ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αλγόριθμοι με Python

Φοιτητές

Ιωάννα-Ευθυμία Αθανασίου

Βασίλειος Τσιμπούρης

Εποπτεύων Καθηγητής

Κωνσταντίνος Στάμος

Πάτρα-2019

Πρόλογος

Θα ήθελα να ευχαριστήσω θερμά τον Επιβλέποντα Καθηγητή μου κύριο Κωνσταντίνο Στάμο για την εμπιστοσύνη που μας έδειξε και την καθοδήγησή του στο να ολοκληρωθεί η παρούσα πτυχιακή. Επίσης θέλω να τον ευχαριστήσω για την βοήθεια που μας παρείχε σε όλη αυτή την προσπάθεια προς την υλοποίηση της πτυχιακής μας εργασίας.

Θα ήθελα επισης να ευχαριστήσω τους γονείς μου οι οποίοι στήριξαν τις σπουδές μου φροντίζοντας για την καλύτερη δυνατή μόρφωσή μου

Και τ□ λοφα δώσω τις ευχαριστίες μου στον καλό μου φίλο Γιώργο για την πολύτιμη συμπαράστασή του και τον συμφοιτητή μου Βασίλη Τσιμπούρη για την εξαιρετική συνεργασιας μας

Ιωάννα-Ευθυμία Αθανασίου

Για την ολοκλήρωση αυτής της πτυχιακής εργασίας χρειαστήκαμε την βοήθεια διάφορων ατόμων. Πρώτα απ' όλα του Επιβλέποντα Καθηγητή της πτυχιακής κύριο Κωνσταντίνο Στάμο όπου με τη δική του καθοδήγηση και τις συμβουλές του μας βοήθησε να φτάσουμε σε ένα άρτιο αποτέλεσμα.

Στη συνέχεια θα ήθελα να ευχαριστήσω τη συμφοιτήτριά και φίλη μου Ιωάννα-Ευθυμία Αθανασίου για την άριστη συνεργασία και την υπομονή της όλο αυτό το χρονικό διάστημα

Στο τέλος θέλω να ευχαριστήσω τους φίλους και την οικογένεια μου που από τη πρώτη στιγμή που ξεκίνησα τις σπουδές μου ήταν διπλά μου και με στήριξαν σε ότι και αν χρειάστηκα.

Βασίλειος Τσιμπουρης

Περίληψη

Στη παρούσα πτυχιακή εργασία αναλύεται η γλώσσα προγραμματισμού Python και η χρήση της στον εκπαιδευτικό τομέα για την εκμάθηση αλγορίθμων. Αρχικά, γίνεται μία αναφορά στις βασικές λειτουργίες της γλώσσας, στα χαρακτηριστικά της και στη δομή της και παραθέτουμε ορισμούς και παραδείγματα για την κατανόηση της. Ταυτόχρονα, κάνουμε μία σύγκριση με τη γλώσσα προγραμματισμού C μιας και είναι η γλώσσα προγραμματισμού που διδάσκεται κατά κύριο λόγο στα αρχικά έτη ενός πανεπιστημιακού τμήματος πληροφορικής. Σκοπός της παρούσας πτυχιακής εργασίας είναι να δοθούν κάποιες κατευθυντήριες γραμμές για την Python και τον τρόπο χρήσης της ως εναλλακτική επιλογή για την εκμάθηση αλγορίθμων. Επιπρόσθετα, μετατρέπουμε —και παραθέτουμε σε αυτή την πτυχιακή— σε γλώσσα Python τις ασκήσεις εκμαθησης της γλώσσας C του τμήματος μας. Χρησιμοποιούμε διαδικασιακό και συναρτησιακό προγραμματισμό για να παρουσιάσουμε τους τρόπους επίλυσης των ασκήσεων, μιας και η γλώσσα C δεν υποστηρίζει αντικειμενοστραφή προγραμματισμό σε αντίθεση με την Python.

Περιεχόμενα

1. Εισαγωγή

Από τις διάφορες γλώσσες προγραμματισμού διδάσκονται σε όλα τα πανεπιστημιακά ιδρύματα του κόσμου, οι κυριότερες είναι η C, η Java και η Python. Εμείς, στην παρούσα πτυχιακή εργασία, μελετάμε το κατά πόσον μπορούν οι ασκήσεις που διδάσκονται σε γλώσσα C, να μετατραπούν στη γλώσσα Python με αποτέλεσμα που να τοποθετεί την Python ως αρχική γλώσσα εκμάθησης προγραμματισμού αντί ή παράλληλα με τη C. Σε αυτό το κεφάλαιο, ξεκινάμε με μία μικρή ιστορική αναδρομή των γλωσσών προγραμματισμού η οποία παρουσιάζεται στην Υποενότητα 1.1. Έπειτα συνεχίζουμε με την ιστορική αναδρομή της γλώσσας Python στην Υποενότητα 1.2 και καταλήγουμε στην Υποενότητα 1.3 με τα πλεονεκτήματα και τα μειονεκτήματα της.

1.1 Γλώσσες Προγραμματισμού

Η Augusta Ada King γνωστή ως Ada Lovelace ήταν μια Αγγλίδα μαθηματικός και συγγραφέας, γνωστή κυρίως για την εργασία της στο μηχανισμό γενικής χρήσης του Charles Babbage, τον Αναλυτικό Κινητήρα. Ήταν η πρώτη που αναγνώρισε ότι το μηχάνημα είχε εφαρμογές πέρα από τον καθαρό υπολογισμό και δημοσίευσε τον πρώτο αλγόριθμο που προοριζόταν να εκτελεστεί από μια τέτοια μηχανή το 1842. Ως αποτέλεσμα, θεωρείται ως η πρώτη που αναγνώρισε την πλήρη δυναμική μιας «υπολογιστικής μηχανής» και μία από τις πρώτες προγραμματίστριες υπολογιστών (Fuegi & Francis, 2003). Ειδικότερα στις σημειώσεις της περιγράφει έναν αλγόριθμο για τον υπολογισμό των αριθμών Bernoulli. Θεωρείται ότι είναι ο πρώτος δημοσιευμένος αλγόριθμος που έχει σχεδιαστεί ειδικά για την υλοποίηση σε έναν υπολογιστή. Προς τιμήν της το 1980 το Υπουργείο Αμύνης των ΗΠΑ παρουσίασε μια γλώσσα προγραμματισμού, την οποία και ονόμασε Ada (https://en.wikipedia.org/wiki/Ada_Lovelace).

Σχεδόν 100 χρόνια αργότερα το 1934 ο μαθηματικός Alan Turing εφηύρε μία μηχανή η οποία είναι ένα μαθηματικό μοντέλο υπολογισμού και ορίζεται ως μια αφηρημένη μηχανή που χειρίζεται τα σύμβολα σε μια ταινία σύμφωνα με ένα σύνολο κανόνων. Ουσιαστικά είναι μία μηχανή η οποία μπορεί να προσομοιώσει τη λογική ενός αλγορίθμου. Λίγο αργότερα το 1937 ο Turing ως μέρος της διδακτορικής του διατριβής δημιουργεί έναν ψηφιακό πολλαπλασιαστή όπου σηματοδότησε την απαρχή των ηλεκτρονικών υπολογιστών ως φυσικές οντότητες (Hodges, 2012).

Η πρώτη γλώσσα προγραμματισμού υψηλού επιπέδου ήταν η Plankalkül, που δημιουργήθηκε από τον Konrad Zuse μεταξύ του 1942 και του 1945 (Knuth & Pardo, 1980). Η πρώτη γλώσσα υψηλού επιπέδου που σχετίζεται και με μεταγλωττιστή δημιουργήθηκε από τον Corrado Böhm το 1951, για τη διδακτορική του διατριβή. Η πρώτη εμπορικά διαθέσιμη γλώσσα ήταν η FORTRAN (FORmula TRANslation) που αναπτύχθηκε το 1956 (το πρώτο εγχειρίδιο εμφανίστηκε το 1956, αλλά αναπτύχθηκε για πρώτη φορά το 1954) από μια ομάδα με επικεφαλής τον John Backus στην IBM.

Στα μέσα της δεκαετίας του 1950 αναπτύχθηκε η ALGOL (Algorithmic Language) που αποτελεί μια οικογένεια προστακτικών γλωσσών προγραμματισμού υπολογιστών η οποία επηρέασε σε μεγάλο βαθμό πολλές άλλες γλώσσες. Ήταν μια τυπική μέθοδος για την περιγραφή του αλγορίθμου που χρησιμοποιείται από την ACM σε εγχειρίδια και ακαδημαϊκές πηγές για περισσότερα από τριάντα χρόνια (Naur et al., 1976). Το 1967 αναπτύσσεται η γλώσσα BCPL (Basic Combined Programming Language) από τον Martin Richards η οποία είναι διαδικαστική γλώσσα προγραμματισμού (Richards & Whitby-Strevens, 1981) και είναι πρόγονος της γλώσσας B. Λίγα χρόνια νωρίτερα το 1963 αναπτύσσεται η γλώσσα CPL (Combined Programming Language) από τους Strachey et al. (Barron, 1963) η οποία είναι προστακτικού και διαδικαστικού προγραμματισμού και επηρέασε τη γλώσσα BCPL αποτελώντας παράλληλα πρόγονο της γνωστής γλώσσας C.

Η C είναι μια γλώσσα προγραμματισμού γενικού σκοπού που υποστηρίζει το δομημένο προγραμματισμό. Αναπτύχθηκε αρχικά στα Bell Labs από τον Dennis Ritchie μεταξύ 1972 και 1973 για να κάνει τα βοηθητικά προγράμματα να τρέχουν στο λειτουργικό σύστημα Unix. Αργότερα, εφαρμόστηκε για την εκ νέου εφαρμογή του πυρήνα του λειτουργικού συστήματος Unix (Ritchie, 1993). Η γλώσσα C ουσιαστικά έχει επηρεάσει τις περισσότερες γλώσσες προγραμματισμού οι οποίες χρησιμοποιούνται ευρέως σήμερα όπως οι C++, PHP, Java και Python, ανάμεσα σε πολλές άλλες.

1.2 Ιστορική αναδρομή της Python

Η ιστορία της γλώσσας Python χρονολογείται από τα τέλη της δεκαετίας του ’80. Η ιδέα συλλήφθηκε και άρχισε να υλοποιείται το Δεκέμβριο του 1989 από τον Guido van Rossum στο πανεπιστήμιο του CWI στην Ολλανδία. Η Python είναι η διάδοχη γλώσσα της ABC προγραμματιστικής γλώσσας η οποία ήταν ικανή να χειρίζεται εξαιρέσεις και διασυνδέσεις για το λειτουργικό σύστημα Amoeba. Ο Van Rossum είναι ο κύριος συγγραφέας της Python και έχει τον κεντρικό ρόλο και λόγο μέχρι και σήμερα στην κατεύθυνση της γλώσσας. Η Python χρησιμοποιείται ευρέως, είναι γενικού σκοπού και υψηλού επιπέδου γλώσσα προγραμματισμού.

Η σχεδιαστική φιλοσοφία δίνει έμφαση στην αναγνωσιμότητα του κώδικα και το συντακτικό του επιτρέπει στους προγραμματιστές να εκφράζουν το σχέδιο τους με λιγότερες γραμμές κώδικα απ’ ότι θα ήταν πιθανό με γλώσσες όπως οι C++ και η Java.

Η γλώσσα παρέχει δομές που προορίζονται για τη διευκόλυνση σαφών προγραμμάτων μικρής και μεγάλης κλίμακας. Η Python υποστηρίζει πολλαπλά προγραμματιστικά υποδείγματα συμπεριλαμβανομένου του αντικειμενοστραφή, διαδικαστικού και προστακτικού προγραμματισμού. Χαρακτηρίζεται από ένα δυναμικού τύπου σύστημα με αυτόματη διαχείριση μνήμης έχοντας ταυτόχρονα μια μεγάλη και κατανοητή τυπική βιβλιοθήκη. Υπάρχουν πάρα πολλοί διαθέσιμοι διερμηνείς για την Python για σχεδόν όλα τα λειτουργικά συστήματα επιτρέποντας σε αυτήν την εκτέλεση κώδικα. Η Python υποστηρίζεται χρησιμοποιώντας πολλές επεκτάσεις και βιβλιοθήκες για τις διάφορες ανάγκες που έχουν οι χρήστες της.

Η Python χρησιμοποιεί ένα δυναμικό σύστημα ελέγχου, δηλαδή κατά μία έννοια ελέγχει τον κώδικα κατά τη διάρκεια εκτέλεσης του προγράμματος. Επίσης χρησιμοποιεί ένα συνδυασμό καταμέτρησης αναφορών και συλλογής απορριμμάτων, δηλαδή απελευθερώνει τη μνήμη από αντικείμενα τα οποία δε χρησιμοποιούνται. Ένα σημαντικό χαρακτηριστικό της Python είναι η δυναμική διαχείριση ονομάτων η οποία δεσμεύει τα ονόματα μεθόδων και μεταβλητών κατά την εκτέλεση του προγράμματος. Ταυτόχρονα η γλώσσα περιέχει έννοιες όπως οι λίστες, τα λεξικά (dictionaries), τα σύνολα και οι εκφράσεις γεννητριών.

Η τυπική βιβλιοθήκη έχει δύο μονάδες, τα itertools και functools. Τα itertools είναι μονάδες οι οποίες παρέχουν εργαλεία για να μπορούμε να χρησιμοποιήσουμε επαναληπτικές διαδικασίες, για παράδειγμα σε λίστες. Η ενότητα functools παρέχει εργαλεία για να προσαρμόσουμε ή να επεκτείνουμε ήδη υπάρχουσες συναρτήσεις και αντικείμενα για νέους σκοπούς χωρίς να τα ξαναγράψουμε από την αρχή. Η Python έχει μία μεγάλη τυπική βιβλιοθήκη και αυτό είναι από τα μεγαλύτερα πλεονεκτήματα της, αφού παρέχει εργαλεία για πολλές εργασίες. Επίσης παρέχει πρότυπα και πρωτόκολλα για εφαρμογές διαδικτύου όπως τα MIME (Multipurpose Internet Mail Extensions) και HTTP (Hypertext Transfer Protocol). Ταυτόχρονα έχει μονάδες για τη δημιουργία διεπαφών, μονάδες για τη σύνδεση σχεσιακών βάσεων δεδομένων, γεννήτριες ψευδό-τυχαίων αριθμών, αριθμητική με αυθαίρετη ακρίβεια στους δεκαδικούς και μονάδες για τη χειραγώγηση τυπικών εκφράσεων. Κάποια μέρη της τυπικής βιβλιοθήκης καλύπτονται από προδιαγραφές, εσωτερική τεκμηρίωση και κώδικα. Ωστόσο, υπάρχουν μονάδες της τυπικής βιβλιοθήκης οι οποίες πρέπει να επαναγραφούν ως εναλλακτικές υλοποιήσεις διότι μπορεί να μην υποστηρίζονται από διάφορες πλατφόρμες. Η υλοποίηση της Python γίνεται από την κοινότητα προγραμματιστών που την υποστηρίζει και αυτή πραγματοποιείται σε ένα αποθετήριο πηγαίου κώδικα που χρησιμοποιεί το Mercurial. Μέχρι και σήμερα η κοινότητα έχει προσφέρει πάνω από 72000 μονάδες λογισμικού στην τυπική βιβλιοθήκη.

Το όνομα της Python έχει προέρθει από την τηλεοπτική σειρά Monty Python's Flying Circus. Από το 2003 η Python σταθερά κατατάσσεται στις δέκα πιο γνωστές γλώσσες προγραμματισμού. Η Python είναι μία σταθερή γλώσσα και χρησιμοποιείται σε διάφορα έργα ως γλώσσα βασικού προγραμματισμού. Χρησιμοποιείται επίσης ευρέως για τη δημιουργία πρωτοτύπων σε μελλοντικά προγράμματα (https://en.wikipedia.org/wiki/Python_(programming_language)).

1.3 Πλεονεκτήματα και μειονεκτήματα

Η Python έχει σημαντικά πλεονεκτήματα έναντι άλλων προγραμματιστικών γλωσσών όπως για παράδειγμα:

- καθαρό συντακτικό,

- είναι ανεκτική στα σφάλματα, ιδιότητα που επιτρέπει σε ένα σύστημα να συνεχίσει να λειτουργεί σωστά σε περίπτωση βλάβης ορισμένων (ενός ή περισσοτέρων σφαλμάτων εντός) των λειτουργιών της,
- η κανονική διανομή έχει πολλές χρήσιμες ενότητες (συμπεριλαμβανομένης της ενότητας για την ανάπτυξη GUI),
- η χρήση της Python γίνεται δια-δραστικά για τη λειτουργία και την επίλυση απλών προβλημάτων,
- η κανονική διανομή είναι απλή αλλά ταυτόχρονα έχει αρκετά ισχυρό περιβάλλον ανάπτυξης, το οποίο ονομάζεται IDLE,
- είναι κατάλληλη για την επίλυση μαθηματικών προβλημάτων, δηλαδή μπορεί κάποιος να εργαστεί με σύνθετους αριθμούς και με αυθαίρετο μέγεθος ακέραιων και να χρησιμοποιηθεί σαν ισχυρή αριθμομηχανή.

Ωστόσο, η Python εξακολουθεί να έχει ορισμένα μειονεκτήματα. Η Python, όπως και πολλές άλλες γλώσσες προγραμματισμού, όπως για παράδειγμα οι μεταγλωττιστές JIT, έχουν ένα κοινό μειονέκτημα - το σχετικά χαμηλό ποσοστό εκτέλεσης του προγράμματος.

2. Βασικά στοιχεία της Python

Σε αυτό το κεφάλαιο αναλύουμε τα βασικά δομικά στοιχεία τη γλώσσας Python όπως δομές ελέγχου-επανάληψης και δεδομένων. Ταυτόχρονα, όπου είναι απαραίτητο συμπληρώνουμε με παραδείγματα για να είναι πιο εύκολα κατανοητά.

2.1 Στοίχιση κώδικα

Στη συγγραφή κώδικα με Python πρέπει να προσέχουμε τους χαρακτήρες διαστήματος. Αν δεν είναι σωστά στοιχισμένος ο κώδικας, ο μεταγλωττιστής παράγει το επόμενο μήνυμα σφάλματος Indentation Error : expected an indented block. Αυτό σημαίνει ότι δεν έχουμε τοποθετήσει σωστά τους χαρακτήρες διαστήματος. Μια καλή πρακτική που συνιστάται είναι να χρησιμοποιούμε tabs ή τέσσερα κενά για μπλοκ εντολών. Η στοίχιση χρησιμοποιείται στην Python για να είναι ξεκάθαρο σε ποιο μπλοκ ανήκουν οι εντολές που γράφουμε. Πιο πρακτικά, μετά από κάθε άνω κάτω τελεία πρέπει χρησιμοποιούμε την επιτρεπόμενη στοίχιση.

2.2 Μεταβλητές και Τελεστές

Μεταβλητές είναι ένας τρόπος για να αποθηκεύουμε δεδομένα. Στην Python δεν δηλώνουμε ρητά τι τύπος δεδομένων χρησιμοποιείται όπως για παράδειγμα στη C. Επομένως, για να δηλώσουμε μία μεταβλητή μπορούμε να το κάνουμε ως εξής, τα ονόματα των μεταβλητών πρέπει να περιέχουν γράμματα του Αγγλικού αλφαβήτου είτε πεζά είτε κεφαλαία, το χαρακτήρα '_' (δηλαδή κάτω παύλα), και όλους τους αριθμούς από '0-9' και το όνομα μπορεί να αρχίζει μόνο με γράμμα ή '_' ή συνδυασμός των δύο. Οι μεταβλητές μπορούν να αποθηκεύσουν αριθμητικές τιμές όπως ακέραιους, πραγματικούς και μιγαδικούς, αλφαριθμητικές εκφράσεις (strings) και λογικές τιμές όπως True ή False.

Επίσης η Python χρησιμοποιεί λέξεις κλειδιά (keywords). Οι λέξεις κλειδιά συνδέονται με τους κανόνες και τη δομή της γλώσσας, και δεν μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών. Παραθέτουμε στον πίνακα παρακάτω τις λέξεις κλειδιά.

Πίνακας 1 - Λέξεις Κλειδιά της Python

FALSE	class	finally	is	return
None	continue	for	lambda	try
TRUE	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Παράλληλα, η Python χρησιμοποιεί τελεστές οι οποίοι είναι αριθμητικοί, σύγκρισης, τελεστές για τις Boolean εκφράσεις και οι λογικοί τελεστές πράξεων με bits (Bitwise).

Αριθμητικοί τελεστές είναι οι εξής

- (+) είναι ο τελεστής της πρόσθεσης, π.χ.
- (-) είναι ο τελεστής της αφαίρεσης, π.χ.
- (*) είναι ο τελεστής του πολλαπλασιασμού, π.χ.
- (/) είναι ο τελεστής της διαίρεσης, π.χ.
- (**) είναι ο τελεστής της δύναμης, π.χ.
- (//) είναι ο τελεστής της διαίρεσης στρογγυλοποιημένη προς τα κάτω (Floor Division), π.χ.
- (%) είναι ο τελεστής για το υπόλοιπο της διαίρεσης ή αλλιώς modulo, π.χ. .

Οι τελεστές σύγκρισης Όλοι οι τελεστές σύγκρισης επιστρέφουν True (Αληθής) ή False

(Ψευδής) και είναι οι εξής

- (<) μικρότερο, π.χ.
- (>) μεγαλύτερο, π.χ.
- (<=) μικρότερο ή ίσο, π.χ.
- (>=) μεγαλύτερο ή ίσο, π.χ.
- (==) ίσο συγκρίνει αν τα αντικείμενα είναι ίσα, π.χ. και
- (!=) είναι το διαφορετικό και συγκρίνει αν τα αντικείμενα δεν είναι ίσα.

