ΕΝΟΤΗΤΕΣ:

Εισαγωγή

Η Python είναι μια δημοφιλής γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία εφαρμογών, την ανάλυση δεδομένων, την ανάπτυξη ιστοσελίδων και πολλά άλλα. Ο λόγος που είναι τόσο δημοφιλής είναι επειδή είναι εύκολη στη χρήση, ακόμα και για αρχάριους, και διαθέτει πολλές βιβλιοθήκες που επιτρέπουν στους προγραμματιστές να κάνουν πολύπλοκα πράγματα με απλές εντολές. Η Python εκτελείται γραμμή προς γραμμή, πράγμα που σημαίνει ότι διαβάζει τον κώδικα που γράφεις και τον εκτελεί αμέσως. Αυτό είναι πολύ χρήσιμο όταν θέλεις να δοκιμάσεις γρήγορα ιδέες ή να κάνεις μικρά προγράμματα, χωρίς να χρειάζεται να περιμένεις την εκτέλεση ολόκληρου του κώδικα.

Μία από τις πιο δημοφιλείς εφαρμογές για να γράψεις κώδικα σε Python είναι μέσω του Visual Studio Code (VSCode). Το VSCode είναι ένα ισχυρό πρόγραμμα για συγγραφή κώδικα. Για να ξεκινήσεις να γράφεις και να εκτελείς προγράμματα σε Python, θα χρειαστείς τα εξής:

- **Visual Studio Code (VSCode)**: Για να γράφεις τον κώδικα σου εύκολα, μπορείς να κατεβάσεις το VSCode. Μετά την εγκατάσταση, θα χρειαστεί να εγκαταστήσεις το Python Extension που βοηθάει το VSCode να διαχειρίζεται καλύτερα τον κώδικα Python.
- **Python Interpreter**: Ο διερμηνευτής της Python, που μπορείς να κατεβάσεις από το python.org. Είναι ένα πρόγραμμα, ο βασικός μηχανισμός που «μεταφράζει» τον κώδικα Python σε οδηγίες που καταλαβαίνει ο υπολογιστής. Είναι απαραίτητος για να τρέξεις οποιοδήποτε πρόγραμμα που γράφεις στην Python. Χωρίς τον διερμηνευτή, ο υπολογιστής δεν θα μπορούσε να καταλάβει τις εντολές σου.

Ενώ αυτά τα εργαλεία είναι πολύ χρήσιμα, δεν είναι απαραίτητο να τα εγκαταστήσεις στη συσκευή σου για να ξεκινήσεις να γράφεις κώδικα. Υπάρχει μια εναλλακτική λύση που σου επιτρέπει να γράφεις και να εκτελείς κώδικα Python απευθείας από τον browser σου, χωρίς καμία εγκατάσταση. Αυτή η λύση είναι το **replit**, ένα διαδικτυακό IDE που υποστηρίζει πολλές γλώσσες προγραμματισμού. Το repl.it είναι πολύ εύκολο στη χρήση και σου επιτρέπει να εργάζεσαι με Python (και άλλες γλώσσες) χωρίς να χρειάζεται να εγκαταστήσεις επιπλέον λογισμικό.

Ένα IDE (Ολοκληρωμένο Προγραμματιστικό Περιβάλλον) είναι ένα λογισμικό που συνδυάζει όλα τα εργαλεία που χρειάζεσαι για να γράψεις και να διαχειριστείς τον κώδικα σου. Συνήθως περιλαμβάνει:

- **Επεξεργαστή Κώδικα:** Ένα εργαλείο για να γράφεις και να επεξεργάζεσαι τον κώδικα σου.

- **Μεταφραστή:** Ελέγχει και μετατρέπει τον κώδικα σε μορφή που καταλαβαίνει ο υπολογιστής.
- **Βοηθητικά Εργαλεία:** Όπως εργαλεία για τη διαχείριση αρχείων και την εύρεση σφαλμάτων.

Με ένα IDE, έχεις όλα όσα χρειάζεσαι σε ένα μέρος, κάνοντάς την ανάπτυξη προγραμμάτων πιο εύκολη και γρήγορη.

Τύποι δεδομένων και τελεστές

Οι τύποι δεδομένων είναι:

- Αριθμητικός:
  - Ακέραιος (Integer) όπως οι αριθμοί 10, 34 και 86
  - Δεκαδικός, δηλαδή αριθμός με υποδιαστολή (Float) όπως οι αριθμοί 3.14, 79.45 και 1.03
  - Μιγαδικός (Complex) που δεν θα μας απασχολήσει στη συνέχεια.
- Λογικός
  - Τιμή True (Αληθής)
  - Τιμή False (Ψευδής)
- Συμβολοσειρές ή Αλφαριθμητικά (Strings)
  - Είναι μια ακολουθία από χαρακτήρες. Μια συμβολοσειρά μπορεί να αποτελείται από περισσότερες από μία λέξεις, σχηματίζοντας ένα κείμενο. Είναι σαν μια αλυσίδα, όπου κάθε κρίκος είναι ένας χαρακτήρας, δηλαδή ένα γράμμα, ένας αριθμός, ένα σύμβολο, ή ακόμα και ένα κενό διάστημα. Για παράδειγμα: ‘1234’, ‘Hello World!’ και ‘231!~9-=’

Οι τελεστές είναι:

- Αριθμητικοί (που χρησιμοποιούμε για να κάνουμε μαθηματικές πράξεις):
  - Πρόσθεση: +
  - Αφαίρεση: -
  - Πολλαπλασιασμός: *
  - Διαίρεση: /
  - Ύψωση σε δύναμη: **
  - Υπόλοιπο ακέραιας διαίρεσης (mod): %

  Σε κάθε έκφραση στην οποία υπάρχουν αριθμητικοί τελεστές, τηρείται η παρακάτω ιεραρχία πράξεων:

  1. Ύψωση σε δύναμη.
  2. Πολλαπλασιασμός/Διαίρεση/Υπόλοιπο ακέραιας διαίρεσης
  3. Πρόσθεση/Αφαίρεση

  Σημαντικό: ΑΝ θέλουμε να αλλάξουμε την ιεραρχία των πράξεων, τότε χρησιμοποιούμε τις παρενθέσεις (). Για παράδειγμα, στην έκφραση 6/(2+3) θα εκτελεστεί πρώτα η πρόσθεση μέσα στις παρενθέσεις και

μετά θα γίνει η διαίρεση. Αντίθετα, στην έκφραση 6/2+3 εκτελείται πρώτα η διαίρεση και μετά η πρόσθεση.

