Κεφάλαιο 4ο
Αλγοριθμικές Δομές

Εισαγωγή

Ένα πρόγραμμα λαμβάνει δεδομένα ως είσοδο, τα επεξεργάζεται σύμφωνα με τις εντολές που περιέχει και στη συνέχεια επιστρέφει τα αποτελέσματα. Έτσι, τα βασικά χαρακτηριστικά ενός προγράμματος είναι:

- Είσοδος
- Τρόπος εκτέλεσης
- Έξοδος

Είσοδος:
Εισάγονται τα δεδομένα που απαιτεί το πρόγραμμα από το περιβάλλον για να παράγει την επιθυμητή έξοδο.

- Με τη **συνάρτηση input ()**, το πρόγραμμα διαβάζει από το πληκτρολόγιο δεδομένα.

Τρόπος Εκτέλεσης:
Γίνεται η επεξεργασία των δεδομένων. Εμπεριέχει μεταξύ άλλων τις ροές εκτέλεσης (ακολουθία, επιλογή, επανάληψη), τα μοντέλα υπολογισμού, τις συναρτήσεις, τις μεταβλητές

Έξοδος:
Εξάγονται τα τελικά αποτελέσματα που παράγονται και συνήθως εμφανίζονται ή τυπώνονται για το χρήστη, με την έξοδο να πραγματοποιείται συνήθως στην οθόνη ή στον εκτυπωτή. Μπορεί να έχει τη μορφή κειμένου/αριθμού ή και γραφικών, όπως για παράδειγμα ένα πρόγραμμα που δημιουργεί ένα παιχνίδι.

- Βασική εντολή εξόδου της γλώσσας προγραμματισμού Python η εντολή **print**.

Αλγοριθμικές δομές - Ροές εκτέλεσης προγράμματος

- Δομή ακολουθίας
- Δομή επιλογής
- Δομή επανάληψης
- Συναρτήσεις.

4.1 Αλγοριθμικές δομές - Ροές εκτέλεσης προγράμματος

4.1.1 Ακολουθία

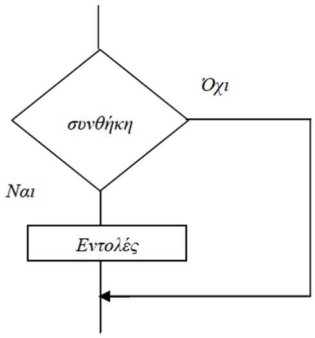
Πρόκειται για μια σειρά από εντολές που εκτελούνται η μία μετά την άλλη, ώστε να δοθεί στην έξοδο ένα επιθυμητό αποτέλεσμα. Η δομή ακολουθίας χρησιμοποιείται πρακτικά για την επίλυση απλών προβλημάτων, όπου είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών.
Χρησιμοποιώντας, αποκλειστικά, τη δομή ακολουθίας, μπορούμε να λύσουμε περιορισμένα προβλήματα στα οποία:

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

4.1.2 Δομή επιλογής if

Συχνά μέσα σε ένα πρόγραμμα χρειάζεται να πούμε τι να κάνει όταν συναντήσει μία μελλοντική κατάσταση, την οποία δεν μπορούμε να ξέρουμε από πριν, αν θα τη συναντήσει και πότε, π.χ. να υπερβεί μία μεταβλητή μία συγκεκριμένη τιμή. Σε αυτήν την περίπτωση θα πρέπει να έχουμε φτιάξει τον κώδικα του προγράμματος μας κατάλληλα, ώστε να είναι σε θέση τη στιγμή εκείνη να πάρει μια απόφαση για τι θα κάνει, στηριζόμενη στην τιμή κάποιας παραμέτρου, η οποία μπορεί να είναι η τιμή μίας μεταβλητής την οποία συγκρίνουμε με μία συγκεκριμένη τιμή.
Η δομή επιλογής χρησιμοποιείται σε προβλήματα όπου χρειάζεται να ληφθούν κάποιες αποφάσεις με βάση κάποια συνθήκη, που μπορεί να είναι διαφορετική για κάθε πρόβλημα (δηλ. λύση του προβλήματος με διαφορετικά δεδομένα εισόδου). Η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης με δύο δυνατές τιμές (Αληθής ή Ψευδής) και στη συνέχεια την απόφαση εκτέλεσης κάποιας εντολής
Με μία εντολή επιλογής δημιουργούνται μέσα στο πρόγραμμα δύο πιθανές διαδρομές (κλάδοι) τις οποίες μπορεί ακολουθήσει το πρόγραμμα μας. Από τους δύο κλάδους αυτούς, το πρόγραμμα θα ακολουθήσει μόνο το έναν, ανάλογα με την κατάσταση που θα έχει διαμορφωθεί εκείνη τη στιγμή, μετά από την πραγματοποίηση ενός ελέγχου.

Απλή Επιλογή (Αν συνθήκη Τότε): Αν θέλουμε να εκτελεστεί μια ακολουθία εντολών, μόνον εφόσον πληρείται μία συγκεκριμένη συνθήκη, τότε χρησιμοποιούμε τη δομή επιλογής if (ΑΝ) με τη συνθήκη την οποία θέλουμε να ελέγξουμε.
Αν τη στιγμή του ελέγχου της συνθήκης, αυτή είναι ΑΛΗΘΗΣ θα εκτελεστούν οι εντολές που βρίσκονται στο εσωτερικό της εντολής αυτής. Διαφορετικά το πρόγραμμα θα τις αγνοήσεις και θα συνεχίσει με την εκτέλεση των εντολών που βρίσκονται αμέσως μετά την εντολή της απλής επιλογής.
Αν η **συνθήκη** είναι **αληθής (True), τότε** το σύνολο των εντολών που περιέχονται στη δομή **if**, θα εκτελεστεί, αλλιώς η ροή του προγράμματος θα προσπεράσει τη δομή if και θα συνεχίσει από το τέλος της if.

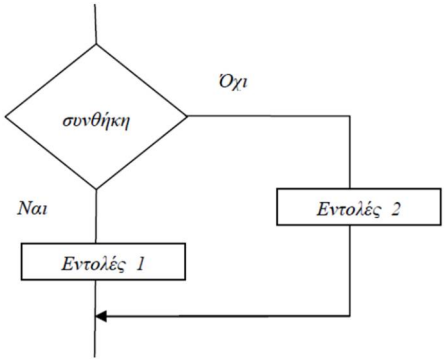


Σύνταξη

```
if <συνθήκη ελέγχου>:
    <εντολές>
```

Σύνθετη Επιλογή (Αν συνθήκη Τότε - Αλλιώς): Αν θέλουμε να εκτελεστεί μια ακολουθία εντολών, μόνον εφόσον πληρείται μία συγκεκριμένη συνθήκη διαφορετικά να εκτελεστεί μία διαφορετική ακολουθία εντολών τότε χρησιμοποιούμε τη δομή επιλογής if…else (ΑΝ…ΑΛΛΙΩΣ**)** με τη συνθήκη την οποία θέλουμε να ελέγξουμε.
Αν τη στιγμή του ελέγχου της συνθήκης αυτή είναι ΑΛΗΘΗΣ θα εκτελεστούν οι εντολές που βρίσκονται στο επάνω άνοιγμα της εντολής, ανάμεσα στο τότε και στο αλλιώς, δηλαδή αυτές που βρίσκονται στο εσωτερικό του τμήματος if. Στην αντίθετη περίπτωση (αν η συνθήκη είναι ΨΕΥΔΗΣ, δηλ δεν ισχύει) θα εκτελεστούν οι εντολές που βρίσκονται στο κάτω άνοιγμα της εντολής, από το αλλιώς και κάτω, δηλαδή αυτές που βρίσκονται στο εσωτερικό του τμήματος else.