Οι τελεστές για τις Boolean εκφράσεις είναι οι εξής

- (not) είναι ο τελεστής της Άρνησης (Λογικό Όχι),
- (and) είναι ο τελεστής για Σύζευξη (Λογικό ΚΑΙ),
- (or) είναι ο τελεστής για τη Διάζευξη (Λογικό Ή)

Τέλος οι λογικοί τελεστές πράξεων με bits

- (<<) είναι ο τελεστής της αριστερής μετάθεσης. Μεταθέτει τα δυαδικά ψηφία (bits) του αριθμού προς τα αριστερά κατά το πλήθος των θέσεων που καθορίστηκε.
- (>>) είναι ο τελεστής της δεξιάς μετάθεσης. Μεταθέτει τα bits του αριθμού προς τα δεξιά κατά το πλήθος των θέσεων που καθορίστηκε.

- (&) είναι ο τελεστής του δυαδικού ΚΑΙ δύο αριθμών. Το 5 & 3 δίνει 1.
- (|) είναι ο τελεστής του δυαδικού Ή δύο αριθμών. Το 5 | 3 δίνει 7.
- (^) είναι ο τελεστής του δυαδικού Αποκλειστικό Ή, π.χ. το 5 ^ 3 δίνει 6 και
- (~) είναι ο τελεστής τής δυαδικής αντιστροφής, π.χ. Το ~5 δίνει -6.

2.3 Δομές ελέγχου

Οι εντολές ελέγχου καθορίζουν τη σειρά με την οποία θα εκτελείται το πρόγραμμα. Οι ροές ελέγχου που καλύπτει η Python είναι οι εξής

2.3.1 Η δομή ελέγχου If …else

Η εντολή if περιέχει μια λογική έκφραση με την οποία τα δεδομένα συγκρίνονται και μια απόφαση γίνεται με βάση το αποτέλεσμα της σύγκρισης. Εάν η λογική έκφραση αξιολογείται ως Αληθής, τότε εκτελείται το μπλοκ των δηλώσεων εντός της εντολής if. Εάν η λογική έκφραση εκτιμάται σε Ψευδής, τότε εκτελείται το πρώτο σύνολο κώδικα μετά το τέλος της εντολής if. Συντακτικά η δομή της ροής είναι ως εξής:

```
if λογική έκφραση:
      εντολές
else λογική έκφραση:
      εντολές
```

2.3.2 Η δομή ελέγχου If … elif … else

Η εντολή ελέγχου if …elif …else μας επιτρέπει να ελέγχουμε πολλαπλές εκφράσεις. Η λέξη elif υπονοεί την έκφραση else if και γενικώς εκτελείτε ως εξής, αν η λογική έκφραση στο if είναι ψευδής τότε ελέγχεται η λογική έκφραση στο επόμενο elif για το αν είναι αληθής και εκτελείται το block εντολών κάτω από το elif, αλλιώς αν είναι ψευδής εκτελείται το block εντολών κάτω από το else. Μόνο ένα block εντολών εκτελείται σε αυτή τη δομή ελέγχου. Γενικά αυτή η δομή μπορεί να έχει περισσότερα του ενός elif, αλλά πρέπει να έχει ακριβώς ένα if και ένα else.

```
if λογική έκφραση:
      εντολές
elif λογική έκφραση:
      εντολές
else λογική έκφραση:
      εντολές
```

2.4 Επαναληπτικές δομές

Οι επαναληπτικές δομές χρησιμοποιούνται έτσι ώστε ένα block εντολών να μπορεί να επαναληφθεί για ορισμένο αριθμό φορών. Τέτοιες δομές είναι οι for και while.

2.4.1 Η δομή επανάληψης for

Η δομή επανάληψης στη Python χρησιμοποιείται για να μπορούμε να διατρέξουμε τα στοιχεία μίας λίστας ή για να επαναλάβουμε μία σειρά εντολών. Για τη δομή for χρησιμοποιείται η συνάρτηση range(), η οποία ουσιαστικά παράγει μία λίστα αριθμών που χρησιμοποιούνται στη for. Η range() μπορεί να πάρει μία σειρά παραμέτρων range(start, stop[, step]), τα ορίσματα stop και step μπορούν να παραληφθούν ορίζοντας μόνο το start. Στο όρισμα start μπορούμε να βάλουμε έναν αριθμό n και η δομή θα επαναληφθεί n φορές από το 0 μέχρι το $n-1$. Το όρισμα stop ορίζει πότε θα σταματήσει η επανάληψη και το όρισμα step ορίζει με τι βήμα θα γίνεται η επανάληψη. Προσέξτε ότι στη δομή επανάληψης αυτή μπορούμε να ορίσουμε και αρνητικό βήμα εφόσον ο δείκτης i είναι μεγαλύτερος του μηδενός. Η δομή for συντάσσεται ως εξής

```
for i in range():
      εντολές
```

2.4.2 Η δομή επανάληψης while

Η δομή επανάληψης while χρησιμοποιείται για να επαναλάβουμε ένα block εντολών για όσο η λογική έκφραση που ελέγχει η while είναι αληθής. Η δομή συντάσσεται ως εξής

```
while λογική έκφραση:
      εντολές
```

2.4.3 Εντολές ελέγχου break και continue

Στην Python οι εντολές break και continue χρησιμοποιούνται για να αλλάξουμε την κανονική ροή του προγράμματος. Οι δομές επανάληψης εκτελούνται μέχρι η λογική έκφραση να αποτιμάται ως ψευδής. Μερικές φορές όμως επιθυμούμε να τερματίσουμε την τρέχουσα επανάληψη δίχως να ελέγξουμε τη λογική έκφραση. Αυτό μπορούμε να το κάνουμε μέσω της εντολής break. Αυτή η εντολή τερματίζει τη δομή επανάληψης η οποία την περιέχει και η ροή προγράμματος εκτελεί της εντολές ακριβώς μετά από αυτή. Εάν για παράδειγμα η εντολή break εμφανίζεται σε εμφωλευμένη δομή επανάληψης τότε θα τερματιστεί ο πιο εσωτερικός βρόγχος. Δίνουμε παράδειγμα παρακάτω όπου εφόσον η λογική έκφραση στο if είναι αληθής θα εκτελεστεί η εντολή break και θα τερματίσει την δομή επανάληψης συνεχίζοντας την ροή του προγράμματος μετά το for loop.

```
for i in range():
      if λογική έκφραση:
            break
εντολές
```

Η εντολή continue χρησιμοποιείται για να παραλειφθούν οι υπόλοιπες εντολές μέσα σε ένα βρόγχο και συνεχίζει με την επόμενη επανάληψη. Στο παράδειγμα παρακάτω εφόσον

η λογική έκφραση στο if είναι αληθής θα εκτελεστεί η εντολή continue όπου και θα συνεχίσει να εκτελείται ο βρόγχος με την επόμενη επανάληψη οπότε οι εντολές δεν θα εκτελεστούν μετά το if.

```
for i in range():
      if λογική έκφραση:
            continue
      εντολές
```

2.5 Δομές δεδομένων

Η Python μας παρέχει από την βιβλιοθήκη της δομές για να μπορούμε να χειριστούμε διαφόρων ειδών δεδομένα. Τέτοιες δομές είναι οι λίστες (lists), τα σύνολα (sets), τα λεξικά (dictionaries), οι πλειάδες (tuples) και τα αλφαριθμητικά (strings).

2.5.1 Λίστες

Η λίστα είναι μία δομή δεδομένων η οποία περιέχει μία σειρά από αντικείμενα με την προϋπόθεση ότι γνωρίζουμε από πριν τον αριθμό των στοιχείων. Μπορούμε να ορίσουμε μία λίστα περιλαμβάνοντας μέσα σε τετράγωνες παρενθέσεις τα στοιχεία που θέλουμε να χειριστούμε. Τα στοιχεία αυτά μπορούν να είναι διαφορετικά μεταξύ τους, για παράδειγμα μπορούμε να έχουμε μία λίστα η οποία να αποτελείται από ακέραιους αριθμούς, αλφαριθμητικά ακόμα και άλλες λίστες. Προϋπόθεση είναι ότι διαχωρίζουμε τα στοιχεία με κόμμα.

Μπορούμε να ορίσουμε την κενή λίστα ως εξής: `myList = []`. Αν η λίστα περιέχει n στοιχεία μπορούμε να έχουμε πρόσβαση διατρέχοντας τη λίστα με δείκτη ξεκινώντας από το 0 μέχρι $n-1$. Για παράδειγμα για να διατρέξουμε όλα τα στοιχεία μπορούμε να το κάνουμε ως εξής:

```
for i in range(n):
      print(myList[i])
```

2.5.2 Σύνολα

Ένα σύνολο είναι μία συλλογή από μη-ταξινομημένα στοιχεία, τα οποία δεν είναι δεικτοδοτημένα. Τα σύνολα δεν μπορούν να περιέχουν διπλότυπα ως στοιχεία και κάθε στοιχείο μπορεί να εμφανίζεται το πολύ μία φορά. Τα σύνολα μας διευκολύνουν στην ομαδοποίηση διαφόρων στοιχείων με τα οποία μπορούμε να εκτελέσουμε μαθηματικές πράξεις συνόλων. Πράξεις συνόλων όπως η ένωση, η τομή, η διαφορά και η συμμετρική διαφορά. Για να δημιουργήσουμε ένα σύνολο χρησιμοποιούμε τις αγκύλες και η Python καταλαβαίνει πως δεν είναι λεξικό, αλλά ένα σύνολο επειδή δεν υπάρχει η άνω κάτω τελεία η οποία υποδηλώνει την τιμή που αντιστοιχίζεται σε κάθε κλειδί. Παραδείγματα ορισμών συνόλων δίνουμε παρακάτω

```
#Το κενό σύνολο ορίζεται ως
s = {}
#Ενα σύνολο με αντικείμενα ορίζεται ως
k = {1, 2, 'e', 'f '}
```

Όπως είπαμε παραπάνω μπορούμε να κάνουμε πράξεις συνόλων στην Python και η πρώτη πράξη που ορίζουμε είναι η ένωση. Με αυτήν την πράξη μπορούμε να συγχωνεύσουμε δύο σύνολα και κάθε στοιχείο θα εμφανιστεί στη συγχώνευση εκτός από τα διπλότυπα που θα εμφανιστούν μία φορά μόνο.

Για παράδειγμα έστω τα σύνολα `s = {1, 2, 3}` και `x = {3, 4, 5}` μπορούμε να κάνουμε την πράξη της ένωσης με δύο τρόπους είτε με το σύμβολο '|' είτε με την εντολή union ως εξής:

```
s|x
{1, 2, 3, 4, 5}

s.union(x)
{1, 2, 3, 4, 5}
```

Η δεύτερη πράξη που μπορούμε να εκτελέσουμε είναι η πράξη της τομής. Με την πράξη της τομής κάθε στοιχείο που είναι κοινό θα εμφανιστεί στην τομή. Για παράδειγμα έστω τα σύνολα `s = {1, 2, 3}` και `x = {3, 4, 5}` μπορούμε να κάνουμε την πράξη της τομής με δύο τρόπους είτε με το σύμβολο '&' είτε με την εντολή intersection ως εξής:

```
s&x
{3}

s.intersection(x)
{3}
```

Η επόμενη πράξη που μπορούμε να ορίσουμε είναι αυτή της διαφοράς συνόλων. Με τη πράξη της διαφοράς τα στοιχεία που εμφανίζονται είναι αυτά του πρώτου συνόλου τα οποία δεν ανήκουν στο δεύτερο σύνολο. Για παράδειγμα έστω τα σύνολα `s = {1, 2, 3}` και `x = {3, 4, 5},` τότε μπορούμε να κάνουμε την πράξη της αφαίρεσης με δύο τρόπους είτε με το σύμβολο της αφαίρεσης '–' είτε με την εντολή difference ως εξής:

```
s-x
{1, 2}

s.difference(x)
{1, 2}
```

Η επόμενη πράξη που μπορούμε να ορίσουμε είναι αυτή της συμμετρικής διαφοράς συνόλων. Με τη πράξη της συμμετρικής διαφοράς τα στοιχεία που εμφανίζονται είναι αυτά που δεν είναι κοινά στα δύο σύνολα. Για παράδειγμα έστω τα σύνολα `s = {1, 2, 3}` και `x = {3, 4, 5},` τότε μπορούμε να κάνουμε την πράξη της αφαίρεσης με δύο τρόπους είτε με το σύμβολο της αφαίρεσης '^' είτε με την εντολή symmetric_difference ως εξής:

```
s^x
{1, 2, 4, 5}

s.symmetric_difference(x)
{1, 2, 4, 5}
```

2.5.3 Λεξικά

Στην Python το λεξικό είναι μία αταξινόμητη συλλογή από στοιχεία. Το λεξικό είναι μία συλλογή από κλειδιά και στοιχεία, δηλαδή κάθε στοιχείο έχει και ένα κλειδί που δεν είναι απαραίτητα ένας ακέραιος αριθμός, αλλά μπορεί να είναι οποιοδήποτε αντικείμενο αρκεί να είναι σταθερό (να μην αλλάζει τιμή) και μοναδικό. Τα λεξικά είναι βελτιστοποιημένα ώστε αν είναι γνωστό το κλειδί να είναι εύκολη η ανάκτηση της τιμής.

Μπορούμε να δημιουργήσουμε λεξικά ως εξής:

```
# Αρχικοποίηση ενός κενού λεξικού
my_dictionary = {}

# Λεξικό με ακέραια κλειδιά
my_dict = {1: 'apple', 2: 'ball'}

# Λεξικό με μεικτά κλειδιά
my_dict = {'name': 'John', 1: [2, 4, 3]}

# Μπορούμε να χρησιμοποιήσουμε επίσης την εντολή dict()
my_dict = dict({1:'apple', 2:'ball'})
```

Μπορούμε να ανακτήσουμε τα στοιχεία από ένα λεξικό ως εξής:

```
# Μπορούμε να τυπώσουμε το στοιχείο αν γνωρίζουμε το κλειδί του
print(my_dict[2])

# Μπορούμε να τυπώσουμε τα στοιχεία επαναληπτικά
for key in my_dict():
    print(my_dict[key])
```

Επίσης με τα λεξικά μπορούμε να χρησιμοποιήσουμε διάφορες ενσωματωμένες μεθόδους τις οποίες αναφέρουμε συνοπτικά στον παρακάτω πίνακα.

Πίνακας 2 - Ενσωματωμένες μέθοδοι για τα λεξικά

Μέθοδος	Περιγραφή
clear()	Διαγράφει όλα τα στοιχεία του λεξικού
copy()	Επιστρέφει αντίγραφο του λεξικού
fromkeys(seq[, v])	Επιστρέφει ένα νέο λεξικό με κλειδιά από το seq και τιμές ίσες με το v (μπορεί να είναι ως προεπιλογή κενό)
get(key[,d])	Επιστρέφει την τιμή του κλειδιού, αλλιώς επιστρέφει το d (μπορεί να είναι ως προεπιλογή κενό)
items()	Επιστρέφει τα περιεχόμενα του λεξικού (κλειδί, στοιχείο)
keys()	Επιστρέφει τα κλειδιά του λεξικού
pop(key[,d])	Αφαιρεί το στοιχείο με κλειδί key και επιστρέφει d εάν το κλειδί δεν υπάρχει
popitem()	Αφαιρεί το τελευταίο στοιχείο στο λεξικό και εμφανίζει το στοιχείο που έχει αφαιρέσει

setdefault(key[,d])	Εάν το κλειδί υπάρχει στο λεξικό επιστρέφει το στοιχείο, αλλιώς εισάγει το κλειδί με στοιχείο d
update([other])	Ανανεώνει το λεξικό με τα ζευγάρια κλειδί: στοιχείο από το other αντικαθιστώντας τα υπάρχοντα κλειδιά
values()	Επιστρέφει τα στοιχεία του λεξικού

Επιπρόσθετα η κατανόηση λεξικού είναι ένας κομψός και ακριβής τρόπος να δημιουργήσουμε λεξικό από ένα διατρέξιμο αντικείμενο. Η κατανόηση αποτελείται από μία έκφραση που περιέχει το ζευγάρι κλειδί:στοιχεία ακολουθούμενο από μία for δήλωση μέσα από κατσαρές αγκύλες{}.

```
d = { k: v for k, v in enumerate ( ' pythonforbeginners ' ) i f v not
in ' python ' }
```

Στο συγκεκριμένο παράδειγμα θα μας εμφανίσει ένα καινούργιο λεξικό d το οποίο θα περιέχει σαν στοιχεία όλα εκείνα τα γράμματα τα οποία δεν περιέχονται στη λέξη python.

2.5.4 Πλειάδες

Στην python μία πλειάδα (tuple) είναι παρόμοια με την έννοια της λίστας. Η διαφορά μεταξύ τους είναι ότι δεν μπορούμε να αλλάξουμε τα στοιχεία μίας πλειάδας όταν τα έχουμε ήδη αναθέσει σε αντίθεση με την λίστα. Μερικά από τα πλεονεκτήματα της πλειάδας είναι αρχικά ότι μπορούμε να εισάγουμε διαφορετικά στοιχεία δεδομένων. Επίσης επειδή η πλειάδα είναι αμετάβλητη η διάτρεξή της είναι πιο γρήγορη από της λίστας, ακόμα τα στοιχεία μίας πλειάδας μπορούν να χρησιμοποιηθούν ως κλειδιά σε ένα λεξικό. Μπορούμε να δημιουργήσουμε μία πλειάδα παραθέτοντας όλα τα στοιχεία χωρισμένα με κόμμα μέσα σε παρένθεση. Αρχικοποίηση μίας πλειάδας παραθέτουμε παρακάτω.

```
my_tuple = (1, "γειά σου", 3.14)
```

Μπορούμε να διατρέξουμε μία πλειάδα όπως κάναμε και με την λίστα, π.χ., `print(my_tuple[0]),` το οποίο και θα μας εμφανίσει το πρώτο στοιχείο στην πλειάδα.

2.5.5 Αλφαριθμητικά

Μια συμβολοσειρά είναι μια ακολουθία χαρακτήρων. Ένας χαρακτήρας είναι απλά ένα σύμβολο. Στην Python, η συμβολοσειρά είναι μια ακολουθία Unicode χαρακτήρων. Το Unicode εισήχθη για να συμπεριλάβει κάθε χαρακτήρα σε όλες τις γλώσσες και να φέρει ομοιομορφία στην κωδικοποίηση. Τα αλφαριθμητικά αποτελούν ένα είδος ακολουθίας και αυτό σημαίνει πως κάθε στοιχείο είναι σε μία αριθμημένη σειρά, η οποία μπορούμε να την προσπελάσουμε. Τα αλφαριθμητικά μπορούν να δημιουργηθούν περικλείοντας χαρακτήρες μέσα σε ένα μονά ή διπλά εισαγωγικά. Ακόμη και τα τριπλά εισαγωγικά μπορούν να χρησιμοποιηθούν στη Python, αλλά γενικά χρησιμοποιούνται για να αντιπροσωπεύουν συμβολοσειρές πολλαπλών γραμμών και docstrings.

Παράδειγμα ανάθεσης αλφαριθμητικού είναι το `myString = 'Γειά σας'`. Μπορούμε να τυπώσουμε ολόκληρο το αλφαριθμητικό με την εντολή **`print()`** ή ακόμα μπορούμε να προσπελάσουμε συγκεκριμένα στοιχεία μέσα σε ένα αλφαριθμητικό αφού βρίσκονται σε αριθμημένη σειρά ξεκινώντας πάντα με δείκτη 0. Ταυτόχρονα, μπορούμε να αρχίσουμε την προσπέλαση των στοιχείων μετρώντας από το τέλος του αλφαριθμητικού προς την αρχή του αν χρησιμοποιήσουμε το σύμβολο μείον (-) πριν από τον δείκτη, για παράδειγμα η εντολή `print(myString[-1])`θα μας εμφανίσει το τελευταίο χαρακτήρα του αλφαριθμητικού.

2.6 Συναρτήσεις

Στην Python συνάρτηση είναι ένα σύνολο από δηλώσεις που εκτελούν ένα συγκεκριμένο έργο. Οι συναρτήσεις βοηθούν στο να σπάσουμε το πρόγραμμά μας σε μικρότερα λειτουργικά κομμάτια έτσι ώστε να έχουμε καλύτερο έλεγχο και μεγαλύτερη λειτουργικότητα. Επίσης μας βοηθάει στο να αποφεύγουμε την επανάληψη τμημάτων του κώδικα και να μπορούμε να τον επαναχρησιμοποιήσουμε. Μπορούμε να δηλώσουμε μία συνάρτηση ως εξής ξεκινάμε πάντα με τη λέξη-κλειδί def έπειτα δηλώνουμε το όνομα της συνάρτησης ακολουθούμενο με παρενθέσεις όπου και μπορούμε να δηλώσουμε ορίσματα και ολοκληρώνουμε με άνω-κάτω τελεία. Έπειτα μπορούμε να χρησιμοποιήσουμε την προαιρετική συμβολοσειρά τεκμηρίωσης (docstring) για να περιγράψουμε τι κάνει η συνάρτησή μας. Τέλος δημιουργούμε το μπλοκ εντολών της συνάρτησής και ολοκληρώνουμε προαιρετικά με την εντολή return αν θέλουμε η συνάρτηση να μας επιστρέφει κάποια συγκεκριμένη έξοδο. Παρακάτω δείχνουμε πως ορίζουμε μία συνάρτηση.

```
def function_name(parameters):
      """docstring"""
      statements
      return
```

2.7 Γεννήτορες και Επαναλήπτες

Η Python περιέχει κάποια χρήσιμα εργαλεία όπως του γεννήτορες (generators) και τους επαναλήπτες (iterators). Γενικά ένας γεννήτορας είναι μια ειδική διαδικασία που μπορεί να χρησιμοποιηθεί για τον έλεγχο της επαναληπτικής συμπεριφοράς ενός βρόχου. Στην πραγματικότητα, όλοι οι γεννήτορες είναι επαναλήπτες. Ένας γεννήτορας είναι παρόμοιος με μια συνάρτηση που επιστρέφει έναν πίνακα (array) και δεδομένου αυτού ένας γεννήτορας μπορεί να δέχεται παραμέτρους και να παράγει μια ακολουθία τιμών. Ωστόσο, αντί να δημιουργήσουμε έναν πίνακα που να περιέχει όλες τις τιμές και να τις επιστρέφουμε ταυτόχρονα, μια γεννήτρια αποδίδει τις τιμές μία κάθε φορά, η οποία απαιτεί λιγότερη μνήμη και επιτρέπει χρήστη να ξεκινήσει την πρώτη επεξεργασία των τιμών αμέσως. Εν ολίγοις, μια γεννήτρια μοιάζει με μια συνάρτηση, αλλά συμπεριφέρεται σαν ένας επαναλήπτης.