- Σχεσιακοί ή Συγκριτικοί (που χρησιμοποιούνται για τη σύγκριση δύο τιμών ή μεταβλητών, με το αποτέλεσμα της σύγκρισης να είναι True ή False):
  - Ίσο με: ==
  - Διάφορο από: !=
  - Μικρότερο από: <
  - Μικρότερο ή ίσο από: <=
  - Μεγαλύτερο από: >
  - Μεγαλύτερο ή ίσο από: >=
- Τελεστές Λογικών Πράξεων (που χρησιμοποιούνται στις λογικές πράξεις και εκφράσεις):
  - Πράξη άρνησης: NOT
  - Πράξη σύζευξης: AND
  - Πράξη διάζευξης: OR

  Το αποτέλεσμα μίας λογικής πράξης είναι True (Αληθής) ή False (Ψευδής) σύμφωνα με τον παρακάτω πίνακα:

A	B	A AND B	A OR B	NOT A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

  Η προτεραιότητα των λογικών τελεστών είναι:

  1. NOT
  2. AND
  3. OR

Αριθμητικές και λογικές πράξεις

Ακολουθούν παρακάτω παραδείγματα αριθμητικών και λογικών εκφράσεων:

Αριθμητική/Λογική έκφραση	Αποτέλεσμα	Αιτιολόγηση
5+3*(8-2)	23	Προτεραιότητα δίνουμε πρώτα στην παρένθεση και μετά στον πολλαπλασιασμό
2**2+4/2+3	9	Προτεραιότητα δίνουμε πρώτα στη δύναμη και μετά στη διαίρεση

64/3	21	Η διαίρεση ακέραιων αριθμών, επιστρέφει το ακέραιο πηλίκο
64%3	1	Επιστρέφει το υπόλοιπο της ακέραιας διαίρεσης
64.0/3	21.333	Η διαίρεση δεκαδικών αριθμών (με κινητή υποδιαστολή), επιστρέφει το πηλίκο ως δεκαδικό αριθμό
25==25	TRUE	Η συνθήκη είναι αληθής γιατί ισχύει η ισότητα.
35==15	FALSE	Η συνθήκη είναι ψευδής γιατί δεν ισχύει η ισότητα.
35!=15	TRUE	Η συνθήκη είναι αληθής γιατί ισχύει η διαφορετικότητα των αριθμών.
25>=26	FALSE	Η συνθήκη είναι ψευδής γιατί ο αριθμός 25 είναι μικρότερος του αριθμού 26
(24>25) AND (23==23)	FALSE	Η συνθήκη (23==23) είναι αληθής αλλά η συνθήκη (24>25) είναι ψευδής. Επειδή τα αποτελέσματα των συνθηκών διαφέρουν και ο τελεστής λογικών πράξεων είναι τελεστής σύζευξης, τότε η συνολική συνθήκη είναι ψευδής
(24>25) OR (23==23)	TRUE	Η συνθήκη (23==23) είναι αληθής αλλά η συνθήκη (24>25) είναι ψευδής. Τα αποτελέσματα των συνθηκών αν και διαφέρουν, ο τελεστής λογικών πράξεων είναι τελεστής διάζευξης και ως αποτέλεσμα, η συνολική συνθήκη είναι αληθής
NOT(25<=28)	FALSE	Η συνθήκη (25<=28) είναι αληθής. Ωστόσο επειδή ο τελεστής λογικών πράξεων, είναι τελεστής άρνησης, η συνθήκη γίνεται τελικά ψευδής
NOT(23==26)	TRUE	Η συνθήκη (23==26) είναι ψευδής. Ωστόσο επειδή ο

		τελεστής λογικών πράξεων, είναι τελεστής άρνησης, η συνθήκη γίνεται τελικά αληθής

Μεταβλητές

Στην Python, δεν χρειάζεται να δηλώσεις από την αρχή τι είδους τιμές (ακέραιοι αριθμοί, δεκαδικοί αριθμοί και συμβολοσειρές) θα αποθηκεύσει μια μεταβλητή. Μπορείς απλά να δώσεις μια τιμή σε μια μεταβλητή και η Python καταλαβαίνει από μόνη της τι τύπος είναι αυτή η τιμή. Επίσης, μπορείς να αλλάζεις τι είδους τιμή αποθηκεύεται σε μια μεταβλητή όποτε θέλεις. Για παράδειγμα, μπορείς να βάλεις έναν αριθμό σε μια μεταβλητή, μετά να τον αλλάξεις σε δεκαδικό αριθμό και αργότερα σε κείμενο. Αυτό κάνει την Python πολύ εύκολη και ευέλικτη για να γράφεις προγράμματα.

