξης. Δέχεται μια λογική έκφραση και παράγει αποτέλεσμα ΠΑΝΤΑ την αντίθετη τιμή της έκφρασης.

20. Ποια είναι η προτεραιότητα μεταξύ όλων των πράξεων/τελεστών στην Python. (Φυλλάδιο)

ΑΠ:

21. Τι είναι εκφράσεις. (Φυλλάδιο)

ΑΠ: Είναι «προτάσεις» που σχηματίζονται από σταθερές, μεταβλητές, συναρτήσεις, τελεστές και παρενθέσεις. Κάθε έκφραση έχει ένα αποτέλεσμα που είναι η τιμή της. Η τιμή μιας έκφρασης προκύπτει όταν αντικαταστήσουμε τις μεταβλητές με τις τιμές τους και εκτελέσουμε τις πράξεις σύμφωνα με την προτεραιότητα των τελεστών.

22. Ποιες κατηγορίες εκφράσεων υπάρχουν. (Φυλλάδιο)

ΑΠ: Υπάρχουν 3 κατηγορίες εκφράσεων:

Αριθμητικές εκφράσεις: Υπολογίζουν αριθμητικές παραστάσεις με τη χρήση αριθμητικών τελεστών και παράγουν μια αριθμητική τιμή.

Λογικές εκφράσεις: Απεικονίζουν απλές παραστάσεις με τη χρήση σχεσιακών τελεστών που το αποτέλεσμά τους είναι μια λογική τιμή.

Σύνθετες εκφράσεις: Δημιουργούνται από το συνδυασμό 2 ή περισσότερων λογικών εκφράσεων και το αποτέλεσμά τους είναι μια λογική τιμή.

23. Τι είναι μια μεταβλητή. (SOS - 2 απαντήσεις: Φυλλάδιο και σελ 35)

ΑΠ: Μια μεταβλητή χρησιμοποιείται για να αποθηκεύσουμε ένα δεδομένο. Αντιστοιχεί σε μια θέση μνήμης, στην οποία κάθε φορά μπορεί να αποθηκευτεί μια μόνο τρέχουσα τιμή και μπορεί να αντικατασταθεί με μια άλλη.

Κάθε μεταβλητή έχει ένα όνομα και μια τιμή. Ο τύπος της μεταβλητής είναι ίδιος με τον τύπο του δεδομένου το οποίο αποθηκεύεται στη μεταβλητή αυτή.

24. Τι χρειάζεται για να χρησιμοποιήσουμε μια μεταβλητή. (σελ 33)

ΑΠ: Για να χρησιμοποιήσουμε μια μεταβλητή πρέπει να της δώσουμε ένα όνομα και στη συνέχεια να εκχωρήσουμε κάποια τιμή.

25. Ποιους κανόνες πρέπει να ακολουθούμε κατά την ονομασία μιας μεταβλητής. (σελ 33-34)

ΑΠ: Δεν επιτρέπεται να ξεκινάει από αριθμό, δεν πρέπει να περιέχει κενό. Μπορεί να ξεκινάει με μικρό ή κεφαλαίο γράμμα. Δεν μπορεί να πάρει όνομα που θα είναι ίδιο με κάποιο όνομα ενσωματωμένης συνάρτησης ή εντολής. Συνηθίζουμε να δίνουμε ένα όνομα σχετικό με το είδος της μεταβλητής με λατινικούς χαρακτήρες, που μπορεί να συνοδεύεται

από κάποιον αριθμό ή την _. Επίσης οι μεταβλητές με πεζά είναι διαφορετικές από μεταβλητές με τα ίδια κεφαλαία (case sensitive).

26. Τι είναι ο τελεστής εκχώρησης τιμής (=). (σελ 34)

ΑΠ: Ο τελεστής εκχώρησης χρησιμοποιείται για να δώσουμε μια τιμή σε μια μεταβλητή.

27. Πώς γίνεται η εκχώρηση τιμής σε μια μεταβλητή. Εξηγήστε με ένα παράδειγμα. (σελ 32)

ΑΠ: Η εκχώρηση μιας τιμής σε μια μεταβλητή γίνεται με την χρήση του τελεστή εκχώρησης. Στο αριστερό τμήμα δίνουμε το όνομα της μεταβλητής, στη συνέχεια χρησιμοποιούμε τον τελεστή εκχώρησης "=" και στο δεξί μέρος βάζουμε την τιμή, μια άλλη μεταβλητή μια έκφραση που έχει αποτέλεσμα μια τιμή. Πχ x=4

28. Πώς αναπαρίστανται τα δεδομένα σε ένα πρόγραμμα Python. (σελ 34-35)

ΑΠ: Όλα τα δεδομένα σε ένα πρόγραμμα Python αναπαρίστανται με αντικείμενα ή με σχέσεις μεταξύ των αντικειμένων, με κάθε αντικείμενο να έχει μια ταυτότητα, έναν τύπο και μια τιμή.

29. Τι ονομάζουμε «συλλογή σκουπιδιών» στην Python. (σελ 35)

ΑΠ: Η Python παρακολουθεί όλες τις τιμές και τις διαγράφει όταν πάψουν να υπάρχουν μεταβλητές που να αναφέρονται σε αυτές. Η διαδικασία αυτή ονομάζεται «*συλλογή σκουπιδιών*».

30. Ποιες είναι οι ενσωματωμένες συναρτήσεις της Python και ποια η λειτουργία τους. (σελ 39)

ΑΠ: Οι ενσωματωμένες συναρτήσεις είναι:

● Η **float()** μετατρέπει ακεραίους και συμβολοσειρές σε δεκαδικούς αριθμούς.

● Η **int()** δέχεται οποιαδήποτε αριθμητική τιμή και τη μετατρέπει σε ακέραιο κόβοντας τα δεκαδικά ψηφία, αν υπάρχουν.

● Η **str()** δέχεται οποιαδήποτε τιμή και την μετατρέπει σε συμβολοσειρά.

● Η **abs()** επιστρέφει την απόλυτη τιμή ενός αριθμού.

● Η **pow(a,b)** επιστρέφει τη δύναμη του α υψωμένη στο β.

● Η **divmod(x,y)** επιστρέφει το ακέραιο πηλίκο και το ακέραιο υπόλοιπο της διαίρεσης x/y.

31. Τι είναι οι εξωτερικές βιβλιοθήκες. Πως χρησιμοποιούνται στην Python. (σελ 40)

ΑΠ: Μια βιβλιοθήκη είναι ένα αρχείο το οποίο περιέχει μια συλλογή από σχετικές συναρτήσεις. Οι βιβλιοθήκες αυτές για να χρησιμοποιηθούν, θα πρέπει πρώτα να εισαχθούν στο πρόγραμμά μας. Η εισαγωγή αυτή γίνεται με την εντολή **import**.

32. Ποιες καλές πρακτικές πρέπει να τηρούμε κατά τη συγγραφή ενός προγράμματος. (σελ 41)

ΑΠ: Μερικές καλές πρακτικές και συντακτικές συμβάσεις που πρέπει να τηρούμε κατά τη συγγραφή ενός προγράμματος, είναι οι παρακάτω:

● Δίνουμε ένα χαρακτηριστικό τίτλο στο πρόγραμμα με τη μορφή σχολίων, τα οποία ξεκινάνε με το σύμβολο #. Αυτό, παρότι δεν είναι αναγκαίο, είναι ιδιαίτερα χρήσιμο.

● Προσέχουμε τα κενά διαστήματα πριν την κάθε εντολή, καθώς η Python βασίζεται σε αυτά, για να ορίσει ομάδες εντολών.

● Επιλέγουμε τους κατάλληλους τελεστές.

● Να χρησιμοποιούμε τα ίδια εισαγωγικά (μονά εισαγωγικά με μονά, διπλά εισαγωγικά με διπλά) για μια συμβολοσειρά.

● Προσθέτουμε, όπου κρίνουμε χρήσιμο, επεξηγηματικά σχόλια μέσα στον κώδικα. Θυμίζουμε ότι τα σχόλια περιγράφουν τη λειτουργία ενός προγράμματος ή γενικότερα, ενός τμήματος κώδικα και γράφονται, για να βοηθήσουν τους ανθρώπους -και όχι τον υπολογιστή- στην κατανόηση και συντήρηση ενός προγράμματος. Όταν ένα πρόγραμμα μεταφράζεται, τα σχόλια αγνοούνται. Τα σχόλια που περιγράφουν τη λειτουργία ενός προγράμματος, το ρόλο των μεταβλητών και τη λειτουργία πολύπλοκων τμημάτων κώδικα, αποτελούν παράδειγμα καλού προγραμματιστικού στυλ.

● Δίνουμε ονόματα μεταβλητών που έχουν σχέση με τη χρήση τους.