Επομένως δεδομένων των παραπάνω ένας επαναλήπτης είναι μία διαδικασία η οποία υλοποιείται μέσω ενός βρόγχου. Τέτοιου είδους επαναλήπτες βρίσκονται παντού μέσα στην Python αλλά είναι κρυμμένοι, παραδείγματα τέτοιων είναι οι γεννήτορες, σε

κατανοήσεις (comprehensions) και άλλα πολλά. Ο επαναλήπτης είναι ένα αντικείμενο το οποίο μπορούμε να έχουμε μία δομή επανάληψης και το αντικείμενο αυτό μας επιστρέφει δεδομένα ένα στοιχείο τη φορά. Ένα αντικείμενο καλείται επαναληπτικό (iterable) αν μπορούμε να πάρουμε έναν επαναλήπτη από αυτό. Οι περισσότερες ενσωματωμένες συλλογές (containers) όπως για παράδειγμα οι λίστες, οι πλειάδες, τα αλφαριθμητικά και άλλα είναι επαναληπτικά.

Όπως αναφέραμε παραπάνω οι επαναλήπτες είναι αντικείμενα και για να μπορέσουμε να τους χρησιμοποιήσουμε χρησιμοποιούμε τη συνάρτηση `next()`για να μπορέσουμε χειροκίνητα να κινηθούμε επαναληπτικά μεταξύ όλων των στοιχείων του επαναλήπτη. Όταν φτάσουμε στο τέλος και δεν υπάρχουν περισσότερα δεδομένα για να επιστραφούν τότε ο επαναλήπτης θα τερματιστεί.

3. Η Python σε σύγκριση με άλλες γνωστές γλώσσες

Σε αυτό το κεφάλαιο αναφέρουμε τις κύριες διαφορές μεταξύ των γλωσσών Python και C αλλά και που πλεονεκτεί η μία έναντι της άλλης.

3.1 Διαφορές μεταξύ Python και C

Η C είναι μία δομημένη γλώσσα προγραμματισμού που αναπτύχθηκε από τα εργαστήρια της Bell το 1972 από τον Dennis Ritchie. Αρχικά η C δημιουργήθηκε για την ανάπτυξη του λειτουργικού συστήματος UNIX και όλες οι γλώσσες έκτοτε προέρχονται έμμεσα ή άμεσα από τη C. Η C είναι μεσαίου επιπέδου γλώσσα και αυτού του είδους οι γλώσσες δεν παρέχουν ενσωματωμένες συναρτήσεις που βρίσκουμε σε γλώσσες υψηλού επιπέδου αλλά παρέχουν όλα τα δομικά στοιχεία που χρειαζόμαστε για να παράγουμε τα αποτελέσματα που θέλουμε.

Η C είναι μια δομημένη γλώσσα όπου το πρόγραμμα χωρίζεται σε μικρότερες συναρτήσεις. Η δομή αυτή ακολουθεί την προσέγγιση "από την κορυφή προς τη βάση". Ένα πρόγραμμα C αποτελείται από διάφορα διακριτικά όπως λέξεις-κλειδιά ή ένα αναγνωριστικό, μια σταθερά, μια λέξη συμβολοσειράς ή ένα σύμβολο. Για να μεταγλωττίσουμε και να εκτελέσουμε προγράμματα C, πρέπει να εγκαταστήσουμε τον μεταγλωττιστή C στον υπολογιστή μας και μέσω ενός επεξεργαστή κειμένου συντάσσουμε προγράμματα σε C όπου μετά μπορούν να μεταγλωττιστούν.

Ένα πρόγραμμα στη C βασικά αποτελείται από τα εξής:

- Εντολές Προ-επεξεργαστή, τα οποία είναι αρχεία κεφαλίδας που χρειάζονται για την εκτέλεση ενός προγράμματος C
- Συναρτήσεις, όπου μας δίνει πολλές πληροφορίες σχετικά με μια συνάρτηση όπως το τι επιστρέφει σαν αποτέλεσμα και τα ονόματα των παραμέτρων που χρησιμοποιούνται μέσα στη συνάρτηση
- Μεταβλητές, οι οποίες ορίζονται στο πρόγραμμα και τις τιμές που ανατίθενται σε αυτές
- Η συνάρτηση main, όπου κάθε πρόγραμμα C ξεκινάει από αυτή τη συνάρτηση και περιέχει δύο τμήματα, το τμήμα δηλώσεων και το εκτελέσιμο τμήμα
- Σχόλια, τα οποία μπορούν να είναι σχετικά με τη δημιουργία του προγράμματος ή την ημερομηνία τροποποίησης, όνομα συγγραφέα κλπ. Οι χαρακτήρες ή οι λέξεις ή οτιδήποτε μέσα σε ένα σχόλιο πρέπει να ξεκινάει και να τελειώνει με το σύμβολο /*.

Τα κύρια σημεία που πρέπει να προσέχουμε όταν γράφουμε προγράμματα στη C είναι τα εξής: η C είναι μια γλώσσα προγραμματισμού ευαίσθητη σε πεζά-κεφαλαία (case-sensitive). Κάθε δήλωση προγράμματος C τελειώνει με ένα ερωτηματικό (;) που αναφέρονται ως τερματιστής δήλωσης.
Αντίθετα η Python είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού. Η απλή σύνταξη και η δυναμική πληκτρολόγηση της Python, μαζί με την ερμηνευμένη φύση της, την καθιστούν ιδανική γλώσσα για scripting και γρήγορη για την ανάπτυξη εφαρμογών σε πολλές περιοχές στις περισσότερες πλατφόρμες. Η Python είναι μια ερμηνευμένη γλώσσα, η οποία εξοικονομεί σημαντικό χρόνο κατά τη διάρκεια της ανάπτυξης του

προγράμματος αφού δεν είναι απαραίτητη η σύνδεση (linking) με τη σύνταξη (compilation). Η σύνταξη αφορά την επεξεργασία αρχείων πηγαίου κώδικα (.c) και τη δημιουργία ενός αρχείου "αντικειμένου". Η σύνδεση αφορά τη δημιουργία ενός και μόνο εκτελέσιμου αρχείου από πολλά αρχεία αντικειμένων.

Η Python μάς επιτρέπει να χωρίσουμε το πρόγραμμά μας σε ενότητες (modules) που μπορούν να επαναχρησιμοποιηθούν σε άλλα προγράμματα Python. Υπάρχει μεγάλη συλλογή από τυποποιημένες ενότητες που μπορούμε να χρησιμοποιήσουμε ως βάση των προγραμμάτων. Ορισμένες από αυτές τις λειτουργικές μονάδες παρέχουν στοιχεία όπως αρχεία εισόδου/εξόδου, κλήσεις συστήματος κ.α.

Η Python είναι παρόμοια με τη C σε όρους σύνταξης αλλά με λίγες λέξεις-κλειδιά, απλά δομημένη και ξεκάθαρο συντακτικό. Τόσο η C όσο και η Python μπορούν να χρησιμοποιηθούν επιτρέποντας την παράλληλη εκτέλεση κάποιων ξεχωριστών διεργασιών με διαφορετικούς σκοπούς το καθένα, μιας μεγαλύτερης διεργασίας (multithreading). Η Python είναι αντικειμενοστραφής και έχει το δικό της συλλέκτη απορριμμάτων. Αντίθετα, στη C, ο χρήστης πρέπει να διαχειριστεί τη μνήμη μόνος του. Ουσιαστικά με τον όρο συλλέκτης απορριμμάτων είναι η διαδικασία στην οποία όταν δεν χρειάζονται πλέον τα αντικείμενα, η Python επαναφέρει αυτόματα τη μνήμη από αυτά.

Η C είναι μια μεταγλωττισμένη γλώσσα. Ο πλήρης πηγαίος κώδικας μετατρέπεται σε γλώσσα μηχανής, η οποία είναι ευκολότερη για τον υπολογιστή να κατανοήσει. Η Python από την άλλη πλευρά ερμηνεύεται. Ο διερμηνέας διαβάζει κάθε πρόταση γραμμής ανά γραμμή. Αυτό καθιστά την Python πιο αργή σε σύγκριση με τη C. Η C χρησιμοποιείται κυρίως για εφαρμογές που σχετίζονται με το υλικό. Ακολουθεί ένα προστακτικό μοντέλο προγραμματισμού. Οι δείκτες είναι διαθέσιμοι στη C. Υπάρχει περιορισμένος αριθμός ενσωματωμένων συναρτήσεων. Η εκτέλεση κώδικα είναι ταχύτερη από την Python. Η εφαρμογή των δομών δεδομένων απαιτεί τη ρητή δημιουργία συναρτήσεων. Είναι υποχρεωτικό να δηλώνεται ο τύπος μεταβλητού σε C.

Η Python είναι γλώσσα προγραμματισμού γενικού σκοπού. Είναι αντικειμενοστραφής. Δεν υπάρχει διαθέσιμη λειτουργία δείκτη. Είναι ερμηνεύσιμη. Έχει μεγάλη βιβλιοθήκη ενσωματωμένων λειτουργιών. Είναι πιο αργή σε σύγκριση με το C καθώς έχει συλλέκτη απορριμμάτων. Παρέχει ευκολία στην υλοποίηση δομών δεδομένων με ενσωματωμένες λειτουργίες εισαγωγής. Δεν χρειάζεται η δήλωση ενός τύπου μεταβλητής. Τα προγράμματα Python είναι πιο εύκολα στην εκμάθηση μιας γλώσσας προγραμματισμού.
Τέλος προκύπτει μία δύσκολη ερώτηση σχετικά με το πότε να χρησιμοποιείτε η Python και πότε η C. Γενικά είναι παρόμοιες γλώσσες αλλά έχουν πολλές βασικές διαφορές. Είναι χρήσιμες γλώσσες για την ανάπτυξη ποικίλων εφαρμογών. Η διαφορά μεταξύ C και Python είναι ότι η Python είναι μια γλώσσα πολλαπλών παραδειγμάτων και η C είναι μια δομημένη γλώσσα προγραμματισμού. Η Python είναι μια γλώσσα γενικής χρήσης που χρησιμοποιείται στους τομείς της μηχανικής μάθησης, στην επεξεργασία φυσικής γλώσσας, στην ανάπτυξη ιστοσελίδων και πολλών άλλων. Η C χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών που σχετίζονται με το υλικό, όπως λειτουργικά συστήματα, προγράμματα οδηγών δικτύου (https://www.educba.com/c-vs-python/).

4. Η Python ως γλώσσα εκμάθησης στη Γ'βαθμια εκπαίδευση παραδείγματα και εφαρμογές

Σε αυτό το κεφάλαιο παραθέτουμε τις ασκήσεις εκμάθησης της γλώσσας C του ΤΕΙ Δυτικής Ελλάδος στο Τμήμα Διοίκησης Επιχειρήσεων τις οποίες μετατρέπουμε σε γλώσσα Python. Παραθέτουμε την εκφώνηση κάθε άσκησης και έπειτα τον κώδικα σε γλώσσα Python που δημιουργήσαμε.

4.1 Ασκήσεις με εντολή print

Εκφώνηση 4.1.1:

```
#Εκτυπώστε τη φράση: Hello World of Business Planning and
Information #Systems

print("Hello World of Business Planning and Information
Systems")
```

4.2 Ασκήσεις με απλούς υπολογισμούς

Εκφώνηση 4.1.2:

```
#Εκτυπώστε τα παρακάτω:
#Hello World of Business Planning and Information
Systems\n");
#TMHMA \t
#EPIXEIRHMATIKOY SXEDIASMOY \t
#KAI \t"
#PLHROFORIAKWN SYSTHMATWN

print("Hello World of Business Planning and Information
Systems\n")
print("TMHMA \t EPIXEIRHMATIKOY SXEDIASMOY \t KAI \t
PLHROFORIAKWN SYSTHMATWN\n")
```

Εκφώνηση 4.2.1:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να προσθέτει
τις τρεις #πλευρές ενός τριγώνου και να εμφανίζει το
άθροισμα στην οθόνη του #υπολογιστή.
```

```
#Πλευρά a = 10, b= 15 και c = 25.


a = 10
b = 15
c = 25
sum = 0

#άθροισμα των τριών πλευρών
sum = a + b + c
#τυπώνω το άθροισμα
print("ΤΟ ΑΘΡΟΙΣΜΑ ΕΙΝΑΙ =",sum)
```

Εκφώνηση 4.2.2:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
υπολογίζει και να #εμφανίζει στην οθόνη του υπολογιστή το
εμβαδόν τραπεζίου, όταν
#δίνονται οι βάσεις και το ύψος του.
#Βάση Μεγάλη b1 = 10, Βάση Μικρή b2 = 4 και Υψος y = 5.
#Το εμβαδόν τραπεζίου δίνεται αν πολλαπλασιάσουμε το
ημιάθροισμα των #βάσεων επί το ύψος. emb = ((b1 + b2) / 2)
* y.

b1 = 10
b2 = 4
y = 5
emb = 0
#Βρίσκω το εμβαδόν του τραπεζίου
emb = ((b1 + b2) / 2) * y
print("εμβαδόν τραπεζίου = ",emb)
```

Εκφώνηση 4.2.3:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
υπολογίζει και να #εμφανίζει στην οθόνη του υπολογιστή το
μέσο όρο της βαθμολογίας ενός #φοιτητή σε 4 μαθήματα. math
= 6, phys = 8, chem = 6  και comp = 9.

math = 6
phys = 8
chem = 6
comp = 9
mo = 0

#Βρίσκω τον μέσο όρο
```

```
mo = ((math + phys + chem + comp)/4)
print("Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ",mo)
```

Εκφώνηση 4.2.4:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να προσθέτει
#τις τρεις πλευρές ενός τριγώνου και να εμφανίζει το
#άθροισμα στην οθόνη του #υπολογιστή. Οι πλευρές του
#τριγώνου θα δίνονται από το χρήστη.

#Παίρνω από το χρήστη τα μήκη των πλευρών
a = int(input("Dwse to mikos tis plevras a = "))
b = int(input("Dwse to mikos tis plevras b = "))
c = int(input("Dwse to mikos tis plevras c = "))
#Υπολογίζω το άθροισμα κατα την τύπωση
print("To athroisma twn plevrwn tou trigwnou einai: ",
a+b+c)
```

Εκφώνηση 4.2.5:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να εμφανίζει στην οθόνη του υπολογιστή το
#εμβαδόν τραπεζίου, όταν δίνονται οι βάσεις και το ύψος
#του. Οι βάσεις και το ύψος θα δίνονται από τον χρήστη. Το
#εμβαδόν τραπεζίου δίνεται αν πολλαπλασιάσουμε το ημι-
#άθροισμα των βάσεων επί το ύψος. emb = ((b1 + b2)/2) * y.



#Παίρνω απο το χρήστη τα μήκη των πλευρών και το ύψος του
#τραπεζίου
b1 = int(input("Dwse to mikos tis mikris basis b1 = "))
b2 = int(input("Dwse to mikos tis megalis basis b2 = "))
y = int(input("Dwse to ypsos y = "))

#Υπολογίζω το εμβαδόν κατα την τύπωση του αποτελέσματος
print("To emvadon tou trapeziou einai: ", ((b1 + b2) / 2) *
y)
```

Εκφώνηση 4.2.6:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να εμφανίζει στην οθόνη του υπολογιστή το
#μέσο όρο της βαθμολογίας και το Ονοματεπώνυμο ενός φοιτητή
#σε 4 μαθήματα. Οι βαθμολογίες του φοιτητή στα 4 μαθήματα
#το Όνομα και το Επώνυμο του θα δίνονται από τον χρήστη.


#Παίρνω απο το χρήστη τα απαραίτητα στοιχεία
math = int(input("ΕΙΣΑΓΕΤΕ ΤΟ ΒΑΘΜΟ ΣΑΣ ΣΤΑ ΜΑΘΗΜΑΤΙΚΑ: "))
chem = int(input("ΕΙΣΑΓΕΤΕ ΤΟ ΒΑΘΜΟ ΣΑΣ ΣΤΗ ΧΗΜΕΙΑ: "))
phys = int(input("ΕΙΣΑΓΕΤΕ ΤΟ ΒΑΘΜΟ ΣΑΣ ΣΤΗ ΦΥΣΙΚΗ: "))
comp = int(input("ΕΙΣΑΓΕΤΕ ΤΟ ΒΑΘΜΟ ΣΑΣ ΣΤΟΥΣ ΥΠΟΛΟΓΙΣΤΕΣ:
"))
onoma = input("ΔΩΣΤΕ ΤΟ ΟΝΟΜΑ ΣΑΣ: ")
epwnimo = input("ΔΩΣΤΕ ΤΟ ΕΠΙΘΕΤΟ ΣΑΣ: ")

#Τυπώνω το Όνομα και Επώνυμο
print("Όνομα: ",onoma,"\t Επώνυμο: ",epwnimo)

#Υπολογίχω το Μέσο Όρο
mo = (math + chem + phys + comp)/4

#Τυπώνω το αποτέλεσμα
print("Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ: ",mo)
```

Εκφώνηση 4.2.7:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#τις ακέραιες μεταβλητές a, b, c να υπολογίζει και να
#εμφανίζει με κατάλληλα μηνύματα στην οθόνη του υπολογιστή
#το άθροισμα του sum, το γινόμενο τους product και το μέσο
#όρο τους aver.


#Παίρνω απο το χρήστη τα απαραίτητα στοιχεία
a = float(input("ΕΙΣΑΓΕΤΕ ΕΝΑΝ ΑΚΕΡΑΙΟ ΑΡΙΘΜΟ: "))
b = int(input("ΕΙΣΑΓΕΤΕ ΕΝΑΝ ΑΚΕΡΑΙΟ ΑΡΙΘΜΟ: "))
c = int(input("ΕΙΣΑΓΕΤΕ ΕΝΑΝ ΑΚΕΡΑΙΟ ΑΡΙΘΜΟ: "))

#Υπολογίζω το άθροισμα
sum = a + b + c

#Υπολογίζω το γινόμενο
product = a * b * c
```

```
#Υπολογίζω το μέσο όρο
aver = (a + b + c)/3

#τυπώνω τα αποτελέσματα
print(" ΤΟ ΑΘΡΟΙΣΜΑ ΕΙΝΑΙ: ",sum)
print("ΤΟ ΓΙΝΟΜΕΝΟ ΕΙΝΑΙ: ",product)
print("Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ: ",aver)
```

Εκφώνηση 4.2.8:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει την τελική αξία ενός προϊόντος του οποίου η
#αρχική αξία υπόκειται σε ΦΠΑ. Το πρόγραμμα θα διαβάζει την
#αρχική αξία (axia), το ΦΠΑ (fpa) και θα #εμφανίζει στην
#οθόνη του υπολογιστή τα εξής:
#Arxiki Axia Proiontos = .............
#FPA                   = .............
#Synoliko FPA          = .............
#Teliki Axia Proiontos = .............


#Παίρνω από το χρήστη τα απαραίτητα στοιχεία
aa = int(input("Dwse tin arxiki aksia tou proiontos aa =
"))
fpa = float(input("Dwse to FPA tou proiontos se dekadiko
fpa = "))

#υπολογίζω το συνολικό ΦΠΑ
synoliko_fpa = aa * fpa

#Υπολογίζω τη συνολική αξία του προϊόντος
teliki_aksia = synoliko_fpa + aa

#Τυπώνω τα αποτελέσματα
print("Arxiki Axia Proiontos = ",aa)
print("FPA = ", fpa)
print("Synoliko FPA = ",synoliko_fpa)
print("Teliki Axia Proiontos = ",teliki_aksia)
```

Εκφώνηση 4.2.9:

```
'''Να γραφεί πρόγραμμα σε γλώσσα Python που να διαβάζει τα
στοιχεία ενός σπουδαστή καθώς και τους βαθμούς του σε 5
```

μαθήματα και να τα τυπώνει στην οθόνη του υπολογιστή ως εξής:

```
------------------------------------------------------------
------------
Tmima : ................
Epwnymo : ....................        Onoma :
................
------------------------------------------------------------
------------
Mesos Oros Mathematwn :............
------------------------------------------------------------
------------
Mathematika  : ............             Physiki    :
.........
Programming : ............             Language :
.........
Ximeia          : ............
------------------------------------------------------------
------------'''

#Παίρνω από το χρήστη τα απαραίτητα στοιχεία
onoma = input("Dwse to Onoma tou Foititi: ")
epwnymo = input("Dwse to Epwnymo tou Foititi: ")
tmima = input("Dwse to tmima tou Foititi: ")
math = float(input("Dwse to vathmo tou Foititi sta
Mathematika: "))
prog = float(input("Dwse to vathmo tou Foititi sto
Programmatismo: "))
chem = float(input("Dwse to vathmo tou Foititi sti Xhmeia:
"))
phys = float(input("Dwse to vathmo tou Foititi sti Physiki:
"))
lang = float(input("Dwse to vathmo tou Foititi sti Glwssa:
"))

#Τυπώνω τα στοιχεία σύμφωνα με το υπόδειγμα
print("------------------------------------------------------
--------------------")
print("Tmima : ",tmima)
print("Epwnymo : {} \t\tOnoma : {}".format(epwnymo,onoma))
print("------------------------------------------------------
--------------------")
print("Mesos Oros Mathematwn :
{}".format((math+prog+chem+phys+lang)/5))
print("------------------------------------------------------
--------------------")
```

```
print("Mathematika  : {} \t\tPhysiki    :
{}".format(math,phys))
print("Programming : {} \t\tLanguage :
{}".format(prog,lang))
print("Ximeia: {}".format(chem))
print("-------------------------------------------------
------------------")
```

4.3 Ασκήσεις με εντολή if

Εκφώνηση 4.3.1:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#δύο ακέραιους αριθμούς a και b να τους συγκρίνει και να
#εμφανίζει ποιος είναι μεγαλύτερος.