Για να χρησιμοποιήσουμε μια μεταβλητή στην Python, πρέπει πρώτα να της δώσουμε ένα όνομα και μετά να της εκχωρήσουμε μια τιμή. Υπάρχουν μερικοί κανόνες για την επιλογή ονόματος για τη μεταβλητή. Για παράδειγμα, το όνομα δεν μπορεί να ξεκινά με αριθμό και δεν πρέπει να είναι το ίδιο με κάποιο όνομα που χρησιμοποιείται ήδη από τη γλώσσα (όπως μια εντολή ή συνάρτηση). Συνήθως, το όνομα της μεταβλητής σχετίζεται με αυτό που αποθηκεύει και γράφεται με λατινικούς χαρακτήρες. Μπορείς να χρησιμοποιήσεις και αριθμούς, όπως "onoma", "ilikia", ή "mesos_oros". Για να δώσεις μια τιμή σε μια μεταβλητή, χρησιμοποιείς το σύμβολο "=", που λέγεται τελεστής εκχώρησης. Πρώτα γράφεις το όνομα της μεταβλητής, μετά το "=", και τέλος την τιμή που θες να αποθηκεύσεις. Για παράδειγμα onoma = ‘Ioannis’, ilikia = 18 και mesos_oros = 19.5

Όταν χρησιμοποιούμε το σύμβολο " = " στην Python, δεν σημαίνει ότι τα δύο μέρη είναι ίσα, όπως στα μαθηματικά. Στην πραγματικότητα, το " = " στην Python σημαίνει ότι δίνουμε μια τιμή σε μια μεταβλητή. Για παράδειγμα, όταν γράφουμε ilikia = 18, λέμε στην Python να βάλει την τιμή 18 στη μεταβλητή ilikia. Αυτό σημαίνει ότι η Python αποθηκεύει την τιμή 18 σε μια θέση μνήμης του υπολογιστή και η μεταβλητή ilikia δείχνει σε αυτήν τη θέση. Αν μετά γράψουμε ilikia = 20, τότε η Python αντικαθιστά την παλιά τιμή με τη νέα τιμή στη μνήμη. Η παλιά τιμή (18) διαγράφεται.

Λίστες

Μία λίστα είναι ένας τύπος δεδομένων που μας επιτρέπει να αποθηκεύσουμε πολλά αντικείμενα μαζί, όπως αριθμούς, λέξεις ή ακόμα και άλλες λίστες. Σκεφτείτε τη λίστα σαν ένα κουτί που περιέχει πολλά αντικείμενα και το καθένα έχει μια θέση μέσα στο κουτί. Οι λίστες είναι πολύ χρήσιμες όταν θέλουμε να αποθηκεύσουμε πολλές τιμές σε μία μεταβλητή και να μπορούμε να τις διαχειριστούμε εύκολα. Οι λίστες δημιουργούνται με αγκύλες ([]) και τα στοιχεία τους χωρίζονται με κόμματα. Για παράδειγμα:

```
# Λίστα με αριθμούς
arithmoi = [1, 2, 3, 4, 5]
# Λίστα με λέξεις
onomata = ["Ioannis", "Maria", "Kostas"]
# Λίστα με διαφορετικούς τύπους δεδομένων
mix = [12, "Markos", 3.14, onomata]

# Εκτυπώνουμε τα περιεχόμενα των λιστών
print(arithmoi)
print(onomata)
print(mix)
```

Το αποτέλεσμα που φορτώνει στο terminal:

```
[1, 2, 3, 4, 5]
['Ioannis', 'Maria', 'Kostas']
[12, 'Markos', 3.14, ['Ioannis', 'Maria', 'Kostas']]
```

Για να προσθέσουμε νέα στοιχεία σε μία λίστα, χρησιμοποιούμε την εντολή append(). Για παράδειγμα:

```
# Λίστα με αριθμούς
arithmoi = [1, 2, 3, 4, 5]
arithmoi.append(6)  # Προσθέτει το 6 στο τέλος της λίστας
print(arithmoi)
```

Το αποτέλεσμα που φορτώνει στο terminal:

```
[1, 2, 3, 4, 5, 6]
```

Για να διαγράψουμε ένα στοιχείο από μία λίστα, χρησιμοποιούμε την εντολή remove(). Για παράδειγμα:

```
# Λίστα με λέξεις
onomata = ["Ioannis", "Maria", "Kostas"]
onomata.remove("Maria")  # Αφαιρεί το στοιχείο "Maria"
print(onomata)
```

Το αποτέλεσμα που φορτώνει στο terminal:

```
['Ioannis', 'Kostas']
```

Για να αλλάξουμε ένα στοιχείο από μία λίστα, χρησιμοποιούμε το index του και απευθείας εκχώρηση νέας τιμής. Για παράδειγμα:

```
# Λίστα με αριθμούς
arithmoi = [1, 2, 3, 4, 5]
arithmoi[2] = 10 # Αλλάζει το 3ο στοιχείο (index 2) σε 10
print(arithmoi)
```

Το αποτέλεσμα που φορτώνει στο terminal:

```
[1, 2, 10, 4, 5]
```

Για να πάρουμε ένα συγκεκριμένο στοιχείο της λίστας, χρησιμοποιούμε τον αριθμό της θέσης του (index). Για παράδειγμα:

```
# Λίστα με λέξεις
onomata = ["Ioannis", "Maria", "Kostas"]
print(onomata[1])  # Επιστρέφει το 2ο στοιχείο (index 1): "Maria"
```

Το αποτέλεσμα που φορτώνει στο terminal:

Maria

Για να αφαιρέσουμε στοιχείο με βάση το index του, χρησιμοποιούμε την εντολή pop(). Για παράδειγμα:

```
# Λίστα με αριθμούς
arithmoi = [1, 2, 3, 4, 5]
arithmoi.pop(3)  # Αφαιρεί το στοιχείο στην 4η θέση (index 3)
print(arithmoi)
```

Το αποτέλεσμα που φορτώνει στο terminal:

[1, 2, 3, 5]