33. Τι είναι τα σχόλια σε ένα πρόγραμμα Python. (σελ 38)

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

34. Τι είναι ένας Τύπος Δεδομένων. Ποιες μεγάλες κατηγορίες υπάρχουν. Πώς διακρίνονται ανάλογα με τη σύσταση των μερών τους; (σελ 42)

ΑΠ: Ένας Τύπος Δεδομένων είναι ένα σύνολο τιμών δεδομένων και λειτουργιών επί αυτών των τιμών. Οι τύποι δεδομένων είτε είναι προκαθορισμένοι από τις γλώσσες προγραμματισμού και καλούνται Πρωτογενείς τύποι δεδομένων είτε δημιουργούνται από τον προγραμματιστή, οπότε καλούνται Μη πρωτογενείς τύποι δεδομένων. Διακρίνονται δε, ανάλογα με τη σύσταση των μερών που τους αποτελούν, σε: Απλούς τύπους και Σύνθετους τύπους δεδομένων.

35. Τι είναι οι απλοί τύποι δεδομένων; Ποιοι απλοί τύποι δεδομένων υπάρχουν στις γλώσσες προγραμματισμού. (σελ 43)

ΑΠ: Στους απλούς τύπους δεδομένων, οι τιμές των δεδομένων είναι στοιχεία μη περαιτέρω-χωριζόμενα (άτομα). Δηλαδή κάθε δεδομένο έχει μία και μοναδική τιμή. Τέτοιοι είναι, οι ακέραιοι αριθμοί, οι πραγματικοί αριθμοί, οι χαρακτήρες, ο λογικός τύπος δεδομένων (με τιμές True και False) και τα αλφαριθμητικά.

36. Τι είναι οι σύνθετοι τύποι δεδομένων; Ποιοι σύνθετοι τύποι δεδομένων υπάρχουν στις γλώσσες προγραμματισμού. (σελ 43)

ΑΠ: Σύνθετος τύπος δεδομένων είναι εκείνος, που αποτελείται από Πρωτογενείς ή / και άλλους σύνθετους τύπους, όπου μια μεταβλητή μπορεί να πάρει ως τιμή μια ενότητα τιμών. Οι σύνθετοι τύποι καλούνται και Δομές Δεδομένων. Παραδείγματα σύνθετων τύπων είναι η Εγγραφή, το Σύνολο, ο Πίνακας, οι Λίστες, οι Στοίβες, ο Σωρός, οι Ουρές, το Δένδρο, ο Γράφος.

37. Τι ονομάζουμε «αφηρημένο τύπο δεδομένων». (σελ 44 ή φυλλάδιο)

ΑΠ: Όταν σχεδιάζεται ένας τύπος δεδομένων, αρχικά υπάρχει μόνο διανοητικά και ορίζεται μόνο με βάση τις λειτουργίες που επιτελεί χωρίς να μας απασχολεί ο τρόπος που θα αναπαρασταθούν τα δεδομένα αλλά ούτε και ο τρόπος που θα υλοποιηθούν οι λειτουργίες σε κώδικα. Μα ενδιαφέρει, δηλαδή, το τι θα υπολογιστεί και όχι το πώς. Με αυτό τον τρόπο ορίζεται ένας αφηρημένος τύπος δεδομένων.

38. Να περιγράψετε τη λειτουργία της εντολής εξόδου print. (σελ 36 και φυλλάδιο)

ΑΠ: Η εντολή print χρησιμοποιείται για να εμφανίσουμε τα αποτελέσματα του προγράμματός μας στην οθόνη. Είναι ένας από τους τρόπους που χαρακτηρίζουν την έξοδο του προγράμματός μας.

Για την εμφάνιση των αποτελεσμάτων γράφεται η εντολή print και μπορεί να ακολουθεί κάποιο από τα παρακάτω στοιχεία:

- Μια ή περισσότερες μεταβλητές, χωρισμένες με κόμμα μεταξύ τους, η τιμή των οποίων θα εμφανιστεί στην οθόνη.

- Ένας αριθμός, ο οποίος εμφανίζεται στην οθόνη.

- Μια συμβολοσειρά, η οποία εμφανίζεται στην οθόνη.
- Μια έκφραση, η οποία υπολογίζεται και το αποτέλεσμα εμφανίζεται στην οθόνη.
- Συνδυασμός των παραπάνω, χωρισμένα με κόμμα μεταξύ τους, τα οποία εμφανίζονται σε μία σειρά.

Μια εντολή print χωρίς κανένα στοιχείο από τα παραπάνω, εμφανίζει μια κενή γραμμή στην οθόνη.

39. Να περιγράψετε τη λειτουργία της εντολής εισόδου input και raw_input. (σελ 37-38 και φυλλάδιο)

ΑΠ: Οι εντολές input και raw_input χρησιμοποιούνται για να εισαχθεί μια τιμή από το πληκτρολόγιο, η οποία θα αποδοθεί σε μια μεταβλητή.

Κατά την εκτέλεση των εντολών αυτών, διακόπτεται προσωρινά το πρόγραμμα και περιμένει από το χρήστη να εισάγει μία τιμή από το πληκτρολόγιο. Η τιμή που θα δοθεί αποδίδεται αυτόματα στη μεταβλητή που βρίσκεται πριν από το =. Είναι ένας από τους τρόπους που χαρακτηρίζουν την είσοδο του προγράμματός μας. Το «μήνυμα» στις παρενθέσεις δεν είναι υποχρεωτικό να υπάρχει, αν υπάρχει, όμως εμφανίζεται στην οθόνη πριν πληκτρολογηθεί η τιμή.

Η διαφορά ανάμεσα στις δυο εντολές βρίσκεται στον τύπο δεδομένων που αναμένει το πρόγραμμα να εισαχθούν. Η εντολή raw_input() διαβάζει μόνο αλφαριθμητικά δεδομένα, ενώ η εντολή input() αριθμητικά ή λογικά δεδομένα.

ΚΕΦΑΛΑΙΟ 4

40. Τι ονομάζουμε αλγοριθμική δομή. (Φυλλάδιο)

ΑΠ: Με τον όρο αλγοριθμική δομή εννοούμε τον τρόπο με τον οποίο εκτελούνται οι εντολές που υπάρχουν σε ένα αλγόριθμο (πρόγραμμα).

41. Να αναφέρετε ποιες βασικές αλγοριθμικές δομές χρησιμοποιούνται για την ανάπτυξη ενός προγράμματος. (Φυλλάδιο)

ΑΠ: Οι αλγοριθμικές δομές που χρησιμοποιούνται σε ένα αλγόριθμο είναι:

- Η δομή ακολουθίας
- Η δομή επιλογής
- Η δομή επανάληψης

42. Να περιγράψετε την αλγοριθμική δομή ακολουθίας. (σελ 45)

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

43. Τι είδους προβλήματα μπορούμε να λύσουμε με τη δομή ακολουθίας. (σελ 45)

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

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

44. Πότε χρησιμοποιείται η δομή επιλογής. (σελ 46)

ΑΠ: Η δομή επιλογής if (ΑΝ) χρησιμοποιείται, όταν θέλουμε να εκτελεστεί μια ακολουθία εντολών, μόνον, εφόσον πληρείται μία συγκεκριμένη συνθήκη. Για να επιλύσουμε πιο σύνθετα προβλήματα, πρέπει να είμαστε σε θέση να δημιουργούμε στον αλγόριθμο λογικά

μονοπάτια, εξετάζοντας απλά λογικά ερωτήματα για την εκτέλεση ή όχι μιας ομάδας εντολών.

45. Με ποια εντολή υλοποιείται η δομή επιλογής στην Python. (σελ 46 ή φυλλάδιο)

ΑΠ: Αν η συνθήκη είναι αληθής, τότε εκτελείται το σύνολο των εντολών που περιέχονται στην δομή if.

46. Πότε χρησιμοποιείται η απλή δομή επιλογής. (σελ 46)

ΑΠ: Αν η συνθήκη είναι αληθής, τότε το σύνολο των εντολών που περιέχονται στην δομή if, θα εκτελεστούν. Αλλιώς, η ροή του προγράμματος θα προσπεράσει τη δομή if και θα συνεχίσει από την εντολή που βρίσκεται αμέσως μετά το τέλος της if.

47. Να γράψετε τον τρόπο χρήσης της εντολής if στην Python και να περιγράψετε τη λειτουργία της. (σελ 46)

ΑΠ: *if <συνθήκη ελέγχου>*:

Οι εντολές στο μπλόκ της if θα εκτελεσθούν αν ισχύει (αληθής-True) η συνθήκη ελέγχου

48. Πότε χρησιμοποιείται η σύνθετη δομή επιλογής. (σελ 47)

ΑΠ: Αν, ανάλογα με την αποτίμηση μιας συνθήκης, θέλουμε να εκτελεστούν διαφορετικές ακολουθίες εντολών, τότε μπορούμε να χρησιμοποιήσουμε τη δομή επιλογής if…else (ΑΝ…ΑΛΛΙΩΣ).