Αν η **συνθήκη** είναι **αληθής (True),** θα εκτελεστεί το μπλοκ εντολών της **if**, αλλιώς, αν δεν ισχύει, είναι δηλαδή **ψευδής (False)**, θα εκτελεστεί το μπλοκ εντολών της **else**.

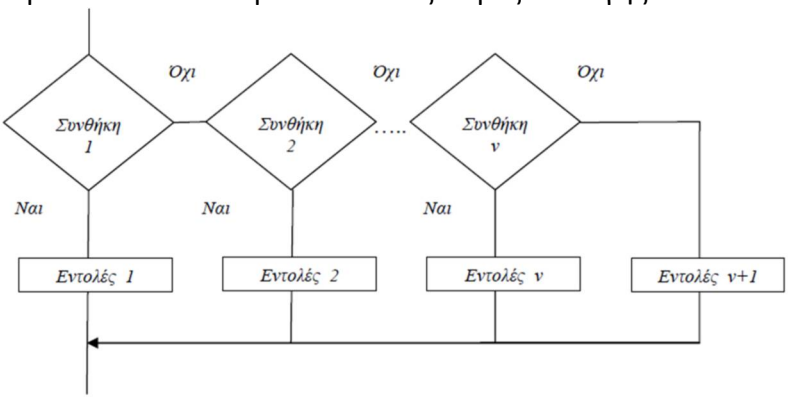


Η εντολή ελέγχου ΑΝ_ΑΛΛΙΩΣ **συντάσσεται** ως:

if <συνθήκη ελέγχου>:
    <εντολές_1>
else:
    <εντολές_2>

Ξεκινάει με μία **επικεφαλίδα** (**header**) που αποτελείται από τη δεσμευμένη λέξη **if**, μία συνθήκη και τελειώνει με μία άνω κάτω τελεία (:). Η συνθήκη μπορεί να είναι μία οποιαδήποτε λογική έκφραση τύπου Bool που αποτιμάται σε True ή False. Αν η συνθήκη είναι αληθής (True), τότε εκτελείται το πρώτο μπλοκ εντολών (true_block), αλλιώς, αν είναι ψευδής (False), εκτελείται το δεύτερο μπλοκ εντολών (false_block). Ακριβώς πριν το δεύτερο μπλοκ εντολών τοποθετείται η δεσμευμένη λέξη else ακολουθούμενη από άνω και κάτω τελεία ‘ : ‘.
Οι εντολές ενός μπλοκ πρέπει να είναι μετατοπισμένες προς τα δεξιά. Η τυπική μετατόπιση ή **εσοχή** (**indentation**) των εντολών είναι τέσσερα κενά. Η Python μας βοηθάει σε αυτό ρυθμίζοντας αυτόματα τις εσοχές για μας, απλά πατώντας Enter μετά την πληκτρολόγηση της άνω κάτω τελείας ‘ : ‘. Δεν πρέπει να διαγράψουμε αυτά τα κενά διαστήματα. Η έλλειψη ενός κενού ή η ύπαρξη επιπλέον κενών μπορεί να οδηγήσει σε λάθος ή απρόσμενη συμπεριφορά σε ένα πρόγραμμα.
Η πρώτη εντολή που ευθυγραμμίζεται με το αριστερό περιθώριο είναι η πρώτη εντολή έξω από το μπλοκ.