Βασικές (ενσωματωμένες) συναρτήσεις

Υπάρχουν κάποιες βασικές συναρτήσεις που μας βοηθούν να αλλάξουμε το είδος των δεδομένων που έχουμε ή να κάνουμε κάποιους υπολογισμούς:

- **float()**: Παίρνει έναν αριθμό και το μετατρέπει σε δεκαδικό αριθμό, δηλαδή σε αριθμό με κόμμα. Για παράδειγμα, αν γράψουμε float(5), θα μας δώσει 5.0.
- **int()**: Παίρνει οποιονδήποτε αριθμό (ακόμη και έναν δεκαδικό) και τον μετατρέπει σε ακέραιο, δηλαδή σε αριθμό χωρίς κόμμα. Αν γράψουμε int(5.99), θα μας δώσει 5. Προσέχει μόνο το κομμάτι πριν το κόμμα.
- **str()**: Παίρνει οποιονδήποτε αριθμό, λέξη, ή κάτι άλλο, και το μετατρέπει σε κείμενο. Για παράδειγμα, αν γράψουμε str(123), θα μας δώσει το "123" σαν κείμενο.
- **abs()**: Δίνει την απόλυτη τιμή ενός αριθμού, δηλαδή πόσο μακριά είναι από το μηδέν, χωρίς να κοιτάζει το πρόσημο. Για παράδειγμα, abs(-10) θα μας δώσει 10.
- **pow(a, b)**: Δίνει τη δύναμη ενός αριθμού. Αν γράψουμε pow(2, 3), θα μας δώσει 8, γιατί σημαίνει 2 υψωμένο στη δύναμη του 3 (2x2x2).
- **divmod(x, y)**: Δίνει το αποτέλεσμα και το υπόλοιπο όταν διαιρούμε δύο αριθμούς. Για παράδειγμα, αν γράψουμε divmod(10, 3), θα μας δώσει (3, 1), γιατί το 10 διαιρούμενο με το 3 δίνει 3 (ακέραιο αποτέλεσμα) και μένει υπόλοιπο 1.

Βασικές εντολές

- **Εκτύπωση μηνύματος**

  Όταν θέλουμε να εκτυπώσουμε μία συμβολοσειρά ή κείμενο με αλφαριθμητικούς χαρακτήρες (strings) στο terminal, χρησιμοποιούμε την εντολή print(). Η εντολή print(), εκτυπώνει το περιεχόμενο που βρίσκεται εντός των παρενθέσεων. Ωστόσο για να λειτουργεί η εντολή σωστά, χωρίς κάποιο πρόβλημα, πρέπει να βάλουμε στην αρχή και στο τέλος του περιεχομένού μας τα εισαγωγικά (είτε μονά είτε διπλά).
  Έστω ότι θέλουμε να εκτυπώσουμε το μήνυμα: Hello World! Για να εκτυπωθεί το συγκεκριμένο μήνυμα θα πρέπει να γράψουμε πρώτα την εντολή print() και μέσα στις παρενθέσεις το περιεχόμενό της με εισαγωγικά στην αρχή και στο τέλος. Δηλαδή η εντολή για να δουλέψει σωστά, πρέπει να γραφτεί με αυτό τον τρόπο: print(“Hello World!”)

  Επιπλέον, η εντολή print() δεν εκτυπώνει μόνο συμβολοσειρές, αλλά μπορεί να χρησιμοποιηθεί για την εκτύπωση και αριθμών, μεταβλητών, ή άλλων τύπων δεδομένων. Για παράδειγμα:

  Επιπλέον, η εντολή print() δεν εκτυπώνει μόνο συμβολοσειρές, αλλά μπορεί να χρησιμοποιηθεί για την εκτύπωση και αριθμών, μεταβλητών, ή άλλων τύπων δεδομένων. Για παράδειγμα:

```
print(42)
print(3.14)
name = "Alice"
print(name)
```

  Παρακάτω ακολουθεί ένας πίνακας που δείχνει περιπτώσεις λανθασμένου κώδικα με σκοπό την εκτύπωση του μηνύματος «Hello World!».

ΠΕΡΙΠΤΩΣΗ ΚΩΔΙΚΑ	ΣΩΣΤΟ/ΛΑΘΟΣ	ΑΙΤΙΟΛΟΓΗΣΗ
print(“Hello World!”)	ΣΩΣΤΟ	
print(‘Hello World!’)	ΣΩΣΤΟ	
print(Hello World!	ΛΑΘΟΣ	Λείπουν εισαγωγικά και δεξιά παρένθεση
print(“Hello World!)	ΛΑΘΟΣ	Λείπει το δεξί εισαγωγικό
print(“Hello World!”	ΛΑΘΟΣ	Λείπει η δεξιά παρένθεση
(“Hello World!”)	ΛΑΘΟΣ	Λείπει η εντολή print()
print(‘Hello World!”)	ΛΑΘΟΣ	Στην αρχή έχει μονό εισαγωγικό ενώ στο τέλος διπλό. Πρέπει να

		είναι ίδιου τύπου τόσο στην αρχή, όσο και στο τέλος (μονό ή διπλό εισαγωγικό)

- **Εισαγωγή τιμής**
  Όταν θέλουμε να ζητήσουμε από το χρήστη να εισάγει μια τιμή μέσω του πληκτρολογίου και να την αποθηκεύσουμε σε μια μεταβλητή, χρησιμοποιούμε την εντολή input(). Η εντολή αυτή εμφανίζει ένα μήνυμα στο χρήστη και περιμένει να πληκτρολογήσει μια τιμή. Αυτή η τιμή αποθηκεύεται στη μεταβλητή που έχουμε ορίσει. Για παράδειγμα:
  ilikia = input(“Poso xronwn eisai?”)
  name = input(“Pws se lene?”)