#Δημιουργώ συνάρτηση που επιστρέφει το μεγαλύτερο αριθμό ή
#αν  είναι ίσοι
def compare(a,b):
     if(a>b):
          return(int(a))
     elif(a<b):
          return(int(b))
     else:
          return("οι αριθμοί είναι ίσοι")

#Ο χρήστης εισάγει τους 2 αριθμούς
x = input("dwse ton prwto akeraio:")
y = input("dwse ton deftero akeraio:")

#Καλώ τη συνάρτηση που συγκρίνει τους αριθμούς
c = compare(x,y)

#τυπώνω το αποτέλεσμα
if(type(c) == str):
     print(c)
else:
     print("O megalyteros einai: ", c)
```

Εκφώνηση 4.3.2:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#αποφασίζει αν ένας φοιτητής προβιβάστηκε ή όχι σύμφωνα με
#τα αποτελέσματα των γραπτών του σε 4 μαθήματα. Ο φοιτητής
#προβιβάζεται αν ο μέσος όρος των βαθμών του στα 4 μαθήματα
#είναι τουλάχιστον 10 υπό την προϋπόθεση ότι δεν πήρε σε
#κανένα μάθημα βαθμό κάτω από 8.

#Παίρνω τα στοιχεία απο το χρήστη
a = int(input("dwse bathmo gia to prwto mathima: "))
b = int(input("dwse bathmo gia to deftero mathima: "))
c = int(input("dwse bathmo gia to trito mathima: "))
d = int(input("dwse bathmo gia to tetarto mathima: "))

#Υπολογίζω το Μέσο Όρο
mo = (a+b+c+d)/4

#Τυπώνω αν ο φοιτητής προβιβάζεται ή όχι
if (mo >= 10 and a>=8 and b>=8 and c>=8 and d>=8):
     print("o foititis provivazetai")
else:
     print("o foititis den provivazetai")
```

Εκφώνηση 4.3.3:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει τις αποδοχές ενός υπαλλήλου ως εξής: Για μικτό
#μισθό μέχρι 2.000 ευρώ το ποσοστό των κρατήσεων είναι
#12,5% και για μεγαλύτερο μισθό 15%.
#Το πρόγραμμα θα διαβάζει το πλήθος των ωρών που εργάστηκε
#ο υπάλληλος και το ύψος του ωρομισθίου και θα εμφανίζει τα
#εξής:
#ΣΥΝΟΛΟ ΩΡΩΝ : .................ΩΡΟΜΙΣΘΙΟ : ................
#ΣΥΝΟΛΙΚΕΣ ΑΠΟΔΟΧΕΣ : ............
#ΣΥΝΟΛΟ ΚΡΑΤΗΣΕΩΝ : ............
#ΚΑΘΑΡΟΣ ΜΙΣΘΟΣ : ............

#Παίρνω τα στοιχεία από το χρήστη
wres = int(input("dwse to synolo twn wrwn: "))
wromisthio = int(input("dwse wromisthio: "))

#Υπολογίζω τις συνολικές αποδοχές
synolikes_apodoxes = wres*wromisthio
if (synolikes_apodoxes <= 2000):
     kratiseis = 0.125 * synolikes_apodoxes
```

```
        katharos_misthos = synolikes_apodoxes - kratiseis
else:
        kratiseis = 0.15 * synolikes_apodoxes
        katharos_misthos = synolikes_apodoxes - kratiseis

#Τυπώνω τα στοιχεία
print("ΣΥΝΟΛΟ ΩΡΩΝ :{} \t ΩΡΟΜΙΣΘΙΟ :
{}".format(wres,wromisthio))
print("ΣΥΝΟΛΙΚΕΣ ΑΠΟΔΟΧΕΣ :", synolikes_apodoxes)
print("ΣΥΝΟΛΟ ΚΡΑΤΗΣΕΩΝ :", kratiseis)
print("ΚΑΘΑΡΟΣ ΜΙΣΘΟΣ :", katharos_misthos)
```

Εκφώνηση 4.3.4:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα ακέραιο αριθμό a και να εμφανίζει στην οθόνη του
#υπολογιστή την απόλυτη τιμή του αριθμού αυτού.


#Παίρνω απο το χρήστη τα στοιχεία
x = int(input("Dwse enan akeraio arithmo: "))

#Υπολογίζω την απόλυτη τιμή του αριθμού
if(x > 0):
        print("Η απόλυτη τιμή του αριθμού είναι: ", x)
elif(x < 0):
        print("Η απόλυτη τιμή του αριθμού είναι: ", x * (-1))
else:
        print("Ο αριθμός ειναι μηδενικός")
```

Εκφώνηση 4.3.5:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα ακέραιο αριθμό n. Να ελέγχει αν ο αριθμός είναι άρτιος
#και να εμφανίζει ένα μήνυμα ότι ο αριθμός είναι άρτιος,
#διαφορετικά να εμφανίζει μήνυμα ότι ο αριθμός είναι
#περιττός.

#Παίρνω απο το χρήστη τα στοιχεία
a = int(input("dwse enan thetiko akeraio: "))

#Υπολογίζω αν είναι άρτιος ο αριθμός
if(a % 2 == 0):
        print("o aritmos einai artios")
```

```
else:
     print("o aritmos einai perittos")
```

Εκφώνηση 4.3.6:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#έναν ακέραιο αριθμό n. Να ελέγχει αν ο αριθμός είναι
#μεγαλύτερος, μικρότερος ή ίσος του 0 και να εμφανίζει ένα
#μήνυμα ότι ο αριθμός είναι θετικός, αρνητικός ή μηδέν.

#Παίρνω τον αριθμό από το χρήστη
a= int(input("dwse enan akeraio arithmo: "))

#Υπολογίζω και τυπώνω τι είναι ο αριθμός
if(a>0):
     print("einai thetikos")
elif(a==0):
     print("einai mhden")
else:
     print("einai arnhtikos")
```

Εκφώνηση 4.3.7:

```
#Να γραφεί πρόγραμμα σε γλώσσα C το οποίο να διαβάζει τρεις
#ακέραιους αριθμούς a, b, c να τους συγκρίνει και να
#εμφανίζει ποιος είναι μεγαλύτερος.


#Παίρνω τα στοιχεία από το χρήστη
a = int(input("Dwse ton prwto akeraio arithmo a = "))
b = int(input("Dwse to deutero akeraio arithmo b = "))
c = int(input("Dwse ton trito akeraio arithmo c = "))

#Βάζω τον ένα αριθμό ως μέγιστο
max = a

#Αποθηκεύω ποιά είναι η ονομασία της μεταβλητής
let = "a"

#Υπολογίζω το μέγιστο
if(max < b):
     max = b
     let = "b"
if(max < c):
```

```
        max = c
        let = "c"

#Τυπώνω το μέγιστο στοιχείο
print("To megisto stoixeio einai to {} = {}".format(let,
max))
```

Εκφώνηση 4.3.8:

```
#Η φορολογία εισοδήματος φυσικών προσώπων για μισθούς και
#συντάξεις οικονομικού έτους 2005, υπολογίζεται από τις
#αρμόδιες υπηρεσίες του υπουργείου των Οικονομικών, με τη
#βοήθεια του παρακάτω πίνακα:
#Κλιμάκιο εισοδήματος    Φορολογικός συντελεστής     Φόρος
#κλιμακίου
```

#Κλιμάκιο εισοδήματος	Φορολογικός συντελεστής	Φόρος κλιμακίου	Σύνολο εισοδήματος	Σύνολο φόρου
#10.000	0%	0	10000	0
#3.400	15%	510	13400	510
#10.000	30%	3000	23400	3510
#Υπερβάλλον	40%			

```
#Για κάθε φορολογούμενο δίνονται τα εξής στοιχεία: αριθμός
#φορολογικού μητρώου (AFM), επώνυμο φορολογούμενου,
#φορολογητέο εισόδημα. Να γραφτεί πρόγραμμα σε γλώσσα
#Python το οποίο να διαβάζει τα στοιχεία του φορολογούμενου
#και να υπολογίζει και να τυπώνει το φόρο που τους
#αντιστοιχεί.

#Παίρνω από το χρήστη τα απαραίτητα στοιχεία
afm = int(input("dwse afm: "))
epwnumo = input("dwse epwnumo: ")
eisodima = int(input("dwse eisodima: "))

#Υπολογίζω το φόρο εισοδήματος
if (eisodima <= 10000):
        poso = 0
elif (eisodima <= 13400):
        poso = 0 + (eisodima - 10000)* 15/100
elif (eisodima <= 23400):
        poso = 0 + 510 + (eisodima - 13400) * 30/100
```

```
else:
     poso = 0 + 510 + 3000 + (eisodima - 23400) * 40/100

#Τυπώνω το φόρο εισοδήματος
print("O forologoumenos me Epwnymo {} kai AFM {}
\n".format(epwnumo,afm));
print("Me forologhteo eisodima {} 8a plirwsei foro =
{}".format(eisodima,poso));
```

Εκφώνηση 4.3.9:

```
#Μια εταιρεία κινητής τηλεφωνίας ακολουθεί ανά μήνα την
#πολιτική τιμών #που φαίνεται στον παρακάτω πίνακα:
#Πάγιο 1500 δραχμές
#Χρόνος τηλεφωνημάτων (δευτερόλεπτα) Χρονοχρέωση
#(δραχμές/δευτερόλεπτο)
#              1-500                          1,5
#              501-800                  0,9
#              801 και άνω                    0,5
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#α) να διαβάζει τη χρονική διάρκεια των τηλεφωνημάτων ενός
#συνδρομητή σε διάστημα ενός μήνα
#β) να υπολογίζει τη μηνιαία χρέωση του συνδρομητή
#γ) να εμφανίζει (τυπώνει) τη λέξη «XREWSH» και τη μηνιαία
χρέωση του #συνδρομητή.
#Διευκρίνηση: η χρονοχρέωση στο πίνακα θεωρείται κλιμακωτή.
#Δηλαδή τα πρώτα 500 δευτερόλεπτα χρεώνονται με 1,5
#δρχ/δευτερόλεπτο,
#τα επόμενα 300 δευτερόλεπτα με 0,9 δρχ/δευτερόλεπτο και
#τα πέραν των 800 με 0,5 δρχ/δευτερόλεπτο.

#Παίρνω από το χρήστη τη συνολική διάρκεια των
#τηλεφωνημάτων σε δευτερόλεπτα
sd_til = int(input("Dwse ti synoliki diarkeia gia ena mina
se deuterolepta: "))

#Υπολογίζω τη χρονοχρέωση
if(sd_til <= 500):
     xrewsi = sd_til * 1.5
elif(sd_til > 500 & sd_til <= 800):
     xrewsi = (500 * 1.5) + ((sd_til - 500)*0.9)
else:
     xrewsi = (500 * 1.5) + ((800-500)*0.9) + ((sd_til-800)
* 0.5)
```

```
#Τυπώνω τη χρέωση
print("XREWSH = ", xrewsi)
```

4.4 Ασκήσεις με εντολή switch

Εκφώνηση 4.4.1:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα ακέραιο αριθμό n από το 1 μέχρι το 12 και να εμφανίζει
#τον αντίστοιχο μήνα του έτους (1 για Ιανουάριο, 2 για
#Φεβρουάριο, ..., 12 για Δεκέμβριο).

#Δημιουργούμε μια συνάρτηση η οποία θα δέχεται ως είσοδο
#τον αριθμό του #μήνα και θα επιστρέφει το μήνα
def months(arg):
     switcher = {
          1: "ianouarios",
          2: "fevrouarios",
          3: "martios",
          4: "aprilios",
          5: "maios",
          6: "iounios",
          7: "ioulios",
          8: "augoustos",
          9: "septemvrios",
          10: "oktobrios",
          11: "noemvrios",
          12: "dekemvrios"
          }
     return(switcher[arg])

#Ο χρήστης μας δίνει ως είσοδο έναν ακέραιο αριθμό από 1-12
minas = int(input("Dwse enan arithmo apo to 1-12: "))

#Τυπώνουμε το μήνα
print("o minas einai o {}".format(months(minas)))
```

Εκφώνηση 4.4.2:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#τα αποτελέσματα των εξετάσεων αγγλικών ενός φοιτητή. Αν ο
```

```
#φοιτητής πήρε Α να εμφανίζει το μήνυμα ARISTA, Β το μήνυμα
#KALA, C το μήνυμα METRIA και F το μήνυμα APETYXE.

#Δημιουργούμε μία συνάρτηση η οποία δέχεται ως είσοδο ένα
#χαρακτήρα και επιστρέφει το αποτέλεσμα
def apotelesma(arg):
     switcher = {
          'A': "arista",
          'B': "kala",
          'C': "metria",
          'F': "apetyxe"
          }
     return(switcher[arg])

#Παίρνουμε ως είσοδο το βαθμό από το χρήστη
vathmos = input("Dwse vathmo (A, B, C, F): ")

#Ελέγχουμε την είσοδο που μας δίνει ο χρήστης
while(vathmos != 'A' and vathmos != 'B' and vathmos != 'C'
and vathmos != 'F'):
     vathmos = input("Ksanadwse Vathmo (A, B, C, F): ")

#Τυπώνουμε το αποτέλεσμα
print("{}".format(apotelesma(vathmos)))
```

Εκφώνηση 4.4.3:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#δύο ακέραιους αριθμούς x, y. Θα διαβάζει επίσης έναν από
#τους τρεις χαρακτήρες (+, -#, *). Ανάλογα με το χαρακτήρα
#που διαβάζει θα κάνει και την αντίστοιχη πράξη (πρόσθεση,
#αφαίρεση ή πολλαπλασιασμό) και θα εμφανίζει τα κατάλληλα
#μηνύματα.

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο το
#σύμβολο της πράξης και δυο αριθμούς και επιστρέφει την
#ανάλογη πράξη
def praxi(x,y,sumbolo):
     switcher = {
          '+': x+y,
          '-': x-y,
          '*': x*y
          }
```

```
     return(switcher[sumbolo])

#Ο χρήστης μας δίνει το σύμβολο της πράξης και δύο
#ακέραιους αριθμούς
sym = input("Dwse sumbolo (+, -, *): ")
arithmos1 = int(input("dwse ton prwto arithmo: "))
arithmos2 = int(input("dwse to deftero arithmo: "))

#Τυπώνουμε τι είδους πράξη κάνουμε
if(sym == '+'):
     print("einai prosthesi")
elif(sym == '-'):
     print("einai aferesi")
else:
     print("einai pollaplasiasmos")

#Τυπώνουμε το αποτέλεσμα της πράξης
print("{}".format(praxi(arithmos1,arithmos2,sym)))
```

Εκφώνηση 4.4.4:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα ακέραιο αριθμό n από το 1 μέχρι το 7 και να εμφανίζει
#την αντίστοιχη ημέρα της εβδομάδας (1 για Κυριακή, 2 για
#Δευτέρα, ..., 7 για Σάββατο).

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#ακέραιο αριθμό ο οποίος αντιστοιχεί σε μία ημέρα
def days(arg):
     switcher = {
          1: "deftera",
          2: "trith",
          3: "tetarth",
          4: "pempth",
          5: "paraskeuh",
          6: "savvato",
          7: "kuriakh"
          }
     return(switcher[arg])

#Παίρνουμε από το χρήστη έναν αριθμό
hmera = int(input("Dwse enan arithmo apo to 1-7: "))

#Τυπώνουμε την ημέρα καλώντας τη συνάρτηση
print("h hmera einai  {}".format(days(hmera)))
```

4.5 Ασκήσεις με εντολή for

Εκφώνηση 4.5.1:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το
#γινόμενο των αριθμών από το 1 έως το N.

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο ένα
#ακέραιο αριθμό και επιστρέφει το άθροισμα όλων των αριθμών
#από 1-N
def summation(N):
     sum = 0
     for i in range(1, N+1):
          sum = sum + i
     return(sum)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο ένα
#ακέραιο αριθμό και επιστρέφει το γινόμενο όλων των αριθμών
#από 1-N
def multiply(N):
     mult = 1
     for i in range(1, N+1):
          mult = mult * i
     return(mult)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο ένα
#ακέραιο αριθμό και επιστρέφει το μέσο όρο των αριθμών από
#1-N
def average(N):
     s = summation(N)
     avg = s/N
     return(avg)

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό
arithmos = int(input("dwse arithmo: "))

#Τυπώνουμε τα αποτελέσματα καλώντας τις συναρτήσεις
print("to athroisma einai {} \nto ginomeno einai {} \no
mesos oros einai {}".format(summation(arithmos),
multiply(arithmos), average(arithmos)))
```

Εκφώνηση 4.5.2:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το
#γινόμενο των περιττών αριθμών από το 1 έως το N.

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#ακέραιο αριθμό N και επιστρέφει το άθροισμα όλων των
#περιττών αριθμών από 1-N
def summation(N):
     sum = 0
     count = 0
     for i in range(1, N+1):
          if(i % 2 == 1):
               sum = sum + i
               count = count + 1
     return([sum, count])

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#ακέραιο αριθμό N και επιστρέφει το γινόμενο όλων των
#περιττών αριθμών από 1-N
def multiply(N):
     mult = 1
     for i in range(1, N+1):
          if(i % 2 == 1):
               mult = mult * i
     return(mult)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#ακέραιο αριθμό N και επιστρέφει το μέσο όρο των περιττών
#αριθμών από 1-N
def average(N):
     s, n = summation(N)
     avg = s/n
     return(avg)

#Παίρνουμε ως είσοδο έναν ακέραιο αριθμό από το χρήστη
arithmos = int(input("dwse arithmo: "))

#Αποθηκεύουμε το αποτέλεσμα της συνάρτησης summation()
sum = summation(arithmos)

#Τυπώνουμε τα αποτελέσματα καλώντας τις συναρτήσεις
print("to athroisma einai {} \nto ginomeno einai {} \no
mesos oros einai {}".format(sum[0], multiply(arithmos),
average(arithmos)))
```

Εκφώνηση 4.5.3:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το να διαβάζει ένα
#αριθμό (ar) και τη δύναμη του αριθμού (n) και να
#υπολογίζει και να τυπώνει το αποτέλεσμά τους (π.χ. ar^n).

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#αριθμό και τη δύναμη αυτού και επιστρέφει το αποτέλεσμα
#της δύναμης
def power(ar,n):
     if(n == 0):
          return(1)
     if(n == 1):
          return(ar)
     else:
          tmp = ar
          for i in range(2,n+1):
               ar = ar * tmp
          return(ar)

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό
arithmos = int(input("Dwse enan arithmo: "))

#Παίρνουμε ως είσοδο από το χρήστη τη δύναμη που θα
υψώσουμε ον ακέραιο
dynami = int(input("Dwse ti dunami tou arithmou: "))

#Τυπώνουμε το αποτέλεσμα καλώντας τη συνάρτηση
print("To ar^n = {}".format(power(arithmos, dynami)))
```

Εκφώνηση 4.5.4:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το άθροισμα της πιο κάτω σειράς:
#Σ = 1^n + 2^n + 3^n + 4^n + 5^n

#Δημιουργούμε τη συνάρτηση η οποία υπολογίζει τις δυνάμεις
#δεχόμενη ως είσοδο έναν ακέραιο αριθμό και τη δύναμη αυτού
def power(ar, n):
     if(n == 0):
          return(1)
     if(n == 1):
          return(ar)
     else:
```

```
            tmp = ar
            for i in range(2,n+1):
                 ar = ar * tmp
            return(ar)

#Παίρνουμε από το χρήστη τη δύναμη την οποία θα
#υπολογίσουμε για το άθροισμα
dynami = int(input("Dwse ti dunami tou arithmou: "))

#Υπολογίζουμε το άθροισμα της σειράς
sum = power(1, dynami) + power(2, dynami) + power(3,
dynami) + power(4, dynami) + power(5, dynami)

#Τυπώνουμε το αποτέλεσμα
print("To athroisma otan n = {} \n Σ = 1^n + 2^n + 3^n +
4^n + 5^n = {}".format(dynami, sum))
```

Εκφώνηση 4.5.5:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το γινόμενο της πιο κάτω σειράς:
#Π = 1^n · 2^n · 3^n · 4^n · 5^n


#Δημιουργούμε τη συνάρτηση η οποία υπολογίζει τις δυνάμεις
#δεχόμενη ως είσοδο έναν ακέραιο αριθμό και τη δύναμη αυτού
def power(ar, n):
     if(n == 0):
          return(1)
     if(n == 1):
          return(ar)
     else:
          tmp = ar
          for i in range(2,n+1):
               ar = ar * tmp
          return(ar)

#Παίρνουμε από το χρήστη έναν ακέραιο που είναι η δύναμη
dynami = int(input("Dwse ti dunami tou arithmou: "))

#Αρχικοποιούμε με 1 το γινόμενο
prod = 1

#υπολογίζουμε το γινόμενο
for i in range(1,6):
```

```
        prod = prod * power(i, dynami)

#Τυπώνουμε το αποτέλεσμα του  γινομένου
print("To ginomeno otan n = {} \n  Π = 1^n * 2^n * 3^n *
4^n * 5^n = {}".format(dynami, prod))
```

Εκφώνηση 4.5.6:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το αποτέλεσμα των πιο κάτω
#παραστάσεων: ∑_{n=1}^{K} n·(2n−1)^3,  ∑_{n=1}^{10} n^2·2^n

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό
K = int(input("Dwse arithmo epanalipsewn athroismatos : "))

#Αρχικοποιούμε τη μεταβλητή την οποία θα αποθηκεύουμε το
#άθροισμα
sum1 = 0

#Υπολογίζουμε την πρώτη παράσταση
for i in range(1,K+1):
        a = i
        b = (2*i - 1)**3
        sum1 = sum1 + a*b

#Αρχικοποιούμε τη δεύτερη μεταβλητή για να αποθηκεύουμε το
#αποτέλεσμα της δεύτερης παράστασης
sum2 = 0

#Υπολογίζουμε τη δεύτερη παράσταση
for j in range(1,11):
        c = j**2
        d = 2**j
        sum2 = sum2 + c*d

#Τυπώνουμε τα αποτελέσματα
print("To athroisma tis prwtis seiras einai :{} \n To
athroisma tis deuteris seiras einai :
{}".format(sum1,sum2))
```

Εκφώνηση 4.5.7:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει τους 10 πρώτους όρους των πιο κάτω σειρών και
#να τυπώνει το αποτέλεσμά τους:
#S1 = 1 + 1/2^2 + 1/3^2 + 1/4^2 + ..., S2 = 1 + X + X^2/2^2 + X^3/3^2 + X^4/4^2 + ...