Πολλαπλή Επιλογή
Όταν οι εναλλακτικές περιπτώσεις είναι περισσότερες από δύο, τότε μπορεί να χρησιμοποιηθεί η πολλαπλή δομή επιλογής, η οποία στην ουσία αποτελείται από πολλαπλές δομές επιλογής κάθε μία μέσα στο δεύτερο κλάδο - τον κλάδο αλλιώς που χρησιμοποιείται στην περίπτωση που η συνθήκη είναι ΨΕΥΔΗΣ - της άλλης. Για τη σύνταξη αυτής της δομής χρησιμοποιούμε τον παρακάτω συνδυασμό από Απλές Δομές Επιλογής



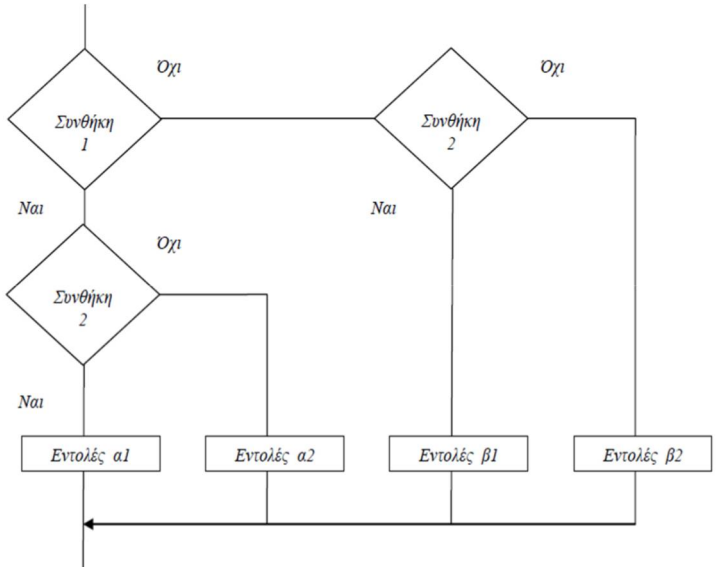
Η Python προσφέρει τη δυνατότητα για σύνταξη σύνθετων δομών επιλογής με τη χρήση της εντολής elif. Η **σύνταξη** είναι ως εξής:

```
if <συνθήκη>:
    <εντολές_1>
elif <συνθήκη2>:
    <εντολές_2>
else:
    <εντολές_3>
```

ΠΡΟΣΟΧΗ : Στο τέλος των εντολών if, elif και else τοποθετούμε άνω και κάτω τελεία (:).

Εμφωλευμένη Επιλογή

Όταν οι εναλλακτικές περιπτώσεις είναι περισσότερες από δύο, τότε μπορεί να χρησιμοποιηθεί η η εμφωλευμένη δομή, η οποία στην ουσία αποτελείται από πολλές δομές επιλογής η μία μέσα στην άλλη,. Για τη σύνταξη αυτής της δομής χρησιμοποιούμε τον παρακάτω συνδυασμό από Απλές Δομές Επιλογής



Η **σύνταξη** είναι ως εξής

```
if <συνθήκη>:
    <εντολές_1>
    if <συνθήκη>:
        <εντολές_1.1>
    elif <συνθήκη1.2>:
        <εντολές_1.2>
    else:
        <εντολές_.13>
elif <συνθήκη2>:
    <εντολές_2>
    if <συνθήκη>:
        <εντολές_2.1>
    elif <συνθήκη2.2>:
        <εντολές_2.2>
    else:
        <εντολές_2.3>
else:
    <εντολές_3>
```

4.1.3 Δομή επανάληψης (for και while)

Συχνά, ορισμένοι υπολογισμοί σε ένα πρόγραμμα είναι αναγκαίο να εκτελούνται περισσότερες από μία φορές, οπότε χρειάζεται να επαναλάβουμε μια συγκεκριμένη σειρά ενεργειών περισσότερες από μία φορές.
Για να το πετύχουμε αυτό χρησιμοποιούμε μία από τις βασικές δομές στον προγραμματισμό, τη δομή επανάληψης, η οποία μας δίνει τη δυνατότητα να επαναλάβουμε μία ή περισσότερες ενέργειές μας, που τις εσωκλείουμε σε αυτή τη δομή.
Μπορούμε να υλοποιήσουμε τις παρακάτω δομές επανάληψης :