  Ισχύουν οι ίδιοι κανόνες σύνταξης, όπως στην εντολή print().

Εισαγωγή σχολίων

Τα σχόλια εισάγονται θέτοντας μπροστά από αυτά το σύμβολο #. Τα σχόλια σε ένα πρόγραμμα διευκολύνουν τη κατανόησή του. Οτιδήποτε βρίσκεται μετά το σύμβολο της δίεσης (#), αγνοείται από τον διερμηνευτή. Για παράδειγμα:

print(‘Hello World!’) #Εκτυπώνει το μήνυμα Hello World!, όπου η πρόταση μετά τη δίεση θεωρείται σχόλιο και αγνοείται από τον διερμηνευτή.

Βιβλιοθήκες

Εκτός από τις ενσωματωμένες συναρτήσεις που έχει η Python, υπάρχουν και άλλες συναρτήσεις που μπορείς να χρησιμοποιήσεις με τη χρήση βιβλιοθηκών. οι βιβλιοθήκες είναι σαν κουτιά εργαλείων γεμάτα με συναρτήσεις που μας βοηθούν να κάνουμε διάφορα πράγματα, όπως μαθηματικούς υπολογισμούς ή γραφικά.

Για να χρησιμοποιήσουμε αυτές τις βιβλιοθήκες στο πρόγραμμά μας, πρέπει πρώτα να τις εισάγουμε με τη λέξη-κλειδί import. Είναι σα να λέμε ότι θέλουμε να ανοίξουμε αυτό το κουτί εργαλείων και να χρησιμοποιήσουμε τα εργαλεία που υπάρχουν μέσα.

- **Βιβλιοθήκη math:**

  Θέλουμε να βρούμε την τετραγωνική ρίζα του αριθμού 25. Για να το κάνουμε αυτό, θα χρησιμοποιήσουμε τη μαθηματική βιβλιοθήκη, η οποία ονομάζεται math. Οπότε πρώτα εισάγουμε στο πρόγραμμά μας την εντολή «import math» για να χρησιμοποιήσουμε τις συναρτήσεις της. Για να βρούμε την τετραγωνική ρίζα, χρησιμοποιούμε τη συνάρτηση sqrt() που υπάρχει στη βιβλιοθήκη. Η πρόσβαση στη συνάρτηση γίνεται με το όνομα της βιβλιοθήκης και μια τελεία

(.), ακολουθούμενα από το όνομα της συνάρτησης, δηλαδή math.sqrt(25). Ακολουθεί το παρακάτω παράδειγμα για περισσότερη κατανόηση:

```
apotelesma = math.sqrt(25)

print(apotelesma)  # Αυτό θα τυπώσει το 5.0
```

Σημαντικό: Αν θέλαμε να τυπώσει τον αριθμό 5 ως ακέραιο, τότε θα έπρεπε να χρησιμοποιήσουμε την ενσωματομένη συνάρτηση int(). Ακολουθεί το ίδιο παράδειγμα με παραπάνω, τροποποιημένο στην ανάγκη μας:

```
apotelesma = int(math.sqrt(25))

print(apotelesma)  #Αυτό θα τυπώσει το 5
```

- **Βιβλιοθήκη random:**

Θέλουμε να δημιουργήσουμε ένα παιχνίδι στο οποίο θα ρίχνουμε ένα ζάρι που θα εμφανίζει τυχαίο αριθμό κάθε φορά. Για να το κάνουμε αυτό, θα χρησιμοποιήσουμε τη βιβλιοθήκη random. Οπότε πρώτα εισάγουμε στο πρόγραμμά μας την εντολή «import random» για να χρησιμοποιήσουμε τις συναρτήσεις της. Υπάρχουν δύο βασικές συναρτήσεις στη βιβλιοθήκη random που μας βοηθούν να δημιουργούμε τυχαίους αριθμούς:

1. **Randint():**
   Δίνει έναν τυχαίο ακέραιο αριθμό ανάμεσα σε δύο αριθμούς που καθορίζουμε εμείς, συμπεριλαμβάνοντας και τα δύο όρια που δίνουμε. Για παράδειγμα:

   ```
   import random
   arithmos = random.randint(1, 10)
   print(arithmos)  #Μπορεί να τυπώσει οποιονδήποτε αριθμό από το 1 έως το 10.
   ```

2. **Randrange() με δύο όρια:**
   Λειτουργεί παρόμοια με τη συνάρτηση randint(), με τη διαφορά ότι δεν συμπεριλαμβάνεται το τελευταίο όριο. Για παράδειγμα:

   ```
   import random
   arithmos = random.randrange(1, 10)
   print(arithmos)  #Μπορεί να τυπώσει οποιονδήποτε αριθμό από το 1 έως το 9.
   ```

3. **Randrange() με ένα όριο:**

Αν δώσουμε μόνο έναν αριθμό ως όριο, τότε η randrange θα παράγει έναν τυχαίο αριθμό από το 0 μέχρι έναν αριθμό λιγότερο από αυτόν που δώσαμε. Για παράδειγμα:

```
arithmos = random.randrange(10)
print(arithmos)  #Μπορεί να τυπώσει οποιονδήποτε αριθμό από το 0 έως το 9.
```