#Αρχικοποιούμε δύο μεταβλητές για να αποθηκεύσουμε τα
#αποτελέσματα των σειρών
sum1 = 0
sum2 = 1

#Παίρνουμε έναν ακέραιο από το χρήστη ο οποίος θα
#χρησιμοποιηθεί για να υπολογίσουμε τη δεύτερη σειρά
X = int(input("Dwse ena akeraio: "))

#Υπολογίζουμε τις σειρές
for i in range(1,11):
     sum1 = sum1 + (1/(i**2))
     sum2 = sum2 + (X**i)/(i**2)

#Τυπώνουμε το αποτέλεσμα
print("To athroisma tis prwtis seiras einai :{} \n To
athroisma tis deuteris seiras einai :
{}".format(sum1,sum2))
```

Εκφώνηση 4.5.8:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο για ένα
#σύνολο Ν τυχαίων ακεραίων αριθμών να υπολογίζει και να
#τυπώνει
#(α) το μέσο όρο τους και
#(β) πόσοι από αυτούς είναι θετικοί, πόσοι αρνητικοί και
#πόσοι μηδέν.

#Εισάγουμε τη βιβλιοθήκη για να παράγουμε τυχαίους αριθμούς
import random as rnd

#Αρχικοποιούμε μία κενή λίστα για να αποθηκεύσουμε τους
#τυχαίους αριθμούς
x = []

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό για το πλήθος
#των τυχαίων αριθμών
```

```
n = int(input("Dwse enan akeraio arithmo gia to plithos twn tuxaiwn arithmwn: "))

#Αποθηκεύουμε τους τυχαίους αριθμούς
for i in range(n):
#Καλούμε τη συνάρτηση randrange() για να παράγουμε
#τυχαίους αριθμούς
     x.append(rnd.randrange(-100, 100, 1))

#Αρχικοποιούμε τις μεταβλητές για να υπολογίσουμε τα
#ζητούμενά μας
count_0 = 0
count_negative = 0
count_positive = 0
sum = 0

#Υπολογίζουμε το άθροισμα και μετράμε πόσοι είναι
#αρνητικοί, θετικοί και μηδενικοί
for j in x:
     sum = sum + j
     if(j == 0):
          count_0 += 1
     elif(j < 0):
          count_negative += 1
     else:
          count_positive += 1

#Τυπώνουμε τα αποτελέσματα
print("Mhdenikoi einai {} arithmoi \nArnitikoi einai {}
arithmoi \nThetikoi einai {} arithmoi kai o M.O. einai
{}".format(count_0, count_negative, count_positive, sum/n))
```

Εκφώνηση 4.5.9:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο για ένα
#σύνολο Ν τυχαίων ακεραίων αριθμών να υπολογίζει και να
#τυπώνει
#(α) το μέσο όρο τους και
#(β) πόσοι από αυτούς είναι άρτιοι και πόσοι περιττοί.

#Εισάγουμε τη βιβλιοθήκη για να παράγουμε τυχαίους αριθμούς
import random as rnd

#Αρχικοποιούμε μία κενή λίστα για να αποθηκεύσουμε τους
#τυχαίους αριθμούς
```

```
x = []

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό για το πλήθος
#των τυχαίων αριθμών
n = int(input("Dwse enan akeraio arithmo gia to plithos twn
tuxaiwn arithmwn: "))

#Αποθηκεύουμε τους τυχαίους αριθμούς
for i in range(n):
#Καλούμε τη συνάρτηση randrange() για να παράγουμε τυχαίους
#αριθμούς
     x.append(rnd.randrange(-100, 100, 1))

#Αρχικοποιούμε τις μεταβλητές μας
count_even = 0
count_odd = 0
sum = 0

#Υπολογίζουμε το άθροισμα και μετράμε πόσοι είναι άρτιοι
#και περιττοί
for j in x:
     sum = sum + j
     if(j % 2 == 0):
          count_even += 1
     else:
          count_odd += 1

#Τυπώνουμε το αποτέλεσμα
print("Artioi einai {} arithmoi \nPerittoi einai {}
arithmoi \nkai o M.O. einai {}".format(count_even,
count_odd,  sum/n))
```

Εκφώνηση 4.5.10:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει την τιμή της πιο κάτω σειράς:
#S₁ = 1! + 2! + 3! + ... + N!

#Δημιουργούμε αναδρομική συνάρτηση η οποία υπολογίζει το
#παραγοντικό ενός αριθμού
def factorial(n):
     if(n == 1):
          return(n)
     else:
          return(n*factorial(n-1))
```

```
#Παίρνουμε από το έναν ακέραιο N
num = int(input("Dwse enan akeraio: "))

#Αρχικοποιούμε τη μεταβλητή μας που θα αποθηκεύσουμε το
#αποτέλεσμα
sum = 0

#Υπολογίζουμε το αποτέλεσμα της σειράς
for i in range(1,num+1):
     sum = sum + factorial(i)

#Τυπώνουμε το αποτέλεσμα
print("To athroisma tis Seiras einai {}".format(sum))
```

Εκφώνηση 4.5.11:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το
#γινόμενο των άρτιων και των περιττών αριθμών από το 1 έως
#το N.


#Εισάγουμε τη βιβλιοθήκη math για να χρησιμοποιήσουμε τις
#συναρτήσεις floor η οποία επιστρέφει τη στρογγυλοποίηση
#προς τα κάτω ενός δεκαδικού αριθμού και τη ceil η οποία
#επιστρέφει τη στρογγυλοποίηση προς τα πάνω ενός δεκαδικού
#αριθμού
from math import floor,ceil

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#ακέραιο αριθμό N και επιστρέφει το άθροισμα των περιττών
#και των άρτιων
def summation(N):
     sum_artioi = 0
     sum_perittoi = 0
     for i in range(1, N+1):
          if(i % 2 == 0):
               sum_artioi = sum_artioi + i
          else:
               sum_perittoi = sum_perittoi + i
     return([sum_artioi, sum_perittoi])
```

```
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#ακέραιο αριθμό και επιστρέφει το γινόμενο των άρτιων και
#των περιττών
def multiply(N):
     mult_artioi = 1
     mult_perittoi = 1
     for i in range(1, N+1):
          if(i % 2 == 0):
               mult_artioi = mult_artioi * i
          else:
               mult_perittoi = mult_perittoi * i
     return([mult_artioi, mult_perittoi])

#Δημιουργούμε συνάρτηση η οποία υπολόγιζε το μέσο όρο
def average(X,N):
     avg = X/N
     return(avg)

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό
arithmos = int(input("dwse arithmo: "))
#Καλούμε τις συναρτήσεις μας και αποθηκεύουμε τα
#αποτελέσματα
athroisma_artiwn, athroisma_perittwn = summation(arithmos)
ginomeno_artiwn,ginomeno_perittwn = multiply(arithmos)

#υπολογίζουμε τους μέσους όρους
if(arithmos % 2 == 0):
     MesosOros_artiwn = average(athroisma_artiwn,
arithmos/2)
     MesosOros_perittwn = average(athroisma_perittwn,
arithmos/2)
else:
     MesosOros_artiwn = average(athroisma_artiwn,
floor(arithmos/2))
     MesosOros_perittwn = average(athroisma_perittwn,
ceil(arithmos/2))

#Τυπώνουμε τα αποτελέσματα
print("gia tous artious")
print("to athroisma  einai {} \nto ginomeno einai {} \no
mesos oros einai {}".format(athroisma_artiwn,
ginomeno_artiwn, MesosOros_artiwn))
print("gia tous perittous")
print("to athroisma  einai {} \nto ginomeno einai {} \no
mesos oros einai {}".format(athroisma_perittwn,
ginomeno_perittwn, MesosOros_perittwn))
```

Εκφώνηση 4.5.12:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει:
#Α)  Την προπαίδεια (μέχρι το 10) ενός συγκεκριμένου
#αριθμού Ν ο οποίος να διαβάζεται από το πληκτρολόγιο.
#Β) Όλη την προπαίδεια των ακεραίων αριθμών από το 1 έως το
#10. Το πρόγραμμα να τυπώνει και τους πολλαπλασιαζόμενους
#αριθμούς.
#Γ) Να δημιουργεί στην οθόνη τα πιο κάτω σχήματα:
'''
   *****            *                    *****
   *****            **                   ****
   *****            ***                  ***
   *****            ****                 **
'''

#Δημιουργούμε συνάρτηση η οποία υπολογίζει και τυπώνει την
#προπαίδεια ενός συγκεκριμένου αριθμού
def propedeia(N):
     for i in range(1,11):
          print("{} * {} = {}".format(N, i, N*i))

#Παίρνουμε έναν ακέραιο αριθμό από το χρήστη
arithmos = int(input("Dwse enan arithmo: "))

#Καλούμε τη συνάρτηση για να υπολογίσουμε την προπαίδεια
propedeia(arithmos)

#Δημιουργούμε συνάρτηση για να υπολογίσουμε και να
τυπώσουμε όλη την προπαίδεια
def synoliki_propedeia():
     for i in range (1,11):
          for j in range(1,11):
               print("{} * {} = {}".format(i, j, i*j))

#Καλούμε τη συνάρτηση της συνολικής προπαίδειας
synoliki_propedeia()

#Δημιουργούμε συνάρτηση για να τυπώσουμε τις παραστάσεις
#από αστεράκια
def asterakia():
     for i in range(1,6):
          for j in range(1,6):
               print("*", end = "")
          print("\t")
```

```
    for i in range(1,6):
        for j in range(1,i+1):
            print("*", end = "")
        print("\t")

    for i in range(5,0,-1):
        for j in range(1,i+1):
            print("*", end = "")
        print("\t")

#Καλούμε τη συνάρτηση
asterakia()
```

Εκφώνηση 4.5.13:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#τη μεταβλητή x και να υπολογίζει την πιο κάτω σειρά.
#Σ = X + X^2/2! + X^3/3! + X^4/4! + ... + X^10/10!

#Δημιουργούμε αναδρομική συνάρτηση η οποία υπολογίζει το
#παραγοντικό ενός αριθμού
def factorial(i):
    if(i==1):
        return(1)
    else:
        return(i*factorial(i-1))

#Παίρνουμε έναν ακέραιο αριθμό από το χρήστη
x = int(input("dwse akeraio arithmo: "))

#Αρχικοποιούμε τη μεταβλητή μας που θα αποθηκεύουμε το
#αποτέλεσμα της σειράς
sum = 0

#Υπολογίζουμε το αποτέλεσμα της σειράς
for i in range(1,11):
    sum = sum + (x**i)/factorial(i)
#Τυπώνουμε το αποτέλεσμα
print(sum)
```

Εκφώνηση 4.5.14:

```
'''Μια φορά κι ένα καιρό, στη μακρινή Κίνα ήταν ένας
αυτοκράτορας που είχε πάθος με τα παιχνίδια - κυρίως τα
επιτραπέζια. Έφτασε όμως μια μέρα που είχε παίξει και είχε
βαρεθεί όσα παιχνίδια υπήρχαν. Διέταξε λοιπόν να του
φτιάξουν ένα παιχνίδι με απλούς κανόνες, το οποίο όμως κάθε
φορά που θα το έπαιζε να είναι διαφορετικό ώστε να μην
βαρεθεί ποτέ. Όποιος κατάφερνε να του φτιάξει ένα τέτοιο
παιχνίδι θα μπορούσε να ζητήσει οποιαδήποτε αμοιβή. Ένας
λοιπόν από τους συμβούλους του σκέφτηκε να δημιουργήσει το
σκάκι. Ο αυτοκράτορας ενθουσιάστηκε και του είπε "Ποια θες
να είναι τώρα η αμοιβή σου; Μήπως θες να παντρευτείς την
κόρη μου και να γίνεις ο διάδοχός μου στο θρόνο;", "Όχι"
απάντησε ο σύμβουλος "κάτι πιο απλό. Θέλω στο πρώτο από τα
64 τετράγωνα που έχει το σκάκι να βάλεις ένα κόκκο ρύζι,
στο δεύτερο 2, στο τρίτο 4, στο τέταρτο 8 κ.ο.κ
διπλασιάζοντας κάθε φορά τον αριθμό των κόκκων από ρύζι. Η
αμοιβή μου θα είναι όλο το ρύζι που θα υπάρχει πάνω στη
σκακιέρα". "Μόνο αυτό;" Είπε ο αυτοκράτορας και διέταξε να
πληρώσουν αμέσως το σύμβουλο. Τελικά όμως το ρύζι που
έπρεπε να δώσουν στο σύμβουλο ήταν τόσο πολύ που ο
αυτοκράτορας έδωσε όλη την περιουσία του για να τον
ξεχρεώσει. (9.223.372.036.854.775.809)

Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να:
Α) υπολογίζει τον αριθμό των κόκκων ρυζιού που πήρε ο
σύμβουλος.
Β) λαμβάνοντας υπόψη ότι κάθε κόκκος ρύζι μπορεί να ζυγίζει
κατά μέσο όρο 0,1 γραμμάρια, να υπολογίζει το βάρος του
ρυζιού σε τόνους.
Γ) θα εμφανίζει το μήνυμα "Ο σύμβουλος πήρε ..... κόκκους
ρύζι που αντιστοιχούν με ..... τόνους".
'''


#Αρχικοποιούμε τη μεταβλητή μας που θα αποθηκεύσουμε το
#αποτέλεσμα
sum = 0

#Υπολογίζουμε τους κόκκους ρυζιού
for i in range(63):
     sum = sum + 2**i

#υπολογίζουμε το βάρος τους
varos = (sum * 0.1)/ 1000000
```

```
#Τυπώνουμε το αποτέλεσμα
print("o symvoulos pire", sum," kokkous ruzi pou
antistoixoun me ",varos," tonous")
```

4.6 Ασκήσεις με εντολή while

Εκφώνηση 4.6.1:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το
#γινόμενο των αριθμών από το 1 έως το N.

#Παίρνουμε απο το χρήστη έναν ακέραιο αριθμί
arithmos = int(input("dwse arithmo: "))

#Αρχικοποιούμε τις μεταβλητές μας
mo = 0
athroisma = 0
ginomeno = 1

#Υπολογίζουμε το άθροισμα και το γινόμενο
i = 1
while(i <= arithmos):
     athroisma = athroisma + i
     ginomeno = ginomeno * i
     i = i + 1

#Υπολογίζουμε το μέσο όρο
mo = athroisma / arithmos

#Τυπώνουμε το αποτέλεσμα
print("to athroisma einai: {} \nto ginomeno einai: {} \nkai
o mesos oros einai: {} ".format(athroisma, ginomeno, mo))
```

Εκφώνηση 4.6.2:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το
#γινόμενο των περιττών αριθμών από το 1 έως το N.
```

```
#Εισάγουμε τη βιβλιοθήκη math για να χρησιμοποιήσουμε την
#εντολή ceil η οποία στρογγυλοποιεί ένα δεκαδικό αριθμό
#προς τα πάνω
from math import ceil

#παίρνω από το χρήστη έναν ακέραιο αριθμό
arithmos = int(input("dwse arithmo: "))

#Αρχικοποιούμε τις μεταβλητές μας που θα αποθηκεύσουμε τα
αποτελέσματα
mo = 0
athroisma = 0
ginomeno = 1

#Υπολογίζουμε το άθροισμα, το γινόμενο
i = 1
while(i <= arithmos):
     if(i % 2 == 1):
          athroisma = athroisma + i
          ginomeno = ginomeno * i
     i = i + 1

#Υπολογίζουμε το μέσο όρο
mo = athroisma / ceil(arithmos/2)

#Τυπώνω τα αποτελέσματα
print("to athroisma einai: {} \nto ginomeno einai: {} \nkai
o mesos oros einai: {} ".format(athroisma, ginomeno, mo))
```

Εκφώνηση 4.6.3:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το να διαβάζει ένα
#αριθμό (ar) και τη δύναμη του αριθμού (n) και να
#υπολογίζει και να τυπώνει το αποτέλεσμά τους (π.χ. ar^n).

#Παίρνουμε από το χρήστη έναν αριθμό
ar = int(input("dwse arithmo: "))
#Παίρνουμε από το χρήστη τη δύναμη του  αριθμού
n = int(input("dwse dunamh arithmou: "))

#Αρχικοποιούμε τις μεταβλητές μας για να αποθηκεύσουμε το
#αποτέλεσμα
ginomeno = 1
i = 1
#Υπολογίζουμε το γινόμενο
```

```
while (i <= n ):
     ginomeno = ginomeno * ar
     i = i+1

#Τυπώνουμε το αποτέλεσμα
print("to ginomeno isoute: ",ginomeno)
```

Εκφώνηση 4.6.4:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το άθροισμα της πιο κάτω σειράς:
#Σ = 1^n + 2^n + 3^n + 4^n + 5^n

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό
arithmos = int(input("dwse arithmo: "))

#Αρχικοποιούμε τις μεταβλητές μας
sum = 0
i = 1
#Υπολογίζουμε το άθροισμα της σειράς
while(i <= 5):
     sum = sum + i**arithmos
     i = i+1

#Τυπώνουμε το αποτέλεσμα
print("to athroisma Σ = 1^{0} + 2^{0} + 3^{0} + 4^{0} +
5^{0} = {1}".format(arithmos,sum))
```

Εκφώνηση 4.6.5:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το γινόμενο της πιο κάτω σειράς:
#Π = 1^n · 2^n · 3^n · 4^n · 5^n

#Παίρνουμε από το χρήστη έναν αριθμό
arithmos = int(input("dwse arithmo: "))
#Αρχικοποιούμε τις μεταβλητές μας
ginomeno = 1
i = 1

#Υπολογίζουμε το γινόμενο
while(i <= 5):
     ginomeno = ginomeno * i**arithmos
```

```
        i = i+1

#Τυπώνουμε το αποτέλεσμα
print("to ginomeno Π = 1^{0} * 2^{0} * 3^{0} * 4^{0} *
5^{0} = {1}".format(arithmos,ginomeno))
```

Εκφώνηση 4.6.6:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το αποτέλεσμα των πιο κάτω
#παραστάσεων:∑_{n=1}^{K} n·(2n−1)^3    ∑_{n=1}^{10} n^2·2^n

#Παίρνουμε από το χρήστη έναν αριθμό
K = int(input("Dwse enan akeraio arithmo: "))

#Αρχικοποιούμε τις μεταβλητές μας για την πρώτη σειρά
sum1 = 0
i = 1

#Υπολογίζουμε το άθροισμα της πρώτης σειράς
while(i <= K):
        sum1 = sum1 + i*((2*i-1)**3)
        i = i +1

#Αρχικοποιούμε τις μεταβλητές μας για την δεύτερη σειρά
j = 1
sum2 = 0

#Υπολογίζουμε το άθροισμα της δεύτερης σειράς
while(j <= 10):
        sum2 = sum2 + (j**2)*(2**j)
        j = j + 1

#Τυπώνουμε τα αποτελέσματα των σειρών
print("To athroisma tis prwtis seiras einai : ", sum1)
print("To athroisma tis deuteris seiras einai : ", sum2)
```

Εκφώνηση 4.6.7:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει τους 10 πρώτους όρους των πιο κάτω σειρών και
#να τυπώνει το αποτέλεσμά τους:S_1 = 1 + 1/2^2 + 1/3^2 + 1/4^2,
#S_2 = 1 + X + X^2/2^2 + X^3/3^2 + X^4/4^2
```

```
#Παίρνουμε από το χρήστη έναν αριθμό
X = int(input("Dwse enan akeraio arithmo: "))

#Αρχικοποιούμε τις μεταβλητές μας για την πρώτη σειρά
sum1 = 0
i = 1

#Υπολογίζουμε το άθροισμα της πρώτης σειράς
while(i <= 10):
     sum1 = sum1 + (1/(i**2))
     i = i +1

#Αρχικοποιούμε τις μεταβλητές μας για την δεύτερη σειρά
j = 1
sum2 = 1

#Υπολογίζουμε το άθροισμα της δεύτερης σειράς
while(j < 10):
     sum2 = sum2 + (X**j)/(j**2)
     j = j + 1

#Τυπώνουμε τα αποτελέσματα των σειρών
print("To athroisma tis prwtis seiras einai : ", sum1)
print("To athroisma tis deuteris seiras einai : ", sum2)
```

Εκφώνηση 4.6.8:

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο για ένα
#σύνολο Ν τυχαίων ακεραίων αριθμών να υπολογίζει και να
#τυπώνει
#(α) το μέσο όρο τους και
#(β) πόσοι από αυτούς είναι θετικοί, πόσοι αρνητικοί και
#πόσοι μηδέν.

#Εισάγουμε τη βιβλιοθήκη για να παράγουμε τυχαίους αριθμούς
import random as rnd

#Αρχικοποιούμε τη λίστα που θα αποθηκεύουμε τους τυχαίους
#αριθμούς
x = []

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό που θα είναι
#το πλήθος των τυχαίων αριθμών που θα δημιουργήσουμε
n = int(input("Dwse enan akeraio arithmo gia to plithos twn
tuxaiwn arithmwn: "))
for i in range(n):
```

```
        #Δημιουργούμε τους τυχαίους αριθμούς
        x.append(rnd.randrange(-100, 100, 1))

#Αρχικοποιούμε τις μεταβλητές μας
sum = 0
count_thetikwn = 0
count_arnitikwn = 0
count_mhdenikwn = 0
j = 0

#Υπολογίζουμε πόσοι αριθμοί είναι θετικοί, αρνητικοί και
#μηδενικοί
while(j < n):
        sum = sum + x[j]
        if(x[j]>0):
                count_thetikwn = count_thetikwn + 1
        elif(x[j]<0):
                count_arnitikwn = count_arnitikwn + 1
        else:
                count_mhdenikwn = count_mhdenikwn + 1
        j = j + 1

#Υπολογίζουμε το μέσο όρο
mo =  sum/n

#Τυπώνουμε το αποτέλεσμα
print("oi thetikoi einai: {} \noi arnhtikoi einai: {}
\nmhden einai: {} \nkai o mesos oros einai: {}
".format(count_thetikwn,count_arnitikwn,count_mhdenikwn,mo)
)
```

Εκφώνηση 4.6.9:

```
#Να γραφεί πρόγραμμα σε γλώσσα C το οποίο για ένα σύνολο N
#τυχαίων ακεραίων αριθμών να υπολογίζει και να τυπώνει
#(α) το μέσο όρο τους και
#(β) πόσοι από αυτούς είναι άρτιοι και πόσοι περιττοί.