49. Να γράψετε τον τρόπο χρήσης της συνθέτης δομής επιλογής στην Python και να περιγράψετε τη λειτουργία της. (σελ 47-48)

ΑΠ: if <συνθήκη ελέγχου>:

```
    #εντολές που θα εκτελεσθούν, αν η συνθήκη ελέγχου είναι αληθής
else:
    #εντολές που θα εκτελεσθούν αν η συνθήκη ελέγχου είναι ψευδής
```

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

50. Πότε χρησιμοποιείται η πολλαπλή επιλογή. (σελ 49)

ΑΠ: Η πολλαπλή επιλογή χρησιμοποιείται όταν θέλουμε να εκτελεστούν διαφορετικές ακολουθίες εντολών ανάλογα με το εάν μια συνθήκη ελέγχεται πολλαπλώς είναι αληθής ή ψευδής.

51. Με ποια εντολή υλοποιείται η πολλαπλή επιλογή στην Python. Να περιγράψετε τον τρόπο χρήσης της και τη λειτουργία της. (σελ 49)

ΑΠ: Η εντολή που χρησιμοποιείται είναι η elif. Η σύνταξη είναι η εξής:

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

Γίνεται έλεγχος μιας συνθήκης, εάν είναι αληθής εκτελούνται το μπλοκ εντολών ενώ εάν είναι ψευδής, τότε εκτελείται έλεγχος της συνθήκης 2. Εάν είναι αληθής, τότε εκτελούνται οι ελντολές_2 ενώ αν είναι ψευδής, τότε εκτελούνται οι εντολές_3.

52. Πότε χρησιμοποιείται η εμφωλευμένη δομή επιλογής. (Φυλλάδιο)
ΑΠ: Η εμφωλευμένη δομή επιλογής χρησιμοποιείται όταν οι περιπτώσεις επιλογής είναι περισσότερες από δύο.
53. Με ποιον τρόπο υλοποιείται η εμφωλευμένη επιλογή στην Python. (Φυλλάδιο)
ΑΠ: Η εμφωλευμένη δομή επιλογής υλοποιείται στην Python τοποθετώντας εντολές **if….else** μέσα σε άλλες εντολές **if…else**
54. Πότε χρησιμοποιείται η δομή επανάληψης. (σελ 51)
ΑΠ: Όταν μια ομάδα εντολών είναι αναγκαίο να εκτελείται περισσότερες από μία φορές
55. Ποιοι τύποι επανάληψης υπάρχουν. (σελ 51)
ΑΠ: Υπάρχουν δύο τύποι επαναλήψεων:
● Οι **προκαθορισμένοι**, όπου το πλήθος των επαναλήψεων είναι δεδομένο, πριν αρχίσουν οι επαναλήψεις. Για παράδειγμα: ο υπολογισμός του μέσου όρου βαθμολογίας των μαθητών ενός τμήματος 22 μαθητών.
● Οι **μη προκαθορισμένοι**, όπου το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών του σώματος της επανάληψης. Για παράδειγμα: ο υπολογισμός των μορίων όσων υποβάλλουν αίτηση σε ένα διαγωνισμό του Δημοσίου.
56. Ποιες εντολές επανάληψης χρησιμοποιεί Python. (σελ 51)
ΑΠ: Στη γλώσσα προγραμματισμού Python, χρησιμοποιούμε την εντολή **for** για να εκτελεστεί ένα τμήμα του κώδικα για έναν καθορισμένο αριθμό επαναλήψεων, ενώ την εντολή **while** για να εκτελείται υπό συνθήκη και μάλιστα, όσο αυτή είναι αληθής
57. Να εξηγήσετε τη λειτουργία της συνάρτησης range() στην Python. (σελ 52 και σελ 137)
ΑΠ: Η range() είναι μια ενσωματωμένη συνάρτηση της γλώσσας Python, η οποία, ανάμεσα σε άλλα, χρησιμοποιείται για την υπόδειξη του αριθμού των επαναλήψεων που θα εκτελεστούν σε ένα βρόχο. Η δομή της είναι range(Α, Μ, Β) και επιστρέφει μια λίστα αριθμών ξεκινώντας με τον αριθμό Α μέχρι το Μ με βήμα Β. Το Μ δεν συμπεριλαμβάνεται στη λίστα.
58. Πότε χρησιμοποιείται η εντολή for. Να γράψετε τον τρόπο χρήσης της εντολής for στην Python και να περιγράψετε τη λειτουργία της. (Φυλλάδιο)
ΑΠ: Η εντολή for χρησιμοποιείται όταν θέλουμε ένα σύνολο εντολών να εκτελεστεί προκαθορισμένες φορές, δηλαδή ο αριθμός των επαναλήψεων να είναι γνωστός πριν από την έναρξη της επανάληψης.
Η εντολή for εκτελείται τόσες φορές όσο είναι το πλήθος των αριθμών που παράγει η συνάρτηση range(). Η μεταβλητή της for στην πρώτη επανάληψη λαμβάνει τον πρώτο αριθμό από τη λίστα των αριθμών που παράγει η range() και σε κάθε νέα επανάληψη λαμβάνει τον επόμενο από τη λίστα των αριθμών αυτών.
59. Πότε χρησιμοποιείται η εντολή while. (σελ 53)
ΑΠ: Η δομή while (Όσο <συνθήκη> επανάλαβε) χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων. Σε κάθε επανάληψη (και στην αρχική) πραγματοποιείται ο έλεγχος της συνθήκης, πριν από την εκτέλεση των εντολών του βρόχου, πράγμα που σημαίνει ότι υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου.
60. Να γράψετε τον τρόπο χρήσης της εντολής while στην Python και να περιγράψετε τη λειτουργία της. (σελ 53)
ΑΠ: Η χρήση της εντολής while είναι η εξής:
Αρχική τιμή μεταβλητής
while ονομα_μεταβλητής <συνθήκη>:
    Εντολή_1
    Εντολή_2
    ….
    Εντολή_κ
Αρχικά δίνουμε μια τιμή σε μια μεταβλητή. Γίνεται ο έλεγχος μια συνθήκης με την μεταβλητή αυτή. Από την στιγμή που είναι αληθής, θα εκτελούνται οι εντολές μέσα στον βρόγχο while. Μια από τις εντολές πρέπει να είναι και αυτή που θα αλλάζει την τιμή της μεταβλητής ώστε μετά από κάποιες επαναλήψεις, η συνθήκη να γίνει ψευδής.

61. Ποιες είναι οι προϋποθέσεις για τη σωστή λειτουργία της εντολής while. (σελ 53, Σημείωση 1 και 2)

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

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

62. Πώς μπορούμε να ελέγξουμε την εγκυρότητα των δεδομένων εισόδου από το χρήστη. (σελ 54)

ΑΠ: Με την εφαρμογή του while βρόχου. Στην περίπτωση αυτή η συνθήκη που ελέγχουμε είναι η επιθυμητή τιμή και σε περίπτωση που είναι εκτός της επιθυμητής τιμής, επαναλαμβάνεται η ερώτηση εισαγωγής δεδομένων.

63. Τι είναι η εμφωλευμένη δομή επανάληψης. (Φυλλάδιο)

ΑΠ: Η εμφωλευμένη δομή επανάληψης δημιουργείται όταν μια εντολή επανάληψης βρίσκεται μέσα σε μια άλλη. Πολλές φορές ονομάζεται και ως «εμφωλευμένοι βρόγχοι». Η εντολή επανάληψης πουβρίσκεται μέσα στην άλλη δεν έχει σημασία. Μπορεί να μια for να βρίσκεται μέσα σε μία while ή να εμφωλεύεται οποιοσδήποτε άλλος συνδυασμός των εντολών επανάληψης.

64. Τι είναι οι συναρτήσεις. (σελ 57 και σελ 106)

ΑΠ: Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Μας επιτρέπουν να δίνουμε ένα όνομα σε ένα σύνολο εντολών και να το εκτελούμε καλώντας το όνομα αυτό, από οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε.

65. Τι ονομάζεται κλήση της συνάρτησης. (σελ 57 και σελ 107-108)

ΑΠ: Ο ορισμός μιας συνάρτησης περιλαμβάνει το όνομά της και τις παραμέτρους εισόδου και καλείται από σημεία του προγράμματος μέσω της λειτουργίας που ονομάζεται κλήση (calling) της συνάρτησης.