Αλγοριθμικές δομές - Ροές εκτέλεσης προγράμματος

- **Δομή ακολουθίας**
  Είναι σαν μια συνταγή μαγειρικής που εκτελούνται που εκτελούνται τα βήματα, το ένα μετά το άλλο με συγκεκριμένη σειρά. Αν αλλάξουμε τη σειρά, το αποτέλεσμα μπορεί να είναι διαφορετικό ή λανθασμένο. Επίσης δεν παραλείπεται κανένα βήμα. Όλα τα βήματα που είναι γραμμένα στο πρόγραμμά μας θα εκτελεστούν, ανεξάρτητα από τις συνθήκες και χωρίς εξαιρέσεις. Για παράδειγμα:

```
#Βήμα 1: Παίρνουμε τον πρώτο αριθμό
a = 5

#Βήμα 2: Παίρνουμε τον δεύτερο αριθμό
b = 10

#Βήμα 3: Προσθέτουμε τους δύο αριθμούς
result = a + b

#Βήμα 4: Τυπώνουμε το αποτέλεσμα
print(result)  #Αυτό θα τυπώσει το 15
```

- **Δομές επιλογής:**
  - **Απλή δομή επιλογής if**
    Η δομή επιλογής if (ΑΝ) χρησιμοποιείται, όταν θέλουμε να εκτελεστεί μια ακολουθία εντολών, μόνον, εφόσον πληρείται μία συγκεκριμένη συνθήκη. Με την δομή αυτή, ελέγχουμε αν η συνθήκη είναι αληθής η ψευδής. Αν η συνθήκη είναι αληθής τότε θα εκτελεστεί το περιεχόμενο της δομής if. Διαφορετικά, αν είναι η συνθήκη ψευδής, δε θα εκτελεστεί. Για παράδειγμα:

```
arithmos = 10

if (arithmos >= 5):

    print("Ο αριθμός είναι μεγαλύτερος από 5 ή ίσος με 5")
```

- **Σύνθετη δομή επιλογής if**
  Η δομή επιλογής if…else (ΑΝ…ΑΛΛΙΩΣ) χρησιμοποιείται όταν θέλουμε να εκτελεστεί μια ακολουθία εντολών αν μια συγκεκριμένη συνθήκη είναι αληθής, και μια άλλη ακολουθία εντολών αν η συνθήκη είναι ψευδής. Δηλαδή, με τη δομή if…else ελέγχουμε μια συνθήκη και, ανάλογα με το αν είναι αληθής ή ψευδής, εκτελούμε διαφορετικές εντολές.

```
arithmos = 10

if (arithmos >= 5):

        print("Ο αριθμός είναι μεγαλύτερος από 5 ή ίσος με 5")

else:
    print("Ο αριθμός είναι μικρότερος από 5")
```

- **Πολλαπλή επιλογή**
  Η δομή επιλογής if...elif...else (ΑΝ...ΑΛΛΙΩΣ ΑΝ…ΑΛΛΙΩΣ) χρησιμοποιείται για να ελέγξεις πολλές διαφορετικές συνθήκες. Είναι χρήσιμη όταν έχεις περισσότερες από δύο επιλογές και θέλεις να εκτελέσεις διαφορετικές εντολές ανάλογα με το ποια συνθήκη είναι αληθής. Για παράδειγμα:

```
arithmos = 0

if arithmos > 0:
        print("Ο αριθμός είναι θετικός")
elif arithmos < 0:
     print("Ο αριθμός είναι αρνητικός")
else:
    print("Ο αριθμός είναι μηδέν")
```

- **Εμφωλευμένες δομές επιλογής**
  Οι εμφωλευμένες δομές επιλογής σημαίνουν ότι μπορείς να έχεις μία δομή if...else μέσα σε μια άλλη. Αυτό είναι χρήσιμο όταν θέλεις να ελέγξεις μια δεύτερη συνθήκη, μόνο αφού μια πρώτη συνθήκη έχει αποδειχθεί αληθής ή ψευδής. Για παράδειγμα:

```
arithmos = 10
if arithmos >= 5:
    if arithmos == 5:
                print("Ο αριθμός είναι ίσος με 5")
    else:
                print("Ο αριθμός είναι μεγαλύτερος από 5")
else:
    print("Ο αριθμός είναι μικρότερος από 5")
```

- **Δομή επανάληψης (for και while)**
  Όταν γράφουμε πρόγραμμα, μερικές φορές χρειαζόμαστε να επαναλάβουμε κάποιες εντολές πολλές φορές. Υπάρχουν δύο ήδη επαναλήψεων:
  - **Προκαθορισμένες Επαναλήψεις:**
    Γνωρίζουμε τον ακριβή αριθμό που θα εκτελεστεί η επανάληψη. Για παράδειγμα:

    ```
    for i in range(5):
        print(“Hello World!”)
    ```

    όπου η επανάληψη εκτελείται για 5 φορές, εκτυπώνοντας το μήνυμα “Hello World”.

    Η range() είναι μία συνάρτηση που μας βοηθά να δημιουργούμε σειρές αριθμών, που μπορούμε να χρησιμοποιήσουμε σε επαναλήψεις. Η συνάρτηση range() μπορεί να πάρει μέχρι τρεις αριθμούς:

    - Αρχή: Από ποιον αριθμό να ξεκινήσει.
    - Μέχρι: Μέχρι ποιον αριθμό να πάει (αλλά χωρίς να τον συμπεριλάβει).
    - Βήμα: Πόσο να αυξάνει ή να μειώνει κάθε φορά.

    Για παράδειγμα:

    1. range(10):
       Θα ξεκινήσει από το 0 και θα πάει μέχρι το 10, αλλά χωρίς να συμπεριλάβει το 10. Δηλαδή η σειρά που δημιουργείται είναι: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].
    2. range(1, 8):
       Θα ξεκινήσει από το 1 και θα πάει μέχρι το 8 (χωρίς να το συμπεριλάβει). Δηλαδή η σειρά είναι: [1, 2, 3, 4, 5, 6, 7].
    3. range(8, -1, -1):
       Θα ξεκινήσει από το 8 και θα πάει μέχρι το -1, με βήμα -1 (δηλαδή μείωση κατά 1 κάθε φορά). Η σειρά που προκύπτει δηλαδή είναι: [8, 7, 6, 5, 4, 3, 2, 1, 0].