#Εισάγουμε τη βιβλιοθήκη για να παράγουμε τυχαίους αριθμούς
import random as rnd

#Αρχικοποιούμε τη λίστα που θα αποθηκεύουμε τους τυχαίους
#αριθμούς
x = []
```

```
#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό που θα είναι
#το πλήθος των τυχαίων αριθμών που θα δημιουργήσουμε
n = int(input("Dwse enan akeraio arithmo gia to plithos twn
tuxaiwn arithmwn: "))
for i in range(n):
     #Δημιουργουμε τους τυχαίους αριθμούς
     x.append(rnd.randrange(1, 100, 1))

#Αρχικοποιούμε τις μεταβλητές μας
sum = 0
count_artiwn = 0
count_perittwn = 0
j = 0

#Υπολογίζουμε πόσοι αριθμοί είναι άρτιοι και πόσοι είναι
#περιττοί
while( j<n):
     sum = sum + x[j]
     if(x[j] % 2 == 0):
          count_artiwn = count_artiwn + 1
     else:
          count_perittwn = count_perittwn + 1
     j = j + 1

#Υπολογίζουμε το μέσο όρο
mo =  sum/n

#Τυπώνουμε το αποτέλεσμα
print("oi artioi einai: {} \noi perittoi einai: {}  \nkai o
mesos oros einai: {}
".format(count_artiwn,count_perittwn,mo))
```

Εκφώνηση 4.6.10:

```
#Να γραφεί πρόγραμμα σε γλώσσα C το οποίο να υπολογίζει και
#να τυπώνει την τιμή της πιο κάτω σειράς:
#S1 = 1! + 2! + 3! + ... + N!

#Δημιουργούμε αναδρομική συνάρτηση η οποία δέχεται σαν
#είσοδο έναν αριθμό και υπολογίζει το παραγοντικό του
def factorial(n):
     if (n == 1):
          return(1)
     else:
          return(n * factorial(n-1))
```

```
#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό
arithmos = int(input("dwse akeraio arithmo: "))

#Αρχικοποιούμε τις μεταβλητές μας
sum = 0
count = 1

#Υπολογίζουμε το άθροισμα της σειράς
while(count <= arithmos):
     sum = sum + factorial(count)
     count = count + 1

#Τυπώνουμε το αποτέλεσμα
print("to athroisma einai: ",sum)
```

Εκφώνηση 4.6.11

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το
#γινόμενο των άρτιων και των περιττών αριθμών από το 1 έως
#το N.

#Από τη βιβλιοθήκη math εισάγουμε τις συναρτήσεις floor και
#ceil οι οποίες στρογγυλοποιούν έναν αριθμό προς τα κάτω
#και προς τα πάνω αντίστοιχα
from math import floor, ceil

#Παίρνουμε απο το χρήστη έναν αριθμό
arithmos = int(input("dwse arithmo: "))

#Αρχικοποιούμε τις μεταβλητές μας
mo_artiwn = 0
mo_perittwn = 0
athroisma_artiwn = 0
athroisma_perittwn = 0
ginomeno_artiwn = 1
ginomeno_perittwn = 1

#Υπολογίζουμε τα αθροίσματα και τα γινόμενα
i = 1
while(i <= arithmos):
     if(i % 2 == 0):
          athroisma_artiwn = athroisma_artiwn + i
          ginomeno_artiwn = ginomeno_artiwn * i
     else:
          athroisma_perittwn = athroisma_perittwn + i
```

```
            ginomeno_perittwn = ginomeno_perittwn * i
        i = i + 1

#Υπολογίζουμε τους μέσους όρους
if(arithmos % 2 == 0):
        mo_artiwn = athroisma_artiwn / (arithmos/2)
        mo_perittwn = athroisma_perittwn / (arithmos/2)
else:
        mo_artiwn = athroisma_artiwn / (floor(arithmos/2))
        mo_perittwn = athroisma_perittwn / (ceil(arithmos/2))

#τυπώνουμε τα αποτελέσματα
print("to athroisma twn artiwn einai: {} \nto ginomeno twn
artiwn einai: {} \nkai o mesos oros twn artiwn einai: {}
".format(athroisma_artiwn, ginomeno_artiwn, mo_artiwn))
print("to athroisma twn perittwn einai: {} \nto ginomeno
twn perittwn einai: {} \nkai o mesos oros twn perittwn
einai: {} ".format(athroisma_perittwn, ginomeno_perittwn,
mo_perittwn))
```

Εκφώνηση 4.6.12

```
'''
Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να υπολογίζει
και να τυπώνει:
Α) Την προπαίδεια (μέχρι το 10) ενός συγκεκριμένου αριθμού
Ν ο οποίος να διαβάζεται από το πληκτρολόγιο.
Β) Όλη την προπαίδεια των ακεραίων αριθμών από το 1 έως το
10. Το πρόγραμμα να τυπώνει και τους πολλαπλασιαζόμενους
αριθμούς.
Γ) Να δημιουργεί στην οθόνη τα πιο κάτω σχήματα:
   *****           *                *****
   *****           **               ****
   *****           ***              ***
   *****           ****             **
   *****           *****            *
'''

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#αριθμό και τυπώνει την προπαίδεια του αριθμού
def propedeia(N):
        i = 1
        while(i <= 10):
                print("{} * {} = {}".format(N, i, N*i))
                i = i + 1
```

```
#Παίρνουμε από το χρήστη έναν αριθμό
arithmos = int(input("Dwse enan arithmo: "))

#Καλούμε τη συνάρτησης για την προπαίδεια του αριθμού
propedeia(arithmos)

#Δημιουργούμε συνάρτηση η οποία τυπώνει όλη την προπαίδεια
#απο το 1-10
def synoliki_propedeia():
     i = 1
     while(i <= 10):
          j = 1
          while(j <= 10):
               print("{} * {} = {}".format(i, j, i*j))
               j = j + 1
          i = i + 1

#Καλούμε τη συνάρτηση της προπαίδειας
synoliki_propedeia()

#Δημιουργούμε συνάρτηση που δημιουργεί τις παραστάσεις από
#αστεράκια
def asterakia():
     i = 1
     while(i <= 6):
          j = 1
          while(j <= 6):
               print("*", end = "")
               j = j + 1
          i = i + 1
          print("\t")
     print("\n")

     i = 1
     while(i <= 5):
          j = 1
          while(j <= i):
               print("*", end = "")
               j = j + 1
          i= i + 1
          print("\t")
     print("\n")

     i = 5
     while(i >= 1):
```

```
            j = 1
            while(j <= i):
                 print("*", end = "")
                 j = j + 1
            i= i - 1
            print("\t")

#Καλούμε τη συνάρτηση
asterakia()
```

Εκφώνηση 4.6.13

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#τη μεταβλητή x και να υπολογίζει την πιο κάτω σειρά.
```
#$S = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \frac{x^8}{8!} + \frac{x^9}{9!} + \frac{x^{10}}{10!}$

```
#Παίρνουμε από το χρήστη έναν αριθμό
arithmos = int(input("Dwse enan arithmo: "))

#Δημιουργούμε αναδρομική συνάρτηση που δέχεται ως είσοδο
#έναν αριθμό και επιστρέφει το παραγοντικό αυτού του
#αριθμού
def factorial(N):
     if(N == 1):
          return(1)
     else:
          return(N*factorial(N-1))

#Αρχικοποιούμε τις μεταβλητές μας
i = 1
sum = 0

#Υπολογίζουμε το άθροισμα της σειράς
while(i <= 10):
     sum = sum + ((arithmos**i) / factorial(i))
     i = i + 1

#Τυπώνουμε το αποτέλεσμα της σειράς
print("To athroisma Σ = {0} + {0}^2/2!+ {0}^3/3!+ {0}^4/4!+
{0}^5/5!+ {0}^6/6!+ {0}^7/7!+ {0}^8/8!+ {0}^9/9!+
{0}^10/10! = {1}".format(arithmos, sum))
```

Εκφώνηση 4.6.14

```
'''
Σε κάποια εξεταστική δοκιμασία ένα γραπτό αξιολογείται από
δύο βαθμολογητές στη βαθμολογική κλίμακα [0, 100].
Αν η διαφορά μεταξύ των βαθμολογιών του α΄ και του β΄
βαθμολογητή είναι μικρότερη ή ίση των 20 μονάδων της
παραπάνω κλίμακας,
ο τελικός βαθμός είναι ο μέσος όρος των δύο βαθμολογιών.

Αν η διαφορά μεταξύ των βαθμολογιών του α΄ και του β΄
βαθμολογητή είναι μεγαλύτερη από 20 μονάδες, το γραπτό
δίνεται για αναβαθμολόγηση σε τρίτο βαθμολογητή. Ο τελικός
βαθμός του γραπτού προκύπτει τότε από τον μέσο όρο των
τριών βαθμολογιών.

Να αναπτύξετε πρόγραμμα σε γλώσσα C το οποίο, αφού ελέγξει
την εγκυρότητα των βαθμών στην βαθμολογική κλίμακα [0,
100], να υλοποιεί την παραπάνω διαδικασία εξαγωγής τελικού
βαθμού και να εμφανίζει τον τελικό βαθμό του γραπτού στην
εικοσαβάθμια κλίμακα.

Παρατήρηση: Να θεωρήσετε ότι όλες οι ποσότητες εκφράζονται
ως πραγματικοί αριθμοί.'''


#Παίρνουμε το βαθμό από τον πρώτο αξιολογητή
vathmos_aksiologiti_1 = int(input("Dwse vathmo apo [0-100]:
"))

#Εξετάζουμε αν ο βαθμός είναι έγκυρος
while(vathmos_aksiologiti_1 < 0 or vathmos_aksiologiti_1 >
100):
     vathmos_aksiologiti_1 = int(input("Ksanadwse vathmo
apo [0-100]: "))

#Παίρνουμε το βαθμό από τον δεύτερο αξιολογητή
vathmos_aksiologiti_2 = int(input("Dwse vathmo apo [0-100]:
"))

#Εξετάζουμε αν ο βαθμός είναι έγκυρος
while(vathmos_aksiologiti_2 < 0 or vathmos_aksiologiti_2 >
100):
     vathmos_aksiologiti_2 = int(input("Ksanadwse vathmo
apo [0-100]: "))

#Αρχικοποιούμε τη μεταβλητή μας
```

```
mo = 0

#Εξετάζουμε πόση είναι η διαφορά του πρώτου αξιολογητή από
#το δεύτερο
if(abs(vathmos_aksiologiti_1 - vathmos_aksiologiti_2) >
20):
     print("H diafora twn dyo prwtwn aksiologitwn einai
megalyteri apo 20 kai anatithetai se triton aksiologhth")

     #Αν είναι μεγαλύτερη από 20 μονάδες παίρνουμε και από
#τον τρίτο αξιολογητή τη βαθμολογία
     vathmos_aksiologiti_3 = int(input("Dwse vathmo apo [0-
100]: "))



#Εξετάζουμε αν ο βαθμός είναι έγκυρος
     while(vathmos_aksiologiti_3 < 0 or
vathmos_aksiologiti_3 > 100):
          vathmos_aksiologiti_3 = int(input("Ksanadwse
vathmo apo [0-100]: "))
     #Βρίσκουμε το μέσο όρο
     mo = (vathmos_aksiologiti_1 + vathmos_aksiologiti_2 +
vathmos_aksiologiti_3) / 3
else:
#αλλιώς αν η διαφορά είναι μικρότερη από 20 μονάδες
#βρίσκουμε το μέσο όρο
     mo = (vathmos_aksiologiti_1+vathmos_aksiologiti_2) / 2

#Τυπώνουμε το αποτέλεσμα
print("O mesos oros einai: ", mo)

#Να γραφεί πρόγραμμα σε γλώσσα C το οποίο να διαβάζει ένα μονοδιάστατο
πίνακα ακέραιων αριθμών a με 5 θέσεις
#και υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το γινόμενο
των στοιχείων του πίνακα a.

#Αρχικοποιούμε τη λίστα που θα αποθηκεύσουμε τους αριθμούς
pinakas = []

#Παίρνουμε από το χρήστη τους αριθμούς που θα αποθηκεύσουμε στον πίνακα
for i in range (5):
     print("Dwse {}o stoixeio".format(i+1))
     pinakas.append(int(input("dwse stoixeio: ")))

#Αρχικοποιούμε τις μεταβλητές μας
athroisma = 0
ginomeno = 1
```

```
#Υπολογίζουμε το άθροισμα και το γινόμενο των στοιχείων του πίνακα
for i in range (5):
      athroisma = athroisma + pinakas[i]
      ginomeno = ginomeno * pinakas[i]

#Υπολογίζουμε το μέσο όρο
mo = athroisma / 5

#Τυπώνουμε το αποτέλεσμα
print("to athroisma einai ",athroisma, " to ginomeno einai ",ginomeno,
" kai o mesos oros einai ",mo)
```

4.7 Ασκήσεις με μονοδιάστατους πίνακες

Εκφώνηση 4.7.1

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα μονοδιάστατο πίνακα ακέραιων αριθμών a με 5 θέσεις
#και υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και
#το γινόμενο των στοιχείων του πίνακα a.

#Αρχικοποιούμε τη λίστα που θα αποθηκεύσουμε τους #αριθμούς
pinakas = []

#Παίρνουμε από το χρήστη τους αριθμούς που θα αποθηκεύσουμε
#στον πίνακα
for i in range (5):
     print("Dwse {}o stoixeio".format(i+1))
     pinakas.append(int(input("dwse stoixeio: ")))

#Αρχικοποιούμε τις μεταβλητές μας
athroisma = 0
ginomeno = 1

#Υπολογίζουμε το άθροισμα και το γινόμενο των στοιχείων του
#πίνακα
for i in range (5):
     athroisma = athroisma + pinakas[i]
     ginomeno = ginomeno * pinakas[i]

#Υπολογίζουμε το μέσο όρο
mo = athroisma / 5

#Τυπώνουμε το αποτέλεσμα
```

```
print("to athroisma einai ",athroisma, " to ginomeno einai
",ginomeno, " kai o mesos oros einai ",mo)
```

Εκφώνηση 4.7.2

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα μονοδιάστατο πίνακα ακέραιων αριθμών a με 5 θέσεις
#και υπολογίζει και να τυπώνει το μέγιστο και το ελάχιστο
#στοιχείο του πίνακα Α και τις θέσεις τους.

#Αρχικοποιούμε τη λίστα που θα αποθηκεύσουμε τους #αριθμούς
pinakas = []

#Παίρνουμε από το χρήστη τους αριθμούς που θα αποθηκεύσουμε
#στον πίνακα
for i in range (5):
     print("Dwse {}o stoixeio".format(i+1))
     pinakas.append(int(input("dwse stoixeio: ")))

#Υπολογίζουμε το μέγιστο και το ελάχιστο στοιχείο του
#πίνακα αλλά και τις θέσεις τους
for i in range (1,5):
     max = pinakas[0]
     min = pinakas[0]
     if (pinakas[i] > max):
          max = pinakas[i]
     if (pinakas[i] < min):
          min = pinakas[i]

#Τυπώνουμε το αποτέλεσμα
print(" o megistos arithmos einai ",max, " o elaxistos
arithmos einai ",min)
```

Εκφώνηση 4.7.3

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα μονοδιάστατο πίνακα ακέραιων αριθμών a με 5 θέσεις
#και υπολογίζει και να τυπώνει το μέσο όρο, και πόσοι από
#τους αριθμούς είναι μεγαλύτεροι από τον πιο πάνω μέσο όρο.

#Αρχικοποιούμε τη λίστα που θα αποθηκεύσουμε τους #αριθμούς
pinakas = []
```

```
#Παίρνουμε από το χρήστη τους αριθμούς που θα αποθηκεύσουμε
#στον πίνακα
for i in range (5):
     print("Dwse {}o stoixeio".format(i+1))
     pinakas.append(int(input("dwse stoixeio: ")))

#Αρχικοποιούμε τη μεταβλητή μας
athroisma = 0
#Υπολογίζουμε το άθροισμα
for i in range (5):
     athroisma = athroisma + pinakas[i]

#Υπολογίζουμε το μέσο όρο
mo = athroisma / 5

#Αρχικοποιούμε ένα μετρητή
count = 0

#Υπολογίζουμε πόσοι αριθμοί είναι πάνω από το μέσο όρο
for i in range (5):
     if(pinakas[i] > mo):
          count = count + 1

#Τυπώνουμε το αποτέλεσμα
print("o mesos oros einai ",mo, " kai oi arithmoi pou einai
megaliteroi apo ton meso oro einai ",count)
```

Εκφώνηση 4.7.4

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#δημιουργεί και να εμφανίζει μονοδιάστατο πίνακα Α 15
#θέσεων σύμφωνα με την σχέση:
#A_i = { x * i,     x < 10
#      { 2 * i,   διαφορετικά
#Αρχικοποιούμε μία κενή λίστα στην οποία θα αποθηκεύσουμε
#τα στοιχεία μας
A = []

#Αποθηκεύουμε τους αριθμούς στη λίστα σύμφωνα με τη δοθείσα
#σχέση
for i in range(15):
     x = int(input("dwse enan akeraio arithmo: "))
     if (x < 10):
          A.append(x*i)
     else:
          A.append(2*i)
```

```
#Τυπώνουμε τον πίνακα
print(A)
```

Εκφώνηση 4.7.5

```
#Να γραφεί πρόγραμμα σε γλώσσα C το οποίο να διαβάζει δύο
#μονοδιάστατους πίνακες με όνομα a και b 5 θέσεων
#και να εμφανίζει το πίνακα c ο οποίος είναι το άθροισμα
#των a και b.

#Αρχικοποιώ τις δύο λίστες που θα αποθηκεύονται οι αριθμοί
#που λαμβάνουμε από το χρήστη
a = []
b = []
#Η τρίτη λίστα στη οποία θα αποθηκεύω το άθροισμα των άλλων
#δύο
c = []

#Παίρνουμε από το χρήστη τα στοιχεία της πρώτης λίστας
for i in range(5):
     x = int(input("Dwse enan akeraio arithmo gia ton
pinaka a: "))
     a.append(x)

#Παίρνουμε από το χρήστη τα στοιχεία της δεύτερης λίστας
for i in range(5):
     y = int(input("Dwse enan akeraio arithmo gia ton
pinaka b: "))
     b.append(y)


#Υπολογίζουμε το άθροισμα των δύο λιστών
for i in range(5):
     c.append(a[i] + b[i])

#Τυπώνουμε το αποτέλεσμα
print(c)
```

Εκφώνηση 4.7.6

```
#Να γραφεί πρόγραμμα σε γλώσσα C το οποίο να κατασκευάζει
#δύο μονοδιάστατους πίνακες με όνομα odd και even
```

```
#αντίστοιχα ως εξής. Ο πρώτος να περιέχει τους περιττούς
#αριθμούς από το 1 ως το 10 και ο δεύτερος τους άρτιους
#αριθμούς.

#Αρχικοποιούμε τις δύο λίστες για να αποθηκεύσουμε τους
#άρτιους και τους περιττούς
odd = []
even = []

#Υπολογίζουμε ποιοί αριθμοί είναι άρτιοι και ποιοι περιττοί
#και τους αποθηκεύουμε στις λίστες
for i in range (1,11):
     if (i%2 == 1):
          odd.append(i)
     else:
          even.append(i)

#Τυπώνουμε τις λίστες
print(odd,"\n",even)
```

Εκφώνηση 4.7.7

```
#Να γραφεί πρόγραμμα σε γλώσσα C το οποίο θα διαβάζει το
#μονοδιάστατο πίνακα Α 10 θέσεων με στοιχεία [9  8  0  7  0
#6  0  5  0  4] και θα δημιουργεί το πίνακα Β 10 θέσεων που
#να περιέχει τα στοιχεία του Α με την ίδια σειρά,
#έχοντας όμως τα μηδενικά μαζεμένα στο τέλος του. π.χ. [9
#8  7  6  5  4  0  0  0  0].


#Δημιουργούμε τη λίστα Α να περιέχει συγκεκριμένα στοιχεία
A = [9, 8, 0, 7, 0, 6, 0, 5, 0, 4]

#Αρχικοποιούμε μία καινούρια λίστα που θα αποθηκεύσουμε την
#τροποποιημένη λίστα Α
B =[]

#Αρχικοποιούμε ένα μετρητή
counter = 0

#Αρχικά αποθηκεύουμε όλους τους αριθμούς που είναι διάφοροι
#του μηδενός στην καινούρια λίστα
for i in A:
     if (i != 0):
          B.append(i)
```

```
        else:
            #Μετράμε πόσα μηδενικά περιέχει η λίστα
            counter = counter + 1

#Προσθέτουμε τέλος όσα μηδενικά μετρήσαμε στο τέλος της
#λίστας
for i in range(counter):
    B.append(0)

#Τυπώνουμε την τελική μας λίστα
print(B)
```

Εκφώνηση 4.7.8

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο θα διαβάζει
#τις ηλικίες 5 φοιτητών και να τις καταχωρεί σε έναν
#μονοδιάστατο πίνακα Α. Έπειτα να υπολογίζει και να
#τυπώνει:
# Το μέσο όρο ηλικίας των φοιτητών.
# Το φοιτητή με τη μικρότερη και μεγαλύτερη ηλικία.
# Το πλήθος των φοιτητών που έχουν ηλικία άνω των 25 ετών.

#Αρχικοποιούμε τη λίστα που θα αποθηκεύουμε τις ηλικίες
A = []

#Παίρνουμε από το χρήστη τις ηλικίες
for i in range(5):
    x = int(input("dwse ilikia foitith: "))
    A.append(x)

#Αρχικοποιούμε τις μεταβλητές μας
sum = 0
counter = 0

#Υπολογίζουμε το άθροισμα των ηλικιών και μετράμε το πλήθος
#των φοιτητών με ηλικία άνω των 25
for i in A:
    sum = sum + i
    if (i > 25):
        counter = counter + 1


#Υπολογίζουμε το μέσο όρο
mo = sum/5
```

```
#Αρχικοποιούμε τις μεταβλητές μας για να βρούμε το min και
max
min = A[0]
max = A[0]

#Υπολογίζουμε το min και max
for i in range(1,5):
     if (min > A[i]):
          min = A[i]
     if (max < A[i]):
          max = A[i]

#Τυπώνουμε το αποτέλεσμα
print("o mesos oros hlikias twn foititwn einai: ",mo,"\nh
mikroterh hlikia einai: ",min,"\nh megalyterh hlikia einai:
",max,"\nkai oi foithtes pou einai panv apo 25 einai:
",counter)
```

Εκφώνηση 4.7.9

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο θα διαβάζει
#τις βαθμολογίες για ένα συγκεκριμένο μάθημα 5 φοιτητών σε
#ένα μονοδιάστατο πίνακα Α. Θα υπολογίζει και θα εμφανίζει
#το πλήθος των φοιτητών οι οποίοι έχουν βαθμό λιγότερο από
#5 και τους φοιτητές που αρίστευσαν και έχουν βαθμό
#μεγαλύτερο ή ίσο από 8,5.