66. Πώς μπορούμε να ορίσουμε μία δική μας συνάρτηση. (σελ 57 και σελ 107)

ΑΠ: Ορίζεται με τη λέξη κλειδί def που την ακολουθεί ένα όνομα το οποίο την ταυτοποιεί και ένα ζεύγος παρενθέσεων που μπορούν να περιέχουν ονόματα μεταβλητών, ενώ η δήλωση τελειώνει με διπλή τελεία (:). Κάτω από τη γραμμή αυτή τοποθετούνται, σε εσοχή, οι εντολές που καθορίζουν τη λειτουργία της συνάρτησης.

67. Τι είναι οι παράμετροι μιας συνάρτησης. (σελ 58 και σελ 110)

ΑΠ: Μια συνάρτηση δέχεται δεδομένα μέσω των **παραμέτρων** και επιστρέφει τα αποτελέσματα μέσω άλλων ή και των ίδιων παραμέτρων στο πρόγραμμα ή σε άλλη συνάρτηση. Οι παράμετροι καθορίζονται μέσα στο ζευγάρι των παρενθέσεων στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα. Όταν καλούμε τη συνάρτηση, δίνουμε και τις τιμές με τον ίδιο τρόπο, οι οποίες τιμές ονομάζονται **ορίσματα**.

68. Πώς χρησιμοποιούνται οι παράμετροι μιας συνάρτησης. (σελ 58 και σελ 110 – 111)

ΑΠ: Οι παράμετροι μοιάζουν με τις μεταβλητές, καθορίζονται μέσα στο ζευγάρι των παρενθέσεων, στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα, ενώ οι τιμές αυτών των μεταβλητών ορίζονται, όταν καλούμε τη συνάρτηση

69. Τι ονομάζονται ορίσματα. (σελ 58 και σελ 110 – 111)

ΑΠ: Όταν καλούμε τη συνάρτηση, δίνουμε και τις τιμές με τον ίδια σειρά που έχουν οριστεί στο ζεύγος των παρενθέσεων κατά τον ορισμό της συνάρτησης. Οι τιμές αυτές ονομάζονται ορίσματα.

70. Ποια η διαφορά ανάμεσα στις παραμέτρους και τα ορίσματα. (σελ 58 και σελ 110 – 111)

ΑΠ: Παράμετροι ονομάζονται οι μεταβλητές μέσα στο ζεύγος των παρενθέσεων κατά τον ορισμό της συνάρτησης ενώ ορίσματα ονομάζονται οι μεταβλητές μέσα στο ζεύγος των παρενθέσεων κατά την κλήση της συνάρτησης.

71. Τι κάνει η εντολή import random. (σελ 54)

ΑΠ: Η εντολή "import random" εισάγει μια βιβλιοθήκη συναρτήσεων για την παραγωγή τυχαίων αριθμών. Η συνάρτηση random επιστρέφει έναν τυχαίο δεκαδικό ανάμεσα στο 0.0 και στο 1.0 (συμπεριλαμβανομένου του 0.0, αλλά όχι του 1.0).

ΚΕΦΑΛΑΙΟ 7

72. Τι ονομάζεται τμηματικός προγραμματισμός. (σελ 105)

ΑΠ: Τμηματικός προγραμματισμός είναι μια τεχνική όπου μπορούμε να γράψουμε ένα πρόγραμμα ως ένα σύνολο από μικρότερα κομμάτια προγράμματος. Με αυτόν τον τρόπο δεν είναι απαραίτητο να ξαναγράφουμε τον ίδιο κώδικα σε διαφορετικές υλοποιήσεις. Τον γράφουμε μία φορά, δημιουργώντας ένα υποπρόγραμμα και τον χρησιμοποιούμε, όσες φορές θέλουμε σε ένα πρόγραμμα, καλώντας το αντίστοιχο υποπρόγραμμα. Ένα υποπρόγραμμα λοιπόν είναι ένα κομμάτι προγράμματος που έχει γραφεί ξεχωριστά από το υπόλοιπο πρόγραμμα και επιτελεί ένα αυτόνομο έργο.

73. Ποια ανάγκη οδήγησε στην ανάπτυξη του τμηματικού προγραμματισμού. (σελ 104 – 105)

ΑΠ: Συχνά κάποια ενέργεια ή και ολόκληρο αλγοριθμικό τμήμα, είναι αναγκαίο να επαναλαμβάνεται, σχεδόν αυτούσια ως λογική επεξεργασία, σε πολλά διαφορετικά σημεία του αλγορίθμου, αλλά κάθε φορά να επεξεργάζεται διαφορετικά δεδομένα. Ο τρόπος αυτός “επανάληψης” δεν μπορεί να υλοποιηθεί με μια δομή επανάληψης, λόγω της δυνατότητας επεξεργασίας διαφορετικών δεδομένων κάθε φορά. Δεν είναι προγραμματιστικά ορθό να επαναλαμβάνουμε το τμήμα κώδικα με άλλα δεδομένα κάθε φορά, αλλά να επιτρέπει η γλώσσα προγραμματισμού τη δημιουργία ομάδας εντολών, ως οντότητα στο πρόγραμμα, που να δέχεται ως είσοδο παραμέτρους και να μπορεί να καλείται από σημεία του προγράμματος και με συγκεκριμένες τιμές για τις παραμέτρους ανά κλήση. Αυτό έχει ως αποτέλεσμα τη δυνατότητα πολλαπλής διενέργειας μιας επεξεργασίας με συγκεκριμένα δεδομένα κάθε φορά

74. Τι είναι ένα υποπρόγραμμα. (σελ 105 και φυλλάδιο)

ΑΠ: Είναι κώδικας που γράφουμε μία φορά, δημιουργώντας ένα υποπρόγραμμα και τον χρησιμοποιούμε, όσες φορές θέλουμε σε ένα πρόγραμμα, καλώντας το αντίστοιχο υποπρόγραμμα. Ένα υποπρόγραμμα λοιπόν είναι ένα κομμάτι προγράμματος που έχει γραφεί ξεχωριστά από το υπόλοιπο πρόγραμμα και επιτελεί ένα αυτόνομο έργο.

75. Τι βοήθεια μας προσφέρει ο τμηματικός προγραμματισμός. (σελ 105)

ΑΠ: Να χωρίσουμε το πρόγραμμα σε ένα σύνολο από μικρότερα κομμάτια προγράμματος.

76. Ποια είναι τα χαρακτηριστικά ενός υποπρογράμματος. (σελ 105)

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

α) Έχει μόνο ένα σημείο εισόδου από το οποίο δέχεται τα δεδομένα του.

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

γ) Ο έλεγχος επιστρέφει στο (υπο)πρόγραμμα το οποίο καλεί, όταν το καλούμενο υποπρόγραμμα σταματήσει να εκτελείται.

77. Ποιες καλές πρακτικές πρέπει να εφαρμόζουμε κατά την υλοποίηση του τμηματικού προγραμματισμού. (σελ 106)

ΑΠ: Πριν ξεκινήσουμε να γράφουμε ένα πρόγραμμα, μελετάμε πώς αυτό μπορεί να αναλυθεί σε επιμέρους τμήματα και αποφασίζουμε για τα αντίστοιχα υποπρογράμματα.

Εξετάζουμε αν κάποια υποπρογράμματα, τα οποία έχουμε ήδη γράψει ή υπάρχουν σε έτοιμες βιβλιοθήκες προγραμμάτων, μπορούν να χρησιμοποιηθούν, για να κερδίσουμε χρόνο και να αποφύγουμε λάθη.

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

78. Πώς υλοποιείται ο τμηματικός προγραμματισμός στην Python. (σελ 106)

ΑΠ: Στην Python, η τεχνική αυτή υλοποιείται με τις συναρτήσεις

79. Πόσοι τύποι υποπρογραμμάτων υπάρχουν στην Python. (σελ 106)

ΑΠ: Η Python παρέχει ένα μόνο τύπο υποπρογραμμάτων, τις συναρτήσεις, τις οποίες τις θεωρεί ως αντικείμενα.

80. Να γράψετε και να εξηγήσετε πώς ορίζεται μία συνάρτηση στην Python. (σελ 107)

ΑΠ: Ο ορισμός μιας συνάρτησης περιλαμβάνει το όνομά της και τις παραμέτρους εισόδου και καλείται από σημεία του προγράμματος μέσω της λειτουργίας που ονομάζεται κλήση

(calling) της συνάρτησης. Ορίζεται με τη λέξη κλειδί def που την ακολουθεί ένα όνομα το οποίο την ταυτοποιεί και ένα ζεύγος παρενθέσεων που μπορούν να περιέχουν ονόματα μεταβλητών, ενώ η δήλωση τελειώνει με διπλή τελεία (:). Κάτω από τη γραμμή αυτή τοποθετούνται, σε εσοχή, οι εντολές που καθορίζουν τη λειτουργία της συνάρτησης.