  - **Μη Προκαθορισμένες Επαναλήψεις:**
    Δε γνωρίζουμε από πριν τον ακριβή αριθμό που θα εκτελεστεί η επανάληψη. Εξαρτάται από κάποια συνθήκη που αλλάζει κατά τη διάρκεια της επανάληψης. Σε κάθε επανάληψη πραγματοποιείται ο έλεγχος της συνθήκης, πριν από την εκτέλεση των εντολών του βρόχου, πράγμα που σημαίνει ότι υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου. Για παράδειγμα:

```
pontoi = 0
while pontoi < 100:
        pontoi = pontoi + 1   #Αυξάνεται ο πόντος κατά 1
```

όπου η επανάληψη εκτελείται όσο η συνθήκη είναι αληθής. Όταν οι πόντοι φτάσουν τους 100, η επανάληψη σταματάει.

- **Συναρτήσεις**

Οι συναρτήσεις είναι κομμάτια κώδικα που επαναχρησιμοποιούνται για να εκτελέσουν μία λειτουργία. Μας επιτρέπεται να ονομάζουμε τη κάθε συνάρτηση και να την εκτελούμε καλώντας το όνομά της όσες φορές χρειάζεται. Αυτό μας βοηθά να κάνουμε τον κώδικά μας πιο οργανωμένο και να τον επαναχρησιμοποιούμε εύκολα.
Για να φτιάξουμε τη δική μας συνάρτηση, χρησιμοποιούμε τη λέξη def, που σημαίνει "ορίζω". Έπειτα, γράφουμε το όνομα που θέλουμε να δώσουμε στη συνάρτηση και στη συνέχεια βάζουμε παρενθέσεις (). Μέσα σε αυτές τις παρενθέσεις μπορούμε να βάλουμε μεταβλητές (πράγματα που θέλουμε να χρησιμοποιήσει η συνάρτηση), αλλά αυτό δεν είναι πάντα απαραίτητο. Τέλος, προσθέτουμε άνω και κάτω τελεία : και από την επόμενη γραμμή γράφουμε τι θέλουμε να κάνει η συνάρτηση. Για παράδειγμα:

```
def hello():
        print('Hello World!')
```

Παρατηρούμε ότι φτιάξαμε μια συνάρτηση με το όνομα hello. Μέσα στη συνάρτηση, γράψαμε την εντολή print('Hello World!') που εκτυπώνει το μήνυμα. Για να «τρέξουμε» τη συνάρτηση και να δούμε το μήνυμα, απλά καλούμε το όνομα της συνάρτησης:

```
hello()
```

Επιπλέον μπορούμε να δημιουργήσουμε μία συνάρτηση που καλεί μία άλλη. Για παράδειγμα:

```
def epanalave_hello():
        hello()
        hello()
```

Έτσι η συνάρτηση epanalave_hello όταν θα καλείται, θα εκτελεί δύο φορές τη συνάρτηση hello().

Όταν φτιάχνουμε μια συνάρτηση, συχνά θέλουμε να της δώσουμε κάποια δεδομένα για να δουλέψει. Αυτά τα δεδομένα, τα δίνουμε στη συνάρτηση μέσω

παραμέτρων. Όταν καλούμε τη συνάρτηση και της δίνουμε συγκεκριμένες τιμές για αυτές τις παραμέτρους, αυτές οι τιμές ονομάζονται ορίσματα. Για παράδειγμα:

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

όπου η συνάρτηση επιστρέφει το γινόμενο ενός πολλαπλασιασμού δύο ορισμάτων. Στις παραμέτρους a και b, τοποθετούμε τους αριθμούς που θέλουμε να πολλαπλασιάσουμε. Στη μεταβλητή x αποθηκεύεται το γινόμενο των αριθμών και επιστρέφει το αποτέλεσμα με την εντολή return όταν καλέσουμε τη συνάρτηση. Αν θέλουμε να καλέσουμε τη συνάρτηση, πρέπει να δώσουμε συγκεκριμένες τιμές στις παραμέτρους. Δηλαδή:

```
print(ginomeno(5, 10))
```

όπου η παράμετρος a παίρνει την τιμή 5 και η παράμετρος b παίρνει την τιμή 10. Η συνάρτηση υπολογίζει το γινόμενο αυτών των δύο αριθμών και επιστρέφει το αποτέλεσμα, το οποίο εκτυπώνεται στην οθόνη με την εντολή print().

- **Παραδείγματα**

  - **Άσκηση 1:**

Δημιούργησε μια λίστα με τα αγαπημένα σου φαγητά. Στη συνέχεια:

Πρόσθεσε ένα νέο φαγητό στο τέλος της λίστας.
Αφαίρεσε το πρώτο στοιχείο της λίστας.
Αντικατάστησε το τελευταίο φαγητό με ένα άλλο.

Λύση:

```
# Λίστα με αγαπημένα φαγητά
fagita = ["Pizza", "Pasta", "Burger", "Sushi"]

# Προσθήκη ενός νέου φαγητού
fagita.append("Salata")
print(fagita)

# Αφαίρεση του πρώτου στοιχείου
fagita.pop(0)
print(fagita)

# Αντικατάσταση του τελευταίου στοιχείου
```

```
fagita[-1] = "Gyros"
print(fagita)
```

Το αποτέλεσμα που φορτώνει στο terminal:

```
["Pizza", "Pasta", "Burger", "Sushi", "Salata"]
["Pasta", "Burger", "Sushi", "Salata"]
["Pasta", "Burger", "Sushi", "Gyros"]
```

- **Άσκηση 2:**

Δημιούργησε μια μεταβλητή vathmos που περιέχει τον βαθμό σου (ακέραιος αριθμός). Γράψε ένα πρόγραμμα που ελέγχει αν ο βαθμός είναι:

Πάνω από 18: Εκτύπωσε "Άριστα".
Από 15 έως 17: Εκτύπωσε "Πολύ καλά".
Κάτω από 15: Εκτύπωσε "Χρειάζεται βελτίωση".