- **Δομή Επανάληψης (Όσο)**
- **Δομή Επανάληψης (Για)**

Υπάρχουν δύο τύποι επαναλήψεων:

- Οι προκαθορισμένοι, όπου το πλήθος των επαναλήψεων είναι δεδομένο πριν αρχίσουν οι επαναλήψεις.
  Για παράδειγμα: ο υπολογισμός του μέσου όρου βαθμολογίας των μαθητών ενός τμήματος 25 μαθητών.
- Οι μη προκαθορισμένοι ή απροσδιόριστοι, όπου το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών του σώματος της επανάληψης.
  Για παράδειγμα: ο υπολογισμός των μορίων όσων υποβάλλουν αίτηση σε ένα διαγωνισμό του Δημοσίου.

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

4.1.3.1 Προσεγγίζοντας τη συνάρτηση range ()

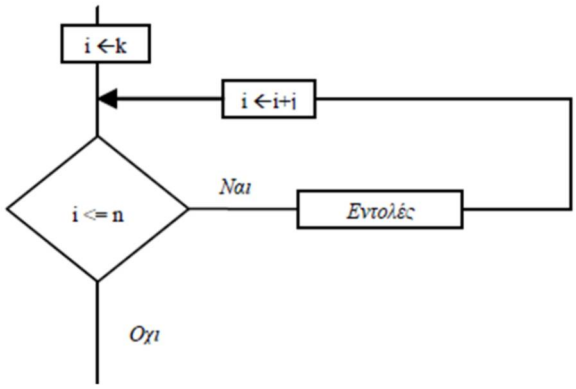
Η **range()** είναι μια ενσωματωμένη συνάρτηση της γλώσσας Python, η οποία, ανάμεσα σε άλλα, χρησιμοποιείται για την υπόδειξη του αριθμού των επαναλήψεων που θα εκτελεστούν σε ένα βρόχο.
Η δομή της συνάρτησης **range** είναι της μορφής **(αρχή, μέχρι, βήμα)**, όπου αρχή, μέχρι, βήμα, είναι ακέραιοι αριθμοί. Η συνάρτηση range επιστρέφει μία ακολουθία αριθμών αρχίζοντας από την ένδειξης της *αρχής* μέχρι *την* ένδειξη *μέχρι*, αλλά χωρίς να την εμπεριέχει, με βήμα την ένδειξη *βήμα.* Οι ενδείξεις της *αρχής* και του *βήματος* δεν είναι υποχρεωτικές, αλλά μπορεί να προστεθούν. Αντίθετα η ένδειξη *μέχρι*, πρέπει πάντα να αναφέρεται.
Αναλυτικά :
Η ένδειξη της *αρχής* δεν είναι υποχρεωτική και, αν δεν αναφέρεται, θα αρχίσει από 0.
Η ένδειξη του *βήματος* δεν είναι υποχρεωτική και, αν δεν αναφέρεται, θα συνεχίσει με βήμα 1.
Αντίθετα η ένδειξη *μέχρι* πρέπει πάντα να αναφέρεται. Η ένδειξη αυτή δεν εμπεριέχεται στα αποτελέσματα.

Δραστηριότητα

Πειραματιστείτε με τη χρήση της συνάρτησης range στο προγραμματιστικό περιβάλλον της Python για να δείτε τι παράγει η συνάρτηση. Για παράδειγμα:
Η εντολή **range** (10) παράγει τη λίστα: [0,1,2,3,4,5,6,7,8,9].
Η εντολή **range** (1, 8) παράγει τη λίστα: [1,2,3,4,5,6,7]
Η εντολή **range** (0, 35, 5) παράγει τη λίστα: [0,5,10,15,20,25,30]
Η εντολή **range** (8, -1, -1) παράγει τη λίστα [8, 7, 6, 5, 4, 3, 2, 1, 0]

4.1.3 Δομή επανάληψης for

Δομή Επανάληψης (Για)
Πρόκειται και πάλι για μία εντολή επανάληψης ένα συγκεκριμένο αριθμό φορών.
Ο βρόχος **for** (Για … μέχρι) χρησιμοποιείται για ένα προκαθορισμένο αριθμό επαναλήψεων.