81. Από ποια σημεία μπορεί να καλείται μια συνάρτηση. (σελ 107)

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

82. Να γράψετε τον τρόπο χρήσης της εντολής return στην Python και να περιγράψετε τη λειτουργία της. (Φυλλάδιο)

ΑΠ: Η εντολή return χρησιμοποιείται για την επιστροφή τιμής από μια συνάρτηση. Η σύνταξη της εντολής είναι:

return <αποτέλεσμα>

Αν κατά τη ροή εκτέλεσης των εντολών μιας συνάρτησης, βρεθεί μια εντολή return, τότε διακόπτεται η εκτέλεση της συνάρτησης και ο έλεγχος ροής επιστρέφει από τη συνάρτηση στο σημείο του προγράμματος που αυτή κλήθηκε, χρησιμοποιώντας την τιμή του αποτελέσματος ως επιστρεφόμενη τιμή. Η εντολή return δεν είναι υποχρεωτικό να υπάρχει σε κάθε συνάρτηση.

83. Πώς καλείται μία συνάρτηση στην Python. (σελ 108)

ΑΠ: Μια συνάρτηση καλείται με το όνομα που έχει οριστεί στην αρχή του προγράμματος, ακολουθούμενη με παρενθέσεις όπου εισάγονται τα ορίσματα (σε περίπτωση που υπάρχουν).

84. Να εξηγήσετε το μηχανισμό κλήσης μιας συνάρτησης από το κύριο πρόγραμμα μέσω ενός παραδείγματος. (σελ 108)

ΑΠ: Κλήση της συνάρτησης find_sum, που δημιουργήσαμε παραπάνω:

```
>>> find_sum(3,4)
7
```

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

85. Τι ονομάζεται επιστρεφόμενη τιμή. (σελ 108)

ΑΠ: Η συνάρτηση δέχεται ορίσματα και επιστρέφει ένα αποτέλεσμα. Το αποτέλεσμα αυτό ονομάζεται επιστρεφόμενη τιμή

86. Με ποιους τρόπους μπορούμε να ακολουθήσουμε τη ροή εκτέλεσης ενός προγράμματος. (σελ 108)

ΑΠ: Πρώτον να διαβάζουμε προγράμματα είναι να ακολουθούμε τη ροή εκτέλεσης των εντολών, κάτι το οποίο μπορεί να γίνει γρήγορα πολύπλοκο και δυσανάγνωστο.

Δεύτερον, όταν βρεθούμε σε μια κλήση συνάρτησης, αντί να ακολουθήσουμε τη ροή της εκτέλεσης στον εσωτερικό κώδικα της συνάρτησης, να υποθέτουμε ότι η συνάρτηση δουλεύει σωστά και επιστρέφει το σωστό αποτέλεσμα. Οπότε, να συνεχίζουμε με την επόμενη εντολή του κύριου προγράμματος.

87. Ποιες κατηγορίες συναρτήσεων υπάρχουν. (σελ 109 και σελ 110)

ΑΠ: Μια πρώτη κατηγορία συναρτήσεων είναι:

α) αυτές οι οποίες δεν τροποποιούν το αντικείμενο στο οποίο εφαρμόζονται.

β) εκείνες που μπορούν να αλλάξουν το αντικείμενο στο οποίο καλούνται

Μια άλλη κατηγορία συναρτήσεων είναι:

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

β) εκείνες που δεν επιστρέφουν κάποια τιμή (κενές/void συναρτήσεις), αλλά εκτελούν ενέργειες μέσω των εντολών τους. Οι κενές αυτές συναρτήσεις μπορεί να εμφανίζουν αποτέλεσμα στην οθόνη ή να έχουν κάποιο άλλο αποτέλεσμα, αλλά δεν επιστρέφουν κάποια τιμή.

88. Τι είναι μία κενή συνάρτηση. (σελ 110)

ΑΠ: Οι συναρτήσεις που δεν επιστρέφουν κάποια τιμή (κενές/void συναρτήσεις), αλλά εκτελούν ενέργειες μέσω των εντολών τους. Οι κενές αυτές συναρτήσεις μπορεί να εμφανίζουν αποτέλεσμα στην οθόνη ή να έχουν κάποιο άλλο αποτέλεσμα, αλλά δεν επιστρέφουν κάποια τιμή.

89. Τι ονομάζεται παράμετρος ενός υποπρογράμματος. (σελ 110)
ΑΠ: Οι παράμετροι είναι μεταβλητές οι οποίες χρησιμοποιούνται για τη μεταβίβαση τιμών μεταξύ υποπρογραμμάτων ή υποπρογραμμάτων και κύριου προγράμματος
90. Ποια είναι η διαφορά μεταξύ μεταβλητής και παραμέτρου. (σελ 110)
ΑΠ: Μια συνάρτηση μπορεί να δεχθεί παραμέτρους, οι οποίες είναι τιμές που δίνονται/εισέρχονται στη συνάρτηση, έτσι ώστε αυτή να μπορεί να λειτουργήσει αξιοποιώντας αυτές τις τιμές. Οι παράμετροι μοιάζουν με τις μεταβλητές, καθορίζονται μέσα στο ζευγάρι των παρενθέσεων, στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα, ενώ οι τιμές αυτών των μεταβλητών ορίζονται, όταν καλούμε τη συνάρτηση.
91. Τι σημαίνει ότι το πέρασμα παραμέτρων είναι κατά τιμή. (σελ 110-112)
ΑΠ: Η παράμετρος κατά τιμή είναι τοπική στη συνάρτηση. Αλλάζοντας την τιμή της παραμέτρου μέσα στη συνάρτηση, η αλλαγή αυτή εφαρμόζεται μόνο στην "περιοχή" της συνάρτησης και δεν αλλάζει την τιμή της παραμέτρου στο πρόγραμμα.
92. Τι σημαίνει ότι το πέρασμα παραμέτρων είναι με αναφορά. (σελ 112)
ΑΠ: Όταν οι παράμετροι περνάνε με αναφορά, σημαίνει ότι αν αλλάξουμε μια παράμετρο μέσα στη συνάρτηση, η αλλαγή είναι μόνιμη και μετά την κλήση της συνάρτησης.
93. Τι ονομάζεται εμβέλεια μεταβλητών. (σελ 113)
ΑΠ: Εμβέλεια μεταβλητών ονομάζεται η πρόσβαση που έχει μια μεταβλητή στα τμήματα ενός προγράμματος.
94. Πώς καθορίζεται η εμβέλεια μιας μεταβλητής. (σελ 113)
ΑΠ: Οι γλώσσες προγραμματισμού επιτρέπουν τη χρήση των μεταβλητών, όχι μόνο στο τμήμα προγράμματος που δηλώνονται, αλλά και σε άλλα υποπρογράμματα
95. Ποια είδη εμβέλειας μεταβλητών υπάρχουν. (σελ 113)
ΑΠ: Απεριόριστη εμβέλεια: Όλες οι μεταβλητές είναι ορατές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος, ανεξάρτητα από το πού δηλώθηκαν. Αυτού του τύπου οι μεταβλητές χαρακτηρίζονται ως **καθολικές (global)**. Το μειονέκτημα είναι ότι περιορίζεται έτσι η ανεξαρτησία των υποπρογραμμάτων.
Περιορισμένη εμβέλεια: Αυτές οι μεταβλητές είναι **τοπικές (local),** ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν. Η περιορισμένη εμβέλεια απαιτεί όλες οι μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, να δηλώνονται σε αυτό το τμήμα.
96. Πότε οι μεταβλητές έχουν απεριόριστη εμβέλεια. Ποιο μειονέκτημα έχει η απεριόριστη εμβέλεια. (σελ 113)
ΑΠ: Όλες οι μεταβλητές είναι ορατές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος, ανεξάρτητα από το πού δηλώθηκαν. Αυτού του τύπου οι μεταβλητές χαρακτηρίζονται ως **καθολικές (global)**. Το μειονέκτημα είναι ότι περιορίζεται έτσι η ανεξαρτησία των υποπρογραμμάτων.
97. Πότε οι μεταβλητές έχουν περιορισμένη εμβέλεια. Τι απαιτεί η περιορισμένη εμβέλεια. (σελ 113)
ΑΠ: Αυτές οι μεταβλητές είναι **τοπικές (local),** ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν. Η περιορισμένη εμβέλεια απαιτεί όλες οι μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, να δηλώνονται σε αυτό το τμήμα.
98. Ποιες μεταβλητές ονομάζονται τοπικές. (σελ 113 – 114)
ΑΠ: Οι μεταβλητές που έχουν οριστεί μέσα στο σώμα της συνάρτησης, έχουν τοπική εμβέλεια.
99. Ποια είναι η σχέση μεταξύ τοπικών μεταβλητών και συνάρτησης. (σελ 113 – 115)
ΑΠ: Όταν δηλώνουμε μεταβλητές κατά τον ορισμό μέσα στο σώμα μιας συνάρτησης, αυτές δεν έχουν καμία σχέση με άλλες μεταβλητές που έχουν την ίδια ονομασία και χρησιμοποιούνται έξω από αυτήν τη συνάρτηση.
100. Ποιες μεταβλητές ονομάζονται καθολικές. (σελ 113 – 115)
ΑΠ: Μεταβλητές που έχουν οριστεί έξω από το σώμα μιας συνάρτησης και είναι προσβάσιμες από όλες τις συναρτήσεις.
101. Πώς ορίζω μία μεταβλητή ως καθολική. (σελ 115)
ΑΠ: Οι μεταβλητές που δηλώνονται έξω από τις συναρτήσεις του προγράμματος, είναι καθολικές μεταβλητές και προσπελαύνονται από οποιοδήποτε σημείο μέσα στο πρόγραμμα.