#Αρχικοποιούμε τη λίστα την οποία θα αποθηκεύουμε τις
βαθμολογίες
A = []

#Διαβάζουμε τα στοιχεία από το χρήστη
for i in range(5):
     x = float(input("dwse bathmologia: "))
     A.append(x)

#Αρχικοποιούμε τις μεταβλητές μας
counterless = 0
counterexcellent = 0

#Υπολογίζουμε ποιοι έχουν βαθμό μικρότερο του 5 και
μεγαλύτερο του 8.5
for i in A:
     if (A[i] < 5):
          counterless = counterless + 1
```

```
     elif (A[i] >= 8.5):
          counterexcellent = counterexcellent +1

#Τυπώνουμε το αποτέλεσμα
print("to plithos twn foititwn pou kopikan einai:
",counterless,"\nto plithos twv foititwn poy aristeusan
einai: ",counterexcellent)
```

Εκφώνηση 4.7.10

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο:
# να διαβάζει τον αριθμό μητρώου και τη βαθμολογία 5
#φοιτητών. Να εμφανίζει τη βαθμολογία και τον αριθμό
#μητρώου των φοιτητών που έχουν βαθμό μεγαλύτερο ή ίσο του
#8. Να εμφανίζει τον αριθμό μητρώου του φοιτητή με την
#υψηλότερη βαθμολογία.

#Αρχικοποιούμε ένα λεξικό για να αποθηκεύσουμε τους ΑΜ και
#τις Βαθμολογίες
A = dict()

#Παίρνουμε απο το χρήστη το ΑΜ του φοιτητή και τη
#βαθμολογία του
for i in range(5):
     key = int(input("Dwse to AM tou foiththh: "))
     x = float(input("dwse bathmologia: "))
     A[key] = x

#Αρχικοποιούμε τις μεταβλητές μας
counterless = 0
counterexcellent = 0
max = 0
am = 0

#Υπολογίζουμε ποιος φοιτητής έχει τη μεγαλύτερη βαθμολογία
#και εμφανίζουμε ποιοι φοιτητές έχουν βαθμολογία >= 8
for k in A:
     if (A[k] >= 8.0):
          print("\n AM = {} me vathmo = {}".format(k,A[k]))

     if(max < A[k]):
          max = A[k]
          am = k

#Τυπώνουμε το αποτέλεσμα
```

```
print("AM = {} me vathmo = {} exei ti megisti
vathmologia".format(am,max))
```

Εκφώνηση 4.7.11

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#κατασκευάζει δύο μονοδιάστατους πίνακες με όνομα ODD και
#EVEN αντίστοιχα ως εξής. Ο πρώτος να περιέχει τους
#περιττούς αριθμούς από το 1 ως το 50 και ο δεύτερος τους
#άρτιους αριθμούς.

#Αρχικοπούμε δύο πίνακες για να αποθηκεύσουμε τους άρτιους
#και τους περιττούς
Even =[]
Odd = []

#Υπολογίζουμε ποιοί αριθμοί είναι άρτιοι και ποιοί περιττοί
#και τους αποθηκεύουμε στους πίνακες
for i in range(1,51):
     if(i % 2 == 0):
          Even.append(i)
     else:
          Odd.append(i)

#Τυπώνουμε τους πίνακες
print("Even \n", Even)
print("Odd \n", Odd)
```

Εκφώνηση 4.7.12

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα μονοδιάστατο πίνακα ακέραιων αριθμών Α με 10 θέσεις
#και να υπολογίζει και να τυπώνει το μέγιστο και το
#ελάχιστο στοιχείο του πίνακα Α και τις θέσεις τους.

#Αρχικοποιούμε τον πίνακα
A = []

#Παίρνουμε από το χρήστη τους αριθμούς και τους
#αποθηκεύουμε στον πίνακα
for i in range(10):
     x = int(input("Dwse enan akeraio arithmo: "))
     A.append(x)

#Αρχικοποιούμε τις μεταβλητές μας
```

```
indx_max = 0
max = 0
indx_min = 0
min = 0

#Υπολογίζουμε τα max, min και τις θέσεις τις οποίες
#βρίσκονται
for j in range(len(A)):
     if(max < A[j]):
          max = A[j]
          indx_max = j
     if(min > A[j]):
          min = A[j]
          indx_min = j

#Τυπώνουμε το αποτέλεσμα
print("O megistos arithnmos einai: {}  kai i thesi tou
einai: {}".format(max, indx_min))
print("O elaxistos arithnmos einai: {}  kai i thesi tou
einai: {}".format(min, indx_min))
```

Εκφώνηση 4.7.13

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#δημιουργεί και να εμφανίζει μονοδιάστατο πίνακα Α 100
#θέσεων σύμφωνα με την σχέση:
```

$$\#A_i = \begin{cases} x * i, & x < 10 \\ 2 * i, & διαφορετικά \end{cases}$$

```
#Αρχικοποιούμε τον πίνακα
A = []

#Παίρνουμε από το χρήστη έναν ακέραιο αριθμό για να τον
#χρησιμοποιήσουμε στην παράσταση
x = int(input("Dwse enan akeraio arithmo: "))

#Υπολογίζουμε την παράσταση και αποθηκεύουμε τους
#υπολογισμούς στον πίνακα
for i in range(100):
     if(x < 10):
          A.append(x*i)
     else:
          A.append(2*i)

#Τυπώνουμε τον πίνακα
```

```
print(A)
```

Εκφώνηση 4.7.14

```
#Στο τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών
#Συστημάτων δόθηκαν τα αποτελέσματα στο μάθημα "Εισαγωγή
#στους Η/Υ" για 10 φοιτητές. Να γραφεί πρόγραμμα σε γλώσσα
#Python το οποίο:
#Θα αποθηκεύει το βαθμό όλων των ατόμων σε έναν
#μονοδιάστατο πίνακα και τον αριθμό μητρώου σε άλλον πίνακα
#με αντιστοιχία θέσεων.
#Θα υπολογίζεται και θα εμφανίζεται το πλήθος των ατόμων
#κατά βαθμολογία συνοδευόμενο από τη φράση:
#·        0 έως και 3,9 : 'ΕΠΑΝΑΛΗΨΗ ΜΑΘΗΜΑΤΟΣ'
#·        4 έως και 4,9 : 'ΚΑΤΟΧΥΡΩΣΗ ΜΑΘΗΜΑΤΟΣ'
#·        άνω του 5      : 'ΠΕΡΑΣΑΝ ΤΟ ΜΑΘΗΜΑ'
#Θα εμφανίζονται οι Βαθμολογίες των 4 καλύτερων φοιτητών.
#Θα εμφανίζονται οι Αριθμοί Μητρώου των φοιτητών που έχουν
#τις 4 καλύτερες βαθμολογίες.

#Αρχικοποιούμε τους δύο πίνακες
AM = []
Vathmos =[]

#Παίρνουμε από το χρήστη το ΑΜ του και τη Βαθμολογία του
for i in range(10):
     AM.append(int(input("Dwse ton AM tou foititi: ")))
     Vathmos.append(float(input("Dwse vathmologia tou
foititi")))

#Αρχικοποιούμε τις μεταβλητές μας
count1 = 0
count2 = 0
count3 = 0
max = 0

#Υπολογίζουμε πόσες βαθμολογίες είναι μικρότερες του 3.9
#πόσες μικρότερες του 4.9 και πόσες τουλάχιστον 5
for i in range(10):
     if(Vathmos[i] <= 3.9):
          count1 = count1 + 1
     elif(Vathmos[i] <= 4.9):
          count2 = count2 +1
     else:
          count3 = count3 +1
```

```
#Τυπώνουμε τα αποτελέσματά μας καταλλήλως
print("0 ews kai 3,9 : Epanalipsi Mathimatos {}
Foitites".format(count1));
print("4 ews kai 4,9 : Katoxyrwsi Mathimatos {}
Foitites".format(count2));
print("anw tou 5 : Perasan to Mathima {}
Foitites".format(count3));

#Αρχικοποιούμε τις μεταβλητές μας
tmp = 0
tmp1 = 0

#Ταξινομούμε τους πίνακες με βάση τις βαθμολογίες σε
#φθίνουσα σειρά
for i in range(10):
     for j in range(10):
          if(Vathmos[i] > Vathmos[j]):
               tmp = Vathmos[j]
               Vathmos[j] = Vathmos[i]
               Vathmos[i] = tmp
               tmp1 = AM[j]
               AM[j] = AM[i]
               AM[i] = tmp1

#Τυπώνουμε τις 4 καλύτερες βαθμολογίες μαζί με τα ΑΜ
print("Oi kaluteres 4 vathmologies einai: ", Vathmos[0:4])
print("Oi antistoixoi Arithmoi Mitroou twn foititwn einai:
", AM[0:4])
```

4.8 Ασκήσεις με πίνακες δύο διαστάσεων

Εκφώνηση 4.8.1

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα πίνακα ακέραιων αριθμών Α, δύο διαστάσεων 3 x 3 και να
#υπολογίζει και να τυπώνει το μέσο όρο, το άθροισμα και το
#γινόμενο των στοιχείων του πίνακα Α, καθώς και τα στοιχεία
#του πίνακα κατά σειρές και στήλες.

#Αρχικοποιούμε τον πίνακα
A = []
```

```
#Παίρνουμε τα στοιχεία από το χρήστη και τα εισάγουμε στο
πίνακα
for i in range(3):
     a = []
     for j in range(3):
          x = int(input("Dwse enan akeraio arithmo: "))
          a.append(x)
     A.append(a)

#Αρχικοποιούμε τις μεταβλητές μας
sum = 0
gin = 0

#Υπολογίζουμε το άθροισμα και το γινόμενο
for i in range(3):
     for j in range(3):
          sum = sum + A[i][j]
          gin = gin * A[i][j]

#Υπολογίζουμε το μέσο όρο
MO = sum/9

#Τυπώνουμε το αποτέλεσμα
print("To athroisma einai = {} \nto ginomeno einai: {} \no
Mesos Oros einai: {}\n".format(sum,gin,MO))

#Τυπώνουμε τον πίνακα 3*3
for i in range(3):
     for j in range(3):
          print(A[i][j], "   ", end = "")
     print("\n")
```

Εκφώνηση 4.8.2

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα πίνακα ακέραιων αριθμών Α, δύο διαστάσεων 3 x 3 και να
#υπολογίζει και να τυπώνει το μέγιστο και το ελάχιστο
#στοιχείο του πίνακα και τις θέσεις τους στο πίνακα Α.

#Αρχικοποιούμε τον πίνακα
A = []

#Παίρνουμε από το χρήστη τους αριθμούς
for i in range(3):
     a = []
     for j in range(3):
```

```
            x = int(input("Dwse enan akeraio arithmo: "))
            a.append(x)
        A.append(a)


#Αρχικοποιούμε τις μεταβλητές μας
max = 0
min = A[0][0]
row_max = 0
col_max = 0
row_min = 0
col_min = 0

#υπολογίζουμε το μέγιστο και ελάχιστό καθώς και το σε ποια
#θέση βρίσκονται
for i in range(3):
    for j in range(3):
        if(max < A[i][j]):
            max = A[i][j]
            row_max = i
            col_max = j
        if(min > A[i][j]):
            min = A[i][j]
            row_min = i
            col_min = j

#Τυπώνουμε τον πίνακα σε κατάλληλη μορφή
for i in range(3):
    for j in range(3):
        print(A[i][j], "   ", end = "")
    print("\n")


#Τυπώνουμε τα αποτελέσματά μας
print("To megisto stoixeio einai to: {} kai vrisketai sth
thesi: ({},{})".format(max,row_max+1,col_max+1))
print("To elaxisto stoixeio einai to: {} kai vrisketai sth
thesi: ({},{})".format(min,row_min+1,col_min+1))
```

Εκφώνηση 4.8.3

```
#Να γραφεί πρόγραμμα σε γλώσσα Pyton το οποίο να διαβάζει
#ένα πίνακα ακέραιων αριθμών Α, δύο διαστάσεων 3 x 3 και να
#υπολογίζει και να τυπώνει
#α)το μέγιστο στοιχείο (και τη θέση του) της 3ης γραμμής
#β)το ελάχιστο στοιχείο (και τη θέση του) της 2ης στήλης
#γ)το άθροισμα των στοιχείων της 1ης γραμμής
```

```
#δ)το γινόμενο των στοιχείων της 3ης στήλης και
#ε)το άθροισμα των στοιχείων της κύριας διαγωνίου.

#Αρχικοποιούμε τον πίνακα
A = []

#Παίρνουμε από το χρήστη τους αριθμούς
for i in range(3):
     a = []
     for j in range(3):
          x = int(input("Dwse enan akeraio arithmo: "))
          a.append(x)
     A.append(a)

#Αρχικοποιούμε τη μεταβλητή μας για να βρούμε το μέγιστο
#της 3ης γραμμής
max = A[2][0]

#Υπολογίζουμε το μέγιστο και σε ποια στήλη βρίσκεται
for j in range(1,3):
     if(max < A[2][j]):
          max = A[2][j]
          indx_max = j

#Αρχικοποιούμε τη μεταβλητή μας για να βρούμε το ελάχιστο
#της 2ης στήλης
min = A[0][1]
indx_min = 0

#Υπολογίζουμε το ελάχιστο και σε ποια γραμμή βρίσκεται
for i in range(1,3):
     if(min > A[i][2]):
          min = A[i][2]
          indx_min = i

#Αρχικοποιούμε τη μεταβλητή μας για το άθροισμα
sum = 0

#Υπολογίζουμε το άθροισμα των στοιχείων της 1ης γραμμής
for k in range(3):
     sum = sum + A[0][k]

#Αρχικοποιούμε τη μεταβλητή μας για το γινόμενο
gin = 1

#Υπολογίζουμε το γινόμενο της 3ης στήλης
```

```
for l in range(3):
     gin = gin * A[l][2]

#Αρχικοποιούμε τη μεταβλητή μας για να υπολογίσουμε το
#ίχνος του πίνακα
trace = 0

#Υπολογίζουμε το ίχνος του πίνακα
for i in range(3):
     trace = trace + A[i][i]

#Τυπώνουμε κατάλληλα τον πίνακα
for i in range(3):
     for j in range(3):
          print(A[i][j], "   ", end = "")
     print("\n")

#Τυπώνουμε τα αποτελέσματά μας
print("το μέγιστο στοιχείο της 3ης γραμμής ειναι το: {} και
η θεση του ειναι η: {}".format(max,[2 + 1, indx_max + 1]))
print("το ελάχιστο στοιχείο της 2ης στήλης ειναι το: {} και
η θεση του ειναι η: {}".format(min,[indx_min + 1, 2]))
print("το άθροισμα των στοιχείων της 1ης γραμμής ειναι:
{}".format(sum))
print("το γινόμενο των στοιχείων της 3ης στήλης είναι:
{}".format(gin))
print("το ίχνος του πίνακα είναι:{}".format(trace))
```

Εκφώνηση 4.8.4

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα πίνακα ακέραιων αριθμών Α, δύο διαστάσεων 3 x 3 και
#να υπολογίζει και να τυπώνει το άθροισμα των στοιχείων της
#δευτερεύουσας διαγωνίου.

#Αρχικοποιούμε τον πίνακα
A = []

#Παίρνουμε από το χρήστη τους αριθμούς
for i in range(3):
     a = []
     for j in range(3):
          x = int(input("Dwse enan akeraio arithmo: "))
          a.append(x)
```

```
        A.append(a)

#Αρχικοποιούμε τη μεταβλητή μας για να βρούμε το άθροισμα
#της δευτερεύουσας διαγωνίου
secondary = 0

#Υπολογίζουμε το άθροισμα
for i in range(3):
        secondary = secondary + A[i][2-i]

#Τυπώνουμε κατάλληλα τον πίνακα
for i in range(3):
        for j in range(3):
                print(A[i][j], "   ", end = "")
        print("\n")

#Τυπώνουμε το αποτέλεσμα
print("To      athroisma      tis      kyrias      diagwniou
einai:{}".format(secondary))
```

Εκφώνηση 4.8.5

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#δύο πίνακες A και B δύο διαστάσεων 3 X 3 και να υπολογίζει
#το άθροισμά τους σε ένα πίνακα C.

#Αρχικοποιούμε τον πίνακα
A = []

#Παίρνουμε από το χρήστη τους αριθμούς
for i in range(3):
        a = []
        for j in range(3):
                x = int(input("Dwse enan akeraio arithmo: "))
                a.append(x)
        A.append(a)

#Αρχικοποιούμε τον πίνακα
B = []

#Παίρνουμε από το χρήστη τους αριθμούς
for i in range(3):
        b = []
        for j in range(3):
                y = int(input("Dwse enan akeraio arithmo: "))
                b.append(y)
        B.append(b)
```

```
#Αρχικοποιούμε τον πίνακα
C = []

#Υπολογίζουμε το άθροισμα των δύο πινάκων
for i in range(3):
     c = []
     for j in range(3):
          c.append(A[i][j] + B[i][j])
     C.append(c)

#τυπώνουμε κατάλληλα τον τελικό πίνακα
for i in range(3):
     for j in range(3):
          print(C[i][j], "   ", end = "")
     print("\n")
```

Εκφώνηση 4.8.6

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#δύο πίνακες a(n,m) και b(m,l). Να υπολογιστεί ο πίνακας
#c(n,l) ο οποίος θα αποτελεί το γινόμενο των a και b.
#Στην άσκηση αυτή χρησιμοποιείστε πίνακα a(2,3) και b(3,2)
#και βάλτε το γινόμενό τους στο πίνακα c(2,2).

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τις
#διαστάσεις ενός πίνακα και δίνει ως έξοδο τον πίνακα με τα
#στοιχεία που δέχεται από το χρήστη
def read_matrix(x,y):

#Αρχικοποιούμε τον πίνακα
     A = []

     #Παίρνουμε από το χρήστη τα στοιχεία του πίνακα
     for i in range(x):
          a = []
          for j in range(y):
               x = int(input("Dwse enan akeraio arithmo:
"))
               a.append(x)
          A.append(a)

     #Επιστρέφουμε τον πίνακα
     return(A)
```

```
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο δύο
#πίνακες και δίνει ως έξοδο ένα νέο πίνακα ο οποίος είναι ο
#πολλαπλασιασμός των δύο πινάκων
def multiply_matrices(k,z):

     #Αρχικοποιούμε τον πίνακα ο οποίος θα αποθηκευτεί το
#αποτέλεσμα του πολλαπλασιασμού
     c = []

     #Υπολογίζουμε το πολλαπλασιασμό των δύο πινάκων
     for i in range(n):
          a =[]
          sum = 0
          for u in range(l):
               sum = 0
               for j in range(m):
                    sum = sum + (k[i][j]*z[j][u])
               a.append(sum)
          c.append(a)

     #Επιστρέφουμε το νέο πίνακα
     return(c)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τυπώνει κατάλληλα αυτόν
def print_matrix(t):
     for i in range(len(t)):
          for j in range(len(t[i])):
               print(t[i][j], "   ", end = "")
          print("\n")

#Αρχικοποιούμε τις διαστάσεις των πινάκων
n = 2
m = 3
l = 2

#Καλούμε τη συνάρτηση για διαβάσουμε από το χρήστη τον
#πρώτο πίνακα
a = read_matrix(n,m)
print_matrix(a)

#Καλούμε τη συνάρτηση για διαβάσουμε από το χρήστη το
#δεύτερο πίνακα
b = read_matrix(m,l)
print_matrix(b)
```

```
print("\n\n")

#καλούμε τη συνάρτηση για να υπολογίσουμε τον
#πολλαπλασιασμό των δύο πινάκων
C = multiply_matrices(a,b)
print_matrix(C)
```

Εκφώνηση 4.8.7

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#δύο πίνακες a(3,3) και b(3,3) και να τους συγχωνεύει σε
#ένα πίνακα c(3,6).