Σύνταξη

for μεταβλητή **in range** (αρχή, μέχρι, βήμα):
<εντολές>

Ένας βρόγχος for ξεκινάει πάντα με τη δεσμευμένη λέξη **for**, έπειτα ακολουθεί μία μεταβλητή, στη συνέχεια η λέξη κλειδί **in**, η οποία συνήθως ακολουθείται από τη συνάρτηση **range**, και τελειώνει με άνω κάτω τελεία ‘ : ’ . Το μπλοκ των εντολών του βρόγχου πρέπει να βρίσκεται σε εσοχή, όπως είδαμε και στην εντολή if.
Αν θέλουμε να επαναλάβουμε ένα μπλοκ εντολών **N** φορές θα πρέπει να φροντίσουμε ώστε η ένδειξη *μέχρι* να περιέχει τον **αριθμό N + βήμα**, πχ N+1 για βήμα = 1
ΠΡΟΣΟΧΗ : Στο τέλος της εντολής for τοποθετούμε άνω και κάτω τελεία ‘ : ’.

Η εντολή for κάνει επανάληψη διατρέχοντας τα στοιχεία μιας ακολουθίας, για παράδειγμα μιας συμβολοσειράς ή μιας λίστας με τη σειρά με την οποία αυτά εμφανίζονται στην ακολουθία.

Δραστηριότητες

α) Τι θα εμφανίσει το ακόλουθο τμήμα προγράμματος Python;

```
for i in range (1, 4):
        for j in range (1, 4):
                print(i, j)
```

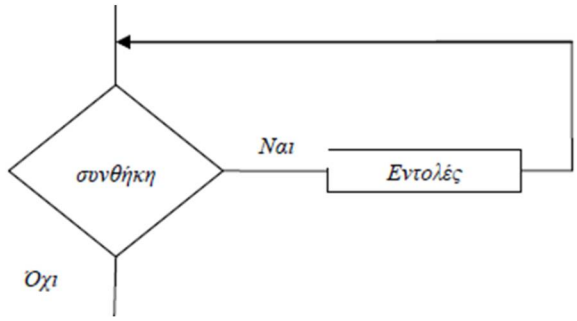
β) Τι θα εμφανίσει το ακόλουθο τμήμα προγράμματος Python;

```
for i in range (1, 11):
    for j in range (1, 11):
        x = I * j
      print i, ' x ', j, ' = ', x
```

4.1.4 Δομή επανάληψης με βρόχο while

Δομή Επανάληψης (Όσο)
Οι υπολογιστές χρησιμοποιούνται συχνά, για να κάνουν ανιαρές και επαναλαμβανόμενες εργασίες. Η επανάληψη ίδιων ή παραπλήσιων εργασιών χωρίς λάθη είναι κάτι που το κάνουν καλά οι υπολογιστές σε αντίθεση με το ανθρώπινο μυαλό. Η Python μάς προσφέρει την εντολή while, η οποία επαναλαμβανόμενα εκτελεί μία ομάδα (μπλοκ) εντολών, όσο μια συνθήκη (προϋπόθεση) παραμένει αληθής.
Σε αυτήν την περίπτωση επανάληψης θέλουμε μία σειρά ενεργειών να επαναλαμβάνεται συνέχεια όχι όμως για πάντα, αλλά για όσο διάστημα υπάρχει μία δεδομένη κατάσταση δηλ. ικανοποιείται μία συνθήκη.