102. Πώς μπορούμε να αλλάξουμε την τιμή μιας καθολικής μεταβλητής μέσα σε μία συνάρτηση. (σελ 115 – 116)

ΑΠ: Εάν θέλουμε μέσα σε μια συνάρτηση να αλλάξουμε την τιμή μιας καθολικής μεταβλητής, δηλαδή μιας μεταβλητής η οποία ορίζεται στο κορυφαίο επίπεδο του προγράμματος, τότε πρέπει να δηλώσουμε στην Python ότι η μεταβλητή αυτή δεν είναι τοπική αλλά καθολική. Αυτό γίνεται με τη χρήση της εντολής **global**, με την οποία γίνεται ξεκάθαρο ότι η μεταβλητή βρίσκεται σε ένα εξωτερικό τμήμα εντολών.

103. Ποια είναι η χρήση της εντολής global στην Python. (σελ 115 – 116)

ΑΠ: Η χρήση της εντολής global είναι η εξής:

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

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

104. Ποιες είναι οι διαφορές μεταξύ τοπικών και Καθολικών μεταβλητών στην Python. (σελ 113 – 116)

ΑΠ: Οι μεταβλητές που έχουν οριστεί μέσα στο σώμα της συνάρτησης, έχουν τοπική εμβέλεια, ενώ αυτές που ορίστηκαν έξω από αυτό έχουν καθολική εμβέλεια. Αυτό σημαίνει ότι οι τοπικές μεταβλητές μπορούν να προσπελαστούν μόνο μέσα στη συνάρτηση στην οποία δηλώθηκαν, ενώ οι καθολικές μεταβλητές μπορεί να είναι προσβάσιμες από όλες τις συναρτήσεις.

ΚΕΦΑΛΑΙΟ 8

105. Τι είναι μία δομή δεδομένων. (σελ 126)

ΑΠ: Μία δομή δεδομένων είναι ένα σχήμα οργάνωσης των δεδομένων, με άλλα λόγια ένα σύνολο αποθηκευμένων δεδομένων πάνω στο οποίο μπορεί να πραγματοποιηθεί ένα σύνολο ενεργειών.

106. Γιατί είναι σημαντική η δομή δεδομένων. (Φυλλάδιο)

ΑΠ: Η επιλογή της κατάλληλης δομής δεδομένων μπορεί να επιφέρει μεγάλα οφέλη στην ταχύτητα εκτέλεσης των υπολογισμών, άρα και στην ταχύτητα εκτέλεσης ενός προγράμματος.

107. Ποιες κατηγορίες δομών δεδομένων υπάρχουν. (Φυλλάδιο)

ΑΠ: Οι δομές δεδομένων χωρίζονται σε 2 μεγάλες κατηγορίες:

i. Τις Μη μεταβαλλόμενες (ή στατικές – immutable) δομές δεδομένων.

ii. Τις δυναμικές δομές δεδομένων (mutabe).

108. Ποια είναι τα χαρακτηριστικά των μη μεταβαλλόμενων (ή στατικών – immutable) δομών δεδομένων. (σελ 126 και φυλλάδιο)

ΑΠ: Οι μη μεταβαλλόμενες δομές δεδομένων έχουν σταθερό μέγεθος που καθορίζεται κατά τη συγγραφή του προγράμματος και για την αναπαράσταση της δομής η μνήμη που δεσμεύεται είναι σταθερή. Κατά συνέπεια τα στοιχεία της δομής είναι μη μεταβαλλόμενα στοιχεία. Αυτό σημαίνει ότι δεν μπορούμε να αφαιρέσουμε ή να προσθέσουμε αντικείμενα σε αυτές, κατά τη διάρκεια εκτέλεσης του προγράμματος.

109. Ποια είναι τα χαρακτηριστικά των δυναμικών (mutabe) δομών δεδομένων. (σελ 130 και φυλλάδιο)

ΑΠ: Οι δυναμικές δομές δεδομένων στηρίζονται στη λειτουργία της «δυναμικής παραχώρησης μνήμης», κατά την οποία το πρόγραμμα μπορεί να ζητήσει όση μνήμη απαιτείται για τη δημιουργία και επεξεργασία της δομής κατά την εκτέλεση του

προγράμματος. Δεν έχουν σταθερό μέγεθος αλλά είναι δυνατό να προσθέτουμε και να αφαιρούμε στοιχεία από αυτές κατά τη διάρκεια εκτέλεσης του προγράμματος

110. Ποιες στατικές δομές δεδομένων υπάρχουν στην Python. (σελ 127 και φυλλάδιο)

ΑΠ: Στην Python στατικές δομές είναι οι συμβολοσειρές και οι πλειάδες

111. Ποιες δυναμικές δομές δεδομένων υπάρχουν στην Python. (Φυλλάδιο)

ΑΠ: Στην Python δυναμικές δομές είναι οι λίστες, η στοίβα, η ουρά, τα λεξικά, οι γράφοι και τα δέντρα.

112. Τι είναι οι συμβολοσειρές στην Python. (σελ 127)

ΑΠ: Τα αλφαριθμητικά ή συμβολοσειρές στην Python είναι ακολουθίες από χαρακτήρες που έχουν σταθερό μέγεθος και μη μεταβαλλόμενα περιεχόμενα. Δηλαδή, δεν μπορούμε να προσθέσουμε ή να αφαιρέσουμε χαρακτήρες, ούτε να τροποποιήσουμε τα περιεχόμενα του αλφαριθμητικού

113. Γιατί οι συμβολοσειρές ανήκουν στις στατικές δομές δεδομένων. (σελ 127 και φυλλάδιο)

ΑΠ: Οι συμβολοσειρές έχουν σταθερό μέγεθος. Δεν μπορούμε να προσθέσουμε ή να αφαιρέσουμε χαρακτήρες σε μια συμβολοσειρά από τη στιγμή που θα δημιουργηθεί, ούτε να τροποποιήσουμε το περιεχόμενό της. Μπορούμε μόνο να τη δημιουργήσουμε ξανά με τις αλλαγές που επιθυμούμε.

1**14. Πώς δημιουργείται μία συμβολοσειρά. (σελ 127)**

ΑΠ:

115. Πώς αποτυπώνεται μία συμβολοσειρά. (σελ 127 και φυλλάδιο)

ΑΠ: Κάθε χαρακτήρας μιας συμβολοσειράς απαριθμείται με ένα αριθμό, με τον πρώτο χαρακτήρα να ξεκινάει με το 0. Η προσπέλαση σε κάθε χαρακτήρα (στοιχείο) της συμβολοσειράς γίνεται γράφοντας το όνομα της μεταβλητής και μέσα σε αγκύλες τον αριθμό της θέσης που αντιστοιχεί στον χαρακτήρα. Ο αριθμός μέσα στην αγκύλη ονομάζεται δείκτης και δηλώνει τη θέση του στοιχείου σε σχέση με τη συμβολοσειρά. Είναι δυνατό να έχουμε και αρνητικούς δείκτες οι οποίοι ξεκινούν από το τέλος της συμβολοσειράς με τον τελευταίο χαρακτήρα να λαμβάνει την τιμή -1.

Αν έχουμε μια μεταβλητή w στην οποία αποθηκεύουμε τη λέξη “ΠΑΝΕΛΛΑΔΙΚΕΣ”, τότε σχηματικά θα είχαμε την εξής αναπαράσταση (αποτύπωση):

w = “ΠΑΝΕΛΛΑΔΙΚΕΣ”

ΠΡΟΣΟΧΗ: Ο δείκτης δηλώνει τη θέση του στοιχείου σε σχέση με τη συμβολοσειρά, η οποία θέση είναι διαφορετική από τη θέση του χαρακτήρα μέσα στη συμβολοσειρά. Ο χαρακτήρας “Δ” είναι ο 8ος χαρακτήρας της λέξης “ΠΑΝΕΛΛΑΔΙΚΕΣ”, αλλά στη συμβολοσειρά w για να αναφερθούμε στον χαρακτήρα αυτό θα χρησιμοποιήσουμε το δείκτη 7 (ή το -5)

116. Ποιος είναι ο τύπος δεδομένων μιας συμβολοσειράς στην Python. (σελ 127)

ΑΠ: Ο τύπος μιας συμβολοσειρας βρίσκεται με την εντολή type και είναι str, από τα αρχικά της λέξης string.