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τις
#διαστάσεις ενός πίνακα και δίνει ως έξοδο τον πίνακα με τα
#στοιχεία που δέχεται από το χρήστη
def read_matrix(x,y):

     #Αρχικοποιούμε τον πίνακα
     A = []

     #Παίρνουμε από το χρήστη τα στοιχεία του πίνακα
     for i in range(x):
          a = []
          for j in range(y):
               x = int(input("Dwse enan akeraio arithmo:
"))
               a.append(x)
          A.append(a)

     #Επιστρέφουμε τον πίνακα
     return(A)

#Δημιουργουμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τυπώνει κατάλληλα αυτόν
def print_matrix(t):
     for i in range(len(t)):
          for j in range(len(t[i])):
               print(t[i][j], "   ", end = "")
          print("\n")

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο δύο
#πίνακες και δλινει ως έξοδο τη συγχώνευση αυτών
def merge_matrices(k,l):

     #Αρχικοποιούμε δύο πίνακες
```

```
        a = []
        C = []

        #Υπολογίζουμε τη συγχώνευση των πινάκων
        for i in range(n):
            a = []
            for j in range(m):
                a.append(k[i][j])
            for k in range(m):
                a.append(l[i][j])
            C.append(a)

        #Επιστρέφουμε το συγχωνευμένο πίνακα
        return(C)

#Αρχικοποιούμε τις διαστάσεις των πινάκων
n = 3
m = 3

#Καλούμε τη συνάρτηση για να διαβάσουμε τους πίνακες
a = read_matrix(n,m)
b = read_matrix(n,m)

#Καλούμε τη συνάρτηση για να υπολογίσουμε τη συγχώνευση
c = merge_matrices(a,b)

#Τυπώνουμε το αποτέλεσμα
print_matrix(c)
```

Εκφώνηση 4.8.8

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα πίνακα a(3,3), να υπολογίζει και να εμφανίζει το
#μέγιστο στοιχείο κάθε στήλης καθώς επίσης και το άθροισμά
#τους.

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τις
#διαστάσεις ενός πίνακα και δίνει ως έξοδο τον πίνακα με τα
#στοιχεία που δέχεται από το χρήστη
def read_matrix(x,y):

        #Αρχικοποιούμε τον πίνακα
        A = []

        #Παίρνουμε από το χρήστη τα στοιχεία του πίνακα
        for i in range(x):
            a = []
```

```
            for j in range(y):
                x = int(input("Dwse enan akeraio arithmo:
"))
                a.append(x)
            A.append(a)

      #Επιστρέφουμε τον πίνακα
      return(A)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το μέγιστο στοιχείο μίας στήλης
def find_max_col(c):
      max = c[0]
      for i in range(1,3):
            if(max < c[i]):
                max = c[i]
      return(max)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τον τυπώνει κατάλληλα
def print_matrix(t):
      for i in range(len(t)):
            for j in range(len(t[i])):
                print(t[i][j], "   ", end = "")
            print("\n")

#Καλούμε τη συνάρτηση για να δημιουργήσουμε τον πίνακα
B = read_matrix()

#Καλούμε τη συνάρτηση για να τυπώσουμε τον πίνακα
print_matrix(B)

#Αρχικοποιούμε μια νέα λίστα για να αποθηκεύσουμε τα
#αποτελέσματα
maximum = []

#Υπολογίζουμε το μέγιστο στοιχείο κάθε στήλης
for col in range(3):
      mat = []
      for row in range(3):
            mat.append(B[row][col])
      maximum.append(find_max_col(mat))

#Υπολογίζουμε το άθροισμα των μέγιστων στοιχείων κάθε
#στήλης
athroisma = sum(maximum)
```

```
#Τυπώνουμε τα αποτελέσματα
print("To megisto tis 1hs sthlhs einai:
{}".format(maximum[0]))
print("To megisto tis 2hs sthlhs einai:
{}".format(maximum[1]))
print("To megisto tis 3hs sthlhs einai:
{}".format(maximum[2]))
print("To athroisma tous einai: {}".format(athroisma))
```

Εκφώνηση 4.8.9

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα πίνακα a(3,3) και να βρίσκει αν ο πίνακας είναι
#αραιός. Θα θεωρήσουμε ότι ο πίνακας είναι αραιός αν
#περιέχει μηδενικά σε ποσοστό μεγαλύτερο του 30%.

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τις
#διαστάσεις ενός πίνακα και δίνει ως έξοδο τον πίνακα με τα
#στοιχεία που δέχεται από το χρήστη
def read_matrix(x,y):
     #Αρχικοποιούμε τον πίνακα
     A = []

     #Παίρνουμε από το χρήστη τα στοιχεία του πίνακα
     for i in range(x):
          a = []
          for j in range(y):
               x = int(input("Dwse enan akeraio arithmo:
"))
               a.append(x)
          A.append(a)

     #Επιστρέφουμε τον πίνακα
     return(A)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τον τυπώνει κατάλληλα
def print_matrix(t):
     for i in range(len(t)):
          for j in range(len(t[i])):
               print(t[i][j], "   ", end = "")
          print("\n")

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και υπολογίζει αν είναι αραιός
```

```
def is_sparse(A):

     #Αρχικοποιούμε τη μεταβλητή μας για να υπολογίσουμε
#πόσα μηδενικά έχει ο πίνακας
     count = 0

     #Υπολογίζουμε πόσα μηδενικά έχει ο πίνακας
     for i in range(3):
          for j in range(3):
               if(A[i][j] == 0):
                    count = count + 1
     #Αν έχει πάνω απο 30% μηδενικά στοιχεία επιστρέφουμε
#(Αλήθεια) δηλαδή ότι ο πίνακας είναι αραιός
     if((count/9) > 0.3):
          return(True)

     #αλλιώς επιστρέφουμε ότι δεν είναι αραιός
     return(False)

#Καλούμε τη συνάρτηση για να διαβάσουμε τον πίνακα
B = read_matrix()

#Τυπώνουμε τον πίνακα
print_matrix(B)

#καλούμε τη συνάρτηση για να υπολογίσουμε αν ο πίνακας
#είναι αραιός
val = is_sparse(B)

#τυπώνουμε το αποτέλεσμα
if(val == False):
     print("O pinakas den einai araios")
else:
     print("O pinakas einai araios")
```

Εκφώνηση 4.8.10

```
#Να γραφεί πρόγραμμα σε γλώσσα  Python το οποίο να διαβάζει
#ένα πίνακα ακέραιων αριθμών Α, δύο διαστάσεων 3 x 3 και να
#υπολογίζει και να τυπώνει
#α)το μέγιστο και το ελάχιστο στοιχείο του πίνακα και τις
#θέσεις τους (στήλη και γραμμή),
#β)το μέγιστο στοιχείο (και τη θέση του) της 1ης γραμμής
#γ)το ελάχιστο στοιχείο (και τη θέση του) της 3ης στήλης
#δ)το άθροισμα των στοιχείων της 1ης γραμμής
#ε)το γινόμενο των στοιχείων της 3ης στήλης και
```

```
# το άθροισμα των στοιχείων της κυρίας διαγωνίου

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τις
#διαστάσεις ενός πίνακα και δίνει ως έξοδο τον πίνακα με τα
#στοιχεία που δέχεται από το χρήστη
def read_matrix(x,y):

     #Αρχικοποιούμε τον πίνακα
     A = []

     #Παίρνουμε από το χρήστη τα στοιχεία του πίνακα
     for i in range(x):
          a = []
          for j in range(y):
               x = int(input("Dwse enan akeraio arithmo:
"))
               a.append(x)
          A.append(a)

     #Επιστρέφουμε τον πίνακα
     return(A)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τον τυπώνει κατάλληλα
def print_matrix(t):
     for i in range(len(t)):
          for j in range(len(t[i])):
               print(t[i][j], "   ", end = "")
          print("\n")

#Erwtima a
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το μέγιστο και ελάχιστο στοιχείο
#όπως και τις θέσεις τους στον πίνακα
def find_min_max_ofMatrix(A):

     #Αρχικοποιούμε τις μεταβλητές μας
     max = A[0][0]
     indx_max = [1,1]
     min = A[0][0]
     indx_min = [1,1]

     #Υπολογίζουμε το μέγιστο και το ελάχιστο καθώς και τις
#θέσεις τους
     for row in range(1,3):
          for col in range(1,3):
```

```
                if(max < A[row][col]):
                    max = A[row][col]
                    indx_max = [row+1,col+1]
                if(min > A[row][col]):
                    min = A[row][col]
                    indx_min = [row+1,col+1]

    #Επιστρέφει μία λίστα απο τα στοιχεία που υπολογίσαμε
    return([max,indx_max,min,indx_min])

#Erwtima b
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το μέγιστο στοιχείο και τη θέση
#του στον πίνακα της 1ης γραμμής
def find_max_of_row(A):

    #Αρχικοποιούμε τις μεταβλητές μας
    max = A[0][0]
    indx_max = [1,1]

    #Υπολογίζουμε το μέγιστο στοιχείο και τη θέση του
    for col in range(3):
        if(max < A[0][col]):
            max = A[0][col]
            indx_max = [1, col+1]

    #Επιστρέφει μία λίστα με τα στοχεία που υπολογίσαμε
    return([max, indx_max])

#Erwtima c
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το ελάχιστο στοιχείο (και τη
#θέση του) της 3ης στήλης
def find_min_of_col(A):

    #Αρχικοποιούμε τις μεταβλητές μας
    min = A[0][2]
    indx_min = [1,3]

    #υπολογίζουμε το ελάχιστο στοιχείο καθώς και τη θέση
#του
    for row in range(3):
        if(min > A[row][2]):
            min = A[row][2]
            indx_min = [row+1, 3]
```

```
        #Επιστρέφει μία λίστα με τα στοιχεία που υπολογίσαμε
        return([min, indx_min])

#Erwtima d
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το άθροισμα των στοιχείων της
#1ης γραμμής
def sum_of_row(A):

        #Αρχικοποιούμε τις μεταβλητές μας
        sum = 0

        #υπολογίζουμε το άθροισμα
        for col in range(3):
                sum = sum + A[0][col]

        #Επιστρέφει το άθροισμα
        return(sum)

#Erwtima e
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το γινόμενο των στοιχείων της
#3ης στήλης
def mult_of_col(A):

        #Αρχικοποιούμε τις μεταβλητές μας
        mult = 1

        #Υπολογίζουμε το γινόμενο
        for row in range(3):
                mult = mult * A[row][2]

        #Επιστρέφει το γινόμενο
        return(mult)

#Erwtima st
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το ίχνος του πίνακα
def trace_of_mat(A):

        #Αρχικοποιούμε τη μεταβλητή μας
        sum = 0

        #Υπολογίζουμε το άθροισμα
        for i in range(3):
                sum = sum + A[i][i]
```

```
    #Επιστρέφει το άθροισμα
    return(sum)

#Καλούμε τη συνάρτηση για να διαβάσουμε τον πίνακα
B = read_matrix()

#Τυπώνουμε τον πίνακα
print_matrix(B)

#Καλούμε τις συναρτίσεις για κάθε ερώτημα
erwtima_a = find_min_max_ofMatrix(B)
erwtima_b = find_max_of_row(B)
erwtima_c = find_min_of_col(B)
erwtima_d = sum_of_row(B)
erwtima_e = mult_of_col(B)
erwtima_st = trace_of_mat(B)

#τυπώνουμε τα αποτελέσματα
print("Erwtima a \nTo megisto tou pinaka einai :{}\nkai i
thesi tou einai i: {}\nTo elaxisto tou pinaka einai: {}\n
kai i thesi tou einai:
{}".format(erwtima_a[0],erwtima_a[1],
erwtima_a[2],erwtima_a[3]))
print("Erwtima b \nTo megisto ths 1hs grammhs einai
:{}\nkai i thesi tou einai i:
{}".format(erwtima_b[0],erwtima_b[1]))
print("Erwtima c \nTo elaxisto ths 3hs stilis einai
:{}\nkai i thesi tou einai i:
{}".format(erwtima_c[0],erwtima_c[1]))
print("Erwtima d \nTo athroisma ths 1hs grammis einai:
{}".format(erwtima_d))
print("Erwtima e \nTo ginomeno ths 3hs sthlhs einai:
{}".format(erwtima_e))
print("Erwtima st \nTo ixnos tou pinaka einai:
{}".format(erwtima_st))
```

Εκφώνηση 4.8.11

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα πίνακα ακέραιων αριθμών Α, δύο διαστάσεων 3 x 3 και
#να υπολογίζει και να τυπώνει το μέσο όρο,
#το άθροισμα και
#το γινόμενο των στοιχείων του πίνακα Α καθώς και
```

```
#πόσα στοιχεία του πίνακα είναι μικρότερα από το μέσο όρο

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τις
#διαστάσεις ενός πίνακα και δίνει ως έξοδο τον πίνακα με τα
#στοιχεία που δέχεται από το χρήστη
def read_matrix(x,y):

     #Αρχικοποιούμε τον πίνακα
     A = []

     #Παίρνουμε από το χρήστη τα στοιχεία του πίνακα
     for i in range(x):
          a = []
          for j in range(y):
               x = int(input("Dwse enan akeraio arithmo:
"))
               a.append(x)
          A.append(a)

     #Επιστρέφουμε τον πίνακα
     return(A)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τον τυπώνει κατάλληλα
def print_matrix(t):
     for i in range(len(t)):
          for j in range(len(t[i])):
               print(t[i][j], "   ", end = "")
          print("\n")

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο το μέσο όρο το άθροισμα και το
#γινόμενο των στοιχείων
def mo_athr_gin(A):

     #Αρχικοποιούμε τις μεταβλητές μας
     sum = 0
     mult = 1

     #υπολογίζουμε το άθροισμα και το γινόμενο
     for i in range(3):
          for j in range(3):
               sum = sum + A[i][j]
               mult = mult * A[i][j]

     #Υπολογίζουμε το μέσο όρο
```

```
        mo = sum/9

        #Επιστρέφουμε μία λίστα με τα αποτελέσματα
        return([mo,sum,mult])

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο πόσα στοιχεία είναι μικρότερα
#του μέσου όρου
def stoixeia_mikrotera_mo(A):

        #Αρχικοποιούμε το μετρητή μας
        count = 0

        #Υπολογίζουμε πόσα στοιχεία είναι μικρότερα του μέσου
#όρου
        for i in range(3):
             for j in range(3):
                  if(A[i][j]< apotelesmata[0]):
                       count = count + 1
        #Επιστρέφουμε το αποτέλεσμα
        return(count)

#Καλούμε τη συνάρτηση για να διαβάσουμε τα στοιχεία του
#πίνακα
B = read_matrix()


#Καλούμε τη συνάρτηση για να τυπώσουμε τον πίνακα
print_matrix(B)

#Καλούμε τις συναρτήσεις και αποθηκεύουμε τα αποτελέσματά
#μας
apotelesmata = mo_athr_gin(B)
stoix = stoixeia_mikrotera_mo(B)

#Τυπώνουμε τα αποτελέσματα
print ("O mo twn stoixeiwn tou pinaka einai: {} \nTo
athroisma twn stoixeiwn einai:{} \n to ginomeno einai:
{}".format(apotelesmata[0],apotelesmata[1],apotelesmata[2])
)
print("{}    stoixeia    einai    mikrotera    tou    mesou
orou".format(stoix))
```

Εκφώνηση 4.8.12

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να
#δημιουργεί και να τυπώνει ένα πίνακα δύο διαστάσεων 10 x
#10 που να περιέχει την προπαίδεια από το 1 έως το 10.

#Δημιουργουμε συνάρτηση η οποία δημιουργεί και επιστρέφει
#έναν πίνακα ο οποίος περιέχει την προπαίδεια
def create_matrix():

     #Αρχικοποιούμε τον πίνακα
     A = []

     #Υπολογίζουμετην προπαίδει για κάθε αριθμό απο το 1-10
     for i in range(1,11):
          a = []
          for j in range(1,11):
               a.append(i*j)
          A.append(a)

     #Επιστρλεφουμε τον πίνακα
     return(A)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τον τυπώνει κατάλληλα
def print_matrix(t):
     for i in range(len(t)):
          for j in range(len(t[i])):
               print(t[i][j], "\t", end = "")
          print("\n")

#Καλούμε τη συνάρτηση για να δημιουργηθεί ο πίνακας με τη
#προπαίδεια
B = create_matrix()

#Τυπώνουμε το αποτέλεσμα
print_matrix(B)
```

Εκφώνηση 4.8.13

```
#Να γραφεί πρόγραμμα σε γλώσσα Python το οποίο να διαβάζει
#ένα πίνακα ακέραιων αριθμών Α, δύο διαστάσεων 3 x 3 και
#να υπολογίζει και να συγκρίνει το άθροισμα της 2ης γραμμής
#με το άθροισμα της 2ης στήλης. Επίσης να υπολογίζει και να
#συγκρίνει το άθροισμα της κυρίας διαγωνίου με το άθροισμα
#των στοιχείων της δευτερεύουσας διαγωνίου.
```

```
#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τις
#διαστάσεις ενός πίνακα και δίνει ως έξοδο τον πίνακα με τα
#στοιχεία που δέχεται από το χρήστη
def read_matrix(x,y):

     #Αρχικοποιούμε τον πίνακα
     A = []

     #Παίρνουμε από το χρήστη τα στοιχεία του πίνακα
     for i in range(x):
          a = []
          for j in range(y):
               x = int(input("Dwse enan akeraio arithmo:
"))
               a.append(x)
          A.append(a)

     #Επιστρέφουμε τον πίνακα
     return(A)

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και τον τυπώνει κατάλληλα
def print_matrix(t):
     for i in range(len(t)):
          for j in range(len(t[i])):
               print(t[i][j], "  ", end = "")
          print("\n")

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο έναν
#πίνακα και δίνει ως έξοδο ο άθροισμα της 2ης γραμμής,
#το άθροισμα της 2ης στήλης, το άθροισμα της κυρίας
#διαγωνίου και το άθροισμα των στοιχείων της δευτερεύουσας
#διαγωνίου
def compute_sums(A):

     #Αρχικοποιούμε τις μεταβλητές μας
     sum_row = 0
     sum_col = 0
     trace = 0
     secondary = 0

     #Υπολογίζουμε τα αθροίσματα
     for row in range(3):
          secondary = secondary + A[row][2-row]
          for col in range(3):
               if(row == 1):
```

```
                    sum_row = sum_row + A[row][col]
               if(col == 1):
                    sum_col = sum_col + A[row][col]
               if(col == row):
                    trace = trace + A[row][col]

     #Επιστρέφουμε μία λίστα από τα αποτελέσματα
     return([sum_row,sum_col,trace,secondary])

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τα
#αθροίσματα της γραμμής και της στήλης και δίνει ως έξοδο
#τη σύγκριση αυτών
def compare_sums(x,y):
     if(x > y):
          return("To athroisma tis grammis einai
megalytero")
     elif(x < y):
          return("To athroisma tis stilis einai
megalytero")
     else:
          return("Ta athroismata einai isa")

#Δημιουργούμε συνάρτηση η οποία δέχεται ως είσοδο τα
#αθροίσματα της κύριας και δευτερεύουσας διαγωνίου και
#δίνει ως έξοδο τη σύγκριση αυτών
def compare_sums_of_diagonals(x,y):
     if(x > y):
          return("To athroisma tis kyrias diagwniou einai
megalytero")
     elif(x < y):
          return("To athroisma tis deuterevousas diagwniou
einai megalytero")
     else:
          return("Ta athroismata einai isa")

#Καλούμε τη συνάρτηση για να διαβάσουμε τον πίνακα
B = read_matrix()

#Τυπώνουμε τον πίνακα
print_matrix(B)

#Καλούμε τις συναρτήσεις για να υπολογίσουμε τα ζητούμενα
apotelesmata = compute_sums(B)
sugkrisi = compare_sums(apotelesmata[0],apotelesmata[1])
sugkrisi2 =
compare_sums_of_diagonals(apotelesmata[2],apotelesmata[3])
```

```
#Τυπώνουμε τα αποτελέσματα
print("To athroisma tis 2hs grammhs
einai:{}".format(apotelesmata[0]))
print("To athroisma tis 2hs sthlhs
einai:{}".format(apotelesmata[1]))
print(sugkrisi)
print("To ixnos tou pinaka
einai:{}".format(apotelesmata[2]))
print("To athroisma tis deuterevousas diagwniou
einai:{}".format(apotelesmata[3]))
print(sugkrisi2)
```

5. Συμπεράσματα

Η Python είναι μια γλώσσα προγραμματισμού πολλαπλών υποδειγμάτων δηλαδή υποστηρίζει διαδικασιακό, αντικειμενοστραφή και συναρτησιακό προγραμματισμό. Επιπλέον, χρησιμοποιεί δυναμική πληκτρολόγηση, ένα συνδυασμό μέτρησης αναφορών και συλλογής σκουπιδιών που ανιχνεύει κύκλους για τη διαχείριση της μνήμης. Η Python είναι απλούστερη, με λιγότερο σύνθετο συντακτικό και γραμματικό, δίνοντας στους προγραμματιστές μία πληθώρα επιλογών στο ποια μεθοδολογία θα χρησιμοποιήσουν. Ταυτόχρονα υπάρχει μία μεγάλη κοινότητα προγραμματιστών οι οποίοι προσφέρουν στη γλώσσα αφιλοκερδώς. Η Python είναι μία γλώσσα γενικού σκοπού και προσφέρεται για εφαρμογές κινητών και διαδικτυακές εφαρμογές. Επιπρόσθετα, μπορεί να χρησιμοποιηθεί σε εφαρμογές τεχνητής νοημοσύνης και υπολογιστικού υπολογισμού. Τέλος καταλήγουμε με τους τέσσερεις αφορισμούς που αποτελούν τη φιλοσοφία της γλώσσας (https://en.wikipedia.org/wiki/Python_(programming_language)):

- Το όμορφο είναι καλύτερο από το άσχημο
- Το σαφές είναι καλύτερο από το υποτιθέμενο
- Το απλό είναι καλύτερο από το σύνθετο
- Το σύνθετο είναι καλύτερο από το πολύπλοκο
- Η αναγνωσιμότητα μετράει

Βιβλιογραφία

Barron, D. W., Buxton, J. N., Hartley, D. F., Nixon, E., & Strachey, C. (1963). The main features of CPL. The Computer Journal, 6(2), 134-143.

Fuegi, J., & Francis, J. (2003). Lovelace & Babbage and the creation of the 1843 'notes'. IEEE Annals of the History of Computing, 25(4), 16-26.

Hodges, A. (2012). Alan Turing: The Enigma: The Enigma. Random House.

Knuth, D. E., & Pardo, L. T. (1980). The early development of programming languages. In A history of computing in the twentieth century (pp. 197-273). Academic Press.

Naur, P., Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., & Perlis, A. J. (1976). Revised report on the algorithmic language Algol 60. Association for Computing Machinery.

Richards, M., & Whitby-Strevens, C. (1981). BCPL: the language and its compiler. Cambridge University Press.

Ritchie, D. M. (1993). The development of the C language. ACM Sigplan Notices, 28(3), 201-208.

https://en.wikipedia.org/wiki/Ada_Lovelace

https://en.wikipedia.org/wiki/Python_(programming_language)

https://www.educba.com/c-vs-python/