Λύση:

```
# Βαθμός
vathmos = int(input("Ποιος είναι ο βαθμός σου; "))

# Έλεγχος βαθμού
if vathmos > 18:
   print("Άριστα")
elif 15 <= vathmos <= 17:
   print("Πολύ καλά")
else:
   print("Χρειάζεται βελτίωση")
```

Σημαντικό: Το αποτέλεσμα του κώδικα εξαρτάται εξ ολοκλήρου από την τιμή που θα εισάγει ο χρήστης. Ανάλογα με την απάντησή του, θα εμφανιστεί στο terminal το αντίστοιχο μήνυμα.

- **Άσκηση 3:**

Δημιούργησε μια συνάρτηση pollaplasiasmos που δέχεται δύο αριθμούς και επιστρέφει το γινόμενό τους. Χρησιμοποίησε αυτή τη συνάρτηση για να υπολογίσεις και να εκτυπώσεις το γινόμενο για τα ζεύγη αριθμών που περιέχονται στη λίστα zevgi (2, 3), (5, 10), (8, 7).

Λύση:

```
# Συνάρτηση για πολλαπλασιασμό
```

```
def pollaplasiasmos(a, b):
return a * b


# Λίστα με ζεύγη αριθμών
zevgi = [(2, 3), (5, 10), (8, 7)]


# Υπολογισμός και εκτύπωση γινομένου για κάθε ζεύγος
for z in zevgi:
apotelesma = pollaplasiasmos(z[0], z[1])
print(f"Το γινόμενο των {z[0]} και {z[1]} είναι: {apotelesma}")
```

Το αποτέλεσμα που φορτώνει στο terminal:

Το γινόμενο των 2 και 3 είναι: 6

Το γινόμενο των 5 και 10 είναι: 50

Το γινόμενο των 8 και 7 είναι: 56

- **Άσκηση 4:**

Γράψε ένα πρόγραμμα που χρησιμοποιεί τη βιβλιοθήκη random για να προσομοιώσει τη ρίψη ενός ζαριού 10 φορές. Για κάθε ρίψη, να εκτυπώνεται το αποτέλεσμα και στο τέλος να εμφανίζεται το μεγαλύτερο αποτέλεσμα που προέκυψε από τις ρίψεις.

Λύση:

```
import random

# Λίστα για να αποθηκεύσουμε τα αποτελέσματα των ρίψεων
apotelesmata = []

# Ρίψη ζαριού 10 φορές
for i in range(10):
    zari = random.randint(1, 6)  # Παράγει τυχαίο αριθμό από 1 έως 6
    apotelesmata.append(zari)
    print(f"Ρίψη {i+1}: {zari}")

# Εύρεση του μεγαλύτερου αποτελέσματος
```

```
megalytero_apotelesma = max(apotelesmata)
print(f"Το μεγαλύτερο αποτέλεσμα από τις ρίψεις είναι:
{megalytero_apotelesma}")
```

Το αποτέλεσμα που φορτώνει στο terminal:

```
Ρίψη 1: 2
Ρίψη 2: 6
Ρίψη 3: 2
Ρίψη 4: 2
Ρίψη 5: 1
Ρίψη 6: 4
Ρίψη 7: 2
Ρίψη 8: 3
Ρίψη 9: 2
Ρίψη 10: 3
Το μεγαλύτερο αποτέλεσμα από τις ρίψεις είναι: 6
```

Σημαντικό: Ο αριθμός που προκύπτει από κάθε ρίψη του ζαριού είναι τυχαίος λόγω της βιβλιοθήκης random. Αυτό σημαίνει ότι όταν θα τρέχουμε τον κώδικα, το αποτέλεσμα μπορεί να είναι διαφορετικό.

- **Άσκηση 5:**

Δημιούργησε μια λίστα με τα ονόματα Ioannis, Maria, Kostas, Eleni. Στη συνέχεια, ζήτα από τον χρήστη να εισάγει το όνομά του. Το πρόγραμμα πρέπει να διαγράψει όλα τα υπόλοιπα ονόματα από τη λίστα, αφήνοντας μόνο το όνομα που εισήγαγε ο χρήστης. Αν το όνομα του χρήστη δεν υπάρχει στη λίστα, το πρόγραμμα πρέπει να προσθέσει το όνομα του χρήστη στη λίστα και να αφαιρέσει όλα τα υπόλοιπα ονόματα.

Λύση:

```
# Λίστα με ονόματα
onomata = ["Ioannis", "Maria", "Kostas", "Eleni"]

# Ζήτηση ονόματος από τον χρήστη
onoma_xristi = input("Ποιο είναι το όνομά σου; ")

# Έλεγχος αν το όνομα του χρήστη είναι στη λίστα
if onoma_xristi in onomata:
    # Κρατάμε μόνο το όνομα του χρήστη
    onomata = [onoma_xristi]
    print("Το όνομα υπήρχε στη λίστα.")
```

```
else:
    # Αν το όνομα δεν υπάρχει, προσθέτουμε το όνομα του χρήστη και
αφαιρούμε τα υπόλοιπα
    onomata = [onoma_xristi]
    print("Το όνομα δεν υπήρχε στη λίστα.")

# Εκτύπωση της νέας λίστας
print("Η νέα λίστα είναι:", onomata)
```

Το αποτέλεσμα που φορτώνει στο terminal:

```
Ποιο είναι το όνομά σου; Christos
Η νέα λίστα είναι: ['Christos']
```