117. Ποια η λειτουργία της συνάρτησης len() στην Python. (σελ 127)

ΑΠ: Η συνάρτηση len επιστρέφει το μήκος, δηλαδή το πλήθος των χαρακτήρων του αλφαριθμητικού

118. Ποιο είναι το αποτέλεσμα της εφαρμογής του τελεστή της πρόσθεσης (+) σε συμβολοσειρές. (σελ 127)

ΑΠ: Ο τελεστής + όταν εφαρμόζεται σε αντικείμενα τύπου string, έχει σαν αποτέλεσμα τη συνένωσή τους σε μια συμβολοσειρά

119. Ποια είναι η λειτουργία του υπαρξιακού τελεστή in. (σελ 128)

ΑΠ: Ο υπαρξιακός τελεστής in ελέγχει αν ένα αντικείμενο ανήκει σε ένα σύνολο αντικειμένων.

120. Ποια είναι η χρήση του υπαρξιακού τελεστή in στις συμβολοσειρές. (σελ 128-129)

ΑΠ: Η χρήση του υπαρξιακού τελεστή in στις συμβολοσειρές είναι:

- Σάρωση των χαρακτήρων μιας συμβολοσειράς
- Καταμέτρηση φωνηέντων μιας φράσης

121. Ποια είναι η χρήση του τελεστή διαμέρισης (:) στις συμβολοσειρές. (σελ 138)

ΑΠ: Ο τελεστής διαμέρισης μπορεί να μας επιστρέψει ένα τμήμα μιας συμβολοσειράς.

122. Πώς γίνεται η σάρωση των χαρακτήρων μιας συμβολοσειράς. (σελ 128)

ΑΠ: Μια συμβολοσειρά είναι μια ακολουθία (sequence) από αντικείμενα, τα οποία, στην προκειμένη περίπτωση, είναι χαρακτήρες. Αν θέλουμε να σαρώσουμε τα αντικείμενα αυτά ένα-ένα, μπορούμε να το κάνουμε με μια εντολή επανάληψης for. Δείτε την παρακάτω συνάρτηση.

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

123. Τι είναι μία λίστα στην Python. (σελ 130)

ΑΠ: Η λίστα είναι μια διατεταγμένη ακολουθία αντικειμένων, όχι απαραίτητα του ίδιου τύπου και αποτελεί τη βασική δομή δεδομένων της Python.

124. Γιατί η λίστα είναι μία δυναμική δομή δεδομένων. (σελ 130 και φυλλάδιο)

ΑΠ: Η λίστα, σε αντίθεση με τη συμβολοσειρά, είναι μια δυναμική δομή στην οποία μπορούμε να προσθέτουμε ή να αφαιρούμε στοιχεία (mutable). Κάθε αντικείμενο της λίστας χαρακτηρίζεται από ένα μοναδικό αύξοντα αριθμό, ο οποίος ορίζει τη θέση του στη λίστα, ενώ η προσπέλαση στα στοιχεία της λίστας γίνεται όπως στις συμβολοσειρές, με το όνομα της λίστας και τον αύξοντα αριθμό του αντικείμενου μέσα σε αγκύλες.

125. Πώς δημιουργείται μία λίστα. (σελ 130)

ΑΠ: Μια λίστα δημιουργείται γράφοντας μια μεταβλητή, μετά την εντολή εκχώρησης και στη συνέχεια τα στοιχεία της λιστας μέσα σε [ ] χωρισμένα με κόμμα. Παράδειγμα η εντολή L = [3, 5, 8, 13, 21, 34] δημιουργεί τη μεταβλητή L που αναφέρεται στη λίστα [3, 5, 8, 13, 21, 34]

126. Πώς αποτυπώνεται μία λίστα. (σελ 130 και φυλλάδιο)

ΑΠ: Η λίστα είναι μια διατεταγμένη ακολουθία δεδομένων. Η αναπαράσταση μιας λίστας μοιάζει αρκετά με την συμβολοσειρά, με τη διαφορά ότι τα στοιχεία μιας λίστας μπορούν να είναι και διαφορετικού τύπου και για αυτό χαρακτηρίζονται ως αντικείμενα.

Κάθε αντικείμενο μιας λίστας χαρακτηρίζεται από ένα μοναδικό αύξοντα αριθμό, ο οποίος ορίζει τη θέση του μέσα στη λίστα, με το πρώτο αντικείμενο να ξεκινάει με το 0. Η προσπέλαση σε κάθε αντικείμενο της λίστας, όπως και στις συμβολοσειρές, γίνεται γράφοντας το όνομα της μεταβλητής και μέσα σε αγκύλες τον αριθμό της θέσης που αντιστοιχεί στο αντικείμενο. Ο αριθμός μέσα στην αγκύλη που δηλώνει τη θέση του στοιχείου ονομάζεται δείκτης. Είναι δυνατό να έχουμε και αρνητικούς δείκτες οι οποίοι ξεκινούν από το τέλος της λίστας με το τελευταίο αντικείμενο να λαμβάνει την τιμή -1.

Αν έχουμε μια μεταβλητή nums στην οποία αποθηκεύουμε μια λίστα αριθμών, όπως παρακάτω, τότε σχηματικά θα είχαμε την εξής αναπαράσταση (αποτύπωση):

nums = [3,6,2,7,9,1]

ΠΡΟΣΟΧΗ: Ο δείκτης δηλώνει τη θέση του αντικειμένου, η οποία θέση είναι διαφορετική από τη θέση του στοιχείου μέσα στη λίστα. Ο αριθμός 9 είναι ο 5ος αριθμός μέσα στη λίστα, αλλά στη λίστα nums για να αναφερθούμε στον αριθμό αυτό αυτό θα χρησιμοποιήσουμε το δείκτη 4 (ή το -2).

127. Πώς γίνεται η προσπέλαση ενός στοιχείου μιας λίστας. (σελ 130 και 133-134)

ΑΠ: Για την προσπέλαση ενός στοιχείου της λίστας, αρχικά δηλώνουμε το όνομα της λίστας και στη συνέχεια μέσα σε [ ] γράφουμε την θέση του στοιχείου στην λίστα πχ L[1] δηλώνουμε το δεύτερο στοιχείο της λίστας L.

128. Τι είναι η άδεια λίστα. (Φυλλάδιο)

ΑΠ: Η άδεια λίστα είναι μια λίστα η οποία δεν περιέχει στοιχεία μέσα της και συμβολίζεται με 2 άδειες αγκύλες [ ]. Πχ lista=[ ]

Αν θέλουμε να ελέγξουμε αν μια λίστα είναι κενή υπάρχουν 2 τρόποι:

i. Να ελέγξουμε απευθείας αν η λίστα είναι κενή: if L == [ ]:

ii. Να ελέγξουμε αν το μέγεθος της λίστας είναι ίσο με το 0: if len(L) == 0:

129. Πώς προσθέτουμε ένα στοιχείο στο τέλος μιας λίστας. (σελ 131)

ΑΠ: Αν θέλουμε να προσθέσουμε ένα στοιχείο στο τέλος μιας λίστας, γράφουμε:

Λίστα = Λίστα + [στοιχείο]

130. πώς προσθέτουμε ένα στοιχείο στην αρχή μιας λίστας. (σελ 131)
ΑΠ: Αν θέλουμε να προσθέσουμε ένα στοιχείο στην αρχή μιας λίστας, γράφουμε
Λίστα = [στοιχείο] + Λίστα
131. Ποιο είναι το αποτέλεσμα της εφαρμογής του τελεστή της πρόσθεσης (+) σε λίστες.(σελ 132)
ΑΠ: Θα προκύψει μια νέα λίστα με στοιχεία όλων των λιστών ξεκινώντας από τα στοιχεία της πρώτης λίστας, μετά της δεύτερης κοκ.
132. Ποια είναι η χρήση του υπαρξιακού τελεστή in στις λίστες. (σελ 132)
ΑΠ: item in List: επιστρέφει True, αν το στοιχείο item υπάρχει μέσα στη λίστα List, αλλιώς επιστρέφει False.
133. Ποια είναι η χρήση του not in στις λίστες. (σελ 132)
ΑΠ: item not in List: επιστρέφει True, αν το στοιχείο item δεν υπάρχει μέσα στη λίστα List, αλλιώς, αν υπάρχει επιστρέφει False.
134. Πώς εφαρμόζεται η συνάρτηση len() σε μία λίστα. (σελ 132)
ΑΠ: len (List): Επιστρέφει το πλήθος των στοιχείων (ή μέγεθος) της λίστας.
135. Πώς λειτουργεί η συνάρτηση list(). Σε ποια άλλα είδη δομών μπορεί να χρησιμοποιηθεί. . (σελ 132)
ΑΠ: list (String): Επιστρέφει μια λίστα με στοιχεία τους χαρακτήρες της συμβολοσειράς string. Η συνάρτηση αυτή μπορεί να μετατρέψει και άλλα είδη δομών σε λίστα, όπως είναι οι πλειάδες και τα λεξικά.
136. Ποια είναι η χρήση του τελεστή διαμέρισης (:) στις λίστες. (σελ 138)
ΑΠ:ς. Ο τελεστής διαμέρισης μπορεί να μας επιστρέψει ένα τμήμα μιας λίστας.Η έκφραση word[a : b] μας επιστρέφει το τμήμα της λίστας από το στοιχείο word[a] μέχρι και το στοιχείο word[b-1].
137. Γιατί ο τελεστής διαμέρισης (:) είναι πολύ σημαντικός στην επεξεργασία λιστών. (σελ139)
ΑΠ: Ο τελεστής : είναι πολύ σημαντικός στην επεξεργασία λιστών στην Python, γιατί μπορούμε να τον χρησιμοποιήσουμε για να πάρουμε ένα αντίγραφο μιας λίστας.
138. Ποια είναι η λειτουργία της μεθόδου append() στην Python. (σελ 133)
ΑΠ: L.append(object): Προσθέτει το στοιχείο object στο τέλος της λίστας L.
139. Ποια είναι η λειτουργία της μεθόδου insert() στην Python. (σελ 133)
ΑΠ: L.insert(index, object): Προσθέτει το στοιχείο object, στη θέση index της λίστας L, μετακινώντας όλα τα στοιχεία από τη θέση index και μετά, κατά μία θέση.
140. Ποια είναι η λειτουργία της μεθόδου pop() στην Python. (σελ 133)
ΑΠ: L.pop([index]): Αφαίρεση από τη λίστα του στοιχείου που βρίσκεται στη θέση index. Αν δεν δοθεί θέση, τότε θα αφαιρεθεί το τελευταίο στοιχείο της λίστας.
141. Πώς γίνεται η διάσχιση των στοιχείων μιας λίστας. (σελ 133-134)
ΑΠ: Για να επεξεργαστούμε τα στοιχεία μιας λίστας, ένα κάθε φορά, κάνοντουμε χρήση του παρακάτω ιδιώματος της δομής επανάληψης for:

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

142. Πώς βρίσκουμε τον μέσο όρο των στοιχείων μιας λίστας. (σελ 134)
ΑΠ: Μέσος όρος των στοιχείων μιας λίστας

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

Μέγιστη τιμή σε μια λίστα

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

143. Ποιο είναι το πλεονέκτημα του αλγορίθμου συγχώνευσης διατεταγμένων λιστών. (σελ 136)

ΑΠ: Ο αλγόριθμος αξιοποιεί το γεγονός ότι οι αρχικές λίστες είναι ήδη ταξινομημένες, ώστε να μη χρειαστεί να ταξινομήσει από την αρχή την τελική λίστα, κάτι το οποίο έχει σημαντικό υπολογιστικό κόστος για πολύ μεγάλες λίστες.

```
def merge( A, B ) :
    L = [ ]
    # όσο οι δυο λίστες έχουν στοιχεία
    while A != [ ] and B != [ ] :
        if A[0] < B[0] : # Αν το 1ο στοιχείο της Α είναι το μικρότερο
            L.append( A.pop(0) ) # το μεταφέρουμε στο τέλος της L
        else :
            L.append( B.pop(0) )
            # αλλιώς μεταφέρουμε το πρώτο στοιχείο της Β στην L
    return L + A + B
    # στο τέλος προσθέτουμε τα στοιχεία που έχουν μείνει
```

ΚΕΦΑΛΑΙΟ 5

144. Ποιο μειονέκτημα παρουσιάζει ο αλγόριθμος της σειριακής αναζήτησης. (σελ 63)

ΑΠ: Ο αλγόριθμος αυτός στην χειρότερη περίπτωση, όπου το στοιχείο δεν υπάρχει στο σύνολο δεδομένων ή είναι το τελευταίο στη σειρά, θα ελέγξει όλα τα στοιχεία του συνόλου στο οποίο ψάχνουμε.

145. Πότε μπορεί να χρησιμοποιηθεί η δυαδική αναζήτηση. Γιατί ονομάζεται έτσι. (σελ 68 και 69)

ΑΠ: Μπορεί να χρησιμοποιηθεί όταν η αναζήτηση γίνεται σε ταξινομημένα στοιχεία πχ μιας λίστας. Ονομάστηκε αλγόριθμος της δυαδικής αναζήτησης, γιατί σε κάθε βήμα μειώνει το χώρο αναζήτησης στο μισό.

146. Τι ονομάζεται πολυμορφισμός στην Python. (σελ 72)

ΑΠ: Πολυμπορφισμός είναι το χαρακτηριστικό της Python όπου οι αλγόριθμοι δυαδικής αναζήτησης ισχύουν για όλους τους τύπους δεδομένων για τους οποίους ορίζονται οι συγκριτικοί τελεστές ==, < . Δηλαδή, μπορούν να χρησιμοποιηθούν για ακέραιους ή πραγματικούς αριθμούς, αλφαριθμητικά ή ακόμα και σύνθετους τύπους για τους οποίους έχουμε ορίσει τους συγκεκριμένους τελεστές

147. Ποια είναι τα χαρακτηριστικά της δυαδικής αναζήτησης. (σελ 72)

ΑΠ: Μπορούμε να πούμε ότι η δυαδική αναζήτηση:

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

148. Ποιο είναι το χαρακτηριστικό του αλγόριθμου ταξινόμησης με επιλογή. (σελ 72)

ΑΠ: Ένα από τα χαρακτηριστικά αυτού του αλγορίθμου είναι ότι εκτελεί πάντα τον ίδιο αριθμό συγκρίσεων για συλλογές δεδομένων με το ίδιο μέγεθος, ακόμα και για αυτές που είναι ήδη ταξινομημένες.

149. Με ποια εντολή μπορούμε να αντιμετωπίσουμε το περιεχόμενο των μεταβλητών a και b. (σελ 74)

ΑΠ: a,b=b,a

150. Να γράψετε ισοδύναμο τμήμα εντολών με το a,b = b,a. (σελ 74)

ΑΠ: temp=a, a=b, b=temp

151. Που βασίζεται ο αλγόριθμος ταξινόμησης ευθείας ανταλλαγής. (σελ 78)

ΑΠ: Ο αλγόριθμος βασίζεται στη σύγκριση και αντιμετάθεση ζευγών που δεν ακολουθούν τη διάταξη της ταξινόμησης

152. Πόσο γρήγορος θεωρείται ο αλγόριθμος ευθείας ανταλλαγής και ποιο είναι το βασικό του πλεονέκτημα. (σελ 78)
ΑΠ: Ο αλγόριθμος ευθείας ανταλλαγής, αν και θεωρείται από τους πιο αργούς αλγορίθμους ταξινόμησης, έχει ένα πολύ σημαντικό πλεονέκτημα: ότι μπορεί να τροποποιηθεί, ώστε να τερματίσει μόλις διαπιστώσει ότι η λίστα έχει ταξινομηθεί

153. Να αναπτύξετε τη βελτιωμένη έκδοση του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής η οποία τερματίζει , όταν διαπιστώσει ότι η λίστα είναι ταξινομημένη, ώστε να αποφεύγονται οι περιττές συγκρίσεις. (Φυλλάδιο).

Έξυπνη Φυσαλίδα
```
def smartbubblesort(lista):
    n=len(lista)
    isSorted=False
    i=0
    while i<n and isSorted==False: #Εναλλακτικά ☾ not isSorted
        isSorted=True
        for j in range(n-1,i,-1):
            if lista[j]<lista[j-1]:
                lista[j],lista[j-1]=lista[j-1],lista[j]
                isSorted=False
        i=i+1
```

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. (Δραστηριότητα 3 σελ 86)
Τρόπος Α
```
def check(lista):
    n=len(lista)
    taxinom=True
    for j in range(1,n-1):
        if lista[j]<lista[j-1]:
            taxinom=False
    return taxinom
```

Τρόπος Β
```
def check (lista):
    taxinom = True
    i = 0
    N = len(lista)-1
    while taxinom and i<N :
        if lista[i] > lista[i+1]:
            taxinom = False
        i = i +1
    return taxinom
```