Σημείωση: Στη δομή αυτή οι εντολές που βρίσκονται στο εσωτερικό της μπορεί να μην εκτελεστούν καμία φορά αν η συνθήκη είναι ψευδής από την αρχή. Οι εντολές θα εκτελεσθούν μόνο αν η συνθήκη είναι αληθής στην αρχή, και θα εκτελούνται μέχρι η συνθήκη να γίνει ψευδή. Με άλλα λόγια ο βρόχος **while** (ή Όσο <συνθήκη> επανάλαβε) χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων, όπου υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου, με τον έλεγχο της συνθήκης να πραγματοποιείται πριν από την εκτέλεση των εντολών του βρόχου.



Σύνταξη

Αρχική τιμή στη μεταβλητή
while μεταβλητή <συνθήκη>:
    <εντολές>

Ένας βρόχος while ξεκινάει με μία επικεφαλίδα που περιλαμβάνει τη δεσμευμένη λέξη while, μία **συνθήκη** (οποιαδήποτε έκφραση τύπου Bool) και τελειώνει με μία άνω κάτω τελεία ‘ : ’. Στη συνέχεια ακολουθεί σε εσοχή ένα μπλοκ εντολών («σώμα» του βρόγχου while) που εκτελούνται επαναλαμβανόμενα, όσο η συνθήκη είναι αληθής. Όταν η συνθήκη γίνει ψευδής, το «σώμα» δεν εκτελείται και η ροή εκτέλεσης των εντολών μεταφέρεται στην πρώτη εντολή μετά το βρόχο while. Αν η συνθήκη είναι ψευδής από την πρώτη φορά που ελέγχεται, το «σώμα» δεν εκτελείται ποτέ. Το «σώμα» του βρόχου θα πρέπει να αλλάζει τις τιμές μιας ή περισσότερων μεταβλητών, τις οποίες ελέγχει η συνθήκη στην επικεφαλίδα, ώστε η συνθήκη να γίνεται κάποια στιγμή ψευδής και ο βρόχος να τερματίζεται. Διαφορετικά, ο βρόχος θα επαναλαμβάνεται επ’ άπειρον, γι΄αυτό και στην περίπτωση αυτή τον ονομάζουμε **ατέρμων βρόχος** (**infinite loop**). Κατά τη συγγραφή ενός προγράμματος δεν είναι πάντα εύκολο να αποδείξουμε ότι ένας βρόχος while δεν θα είναι ατέρμων. Επίσης, η μεταβλητή ή οι μεταβλητές που αναφέρονται στη συνθήκη θα πρέπει να έχουν πάρει στον κώδικά μας αρχικές τιμές με τις λεγόμενες εντολές αρχικοποίησης πριν να ξεκινήσουμε τη σύνταξη του βρόχου
ΠΡΟΣΟΧΗ : Στο τέλος της εντολής while τοποθετούμε άνω και κάτω τελεία ‘ : ’.

Παράδειγμα

```
# Πρόγραμμα επανάληψη
x = 40            # αρχική τιμή στη  μεταβλητή x
while x < 50 :    # έλεγχος της  επανάληψης
      x = x + 2   # αύξηση του μετρητή x
      print x
```

ΑΡΧΗ: Πριν το while θα πρέπει να δώσουμε μία τιμή στη μεταβλητή που ελέγχει τη συνθήκη, ώστε ανάλογα να εκτελεστεί ή όχι ο βρόχος.
ΈΛΕΓΧΟΣ: Μετά το while πραγματοποιείται (και σε κάθε επανάληψη των εντολών του βρόχου) ο έλεγχος της συνθήκης, πριν από την εκτέλεση των εντολών του βρόχου. Αυτός ο έλεγχος καθορίζει και τον αριθμό των επαναλήψεων που θα συμβούν. Υπάρχει βεβαίως και η περίπτωση και μην εκτελεστούν καθόλου οι εντολές του βρόχου.
ΣΕ ΚΑΘΕ ΕΠΑΝΑΛΗΨΗ : θα πρέπει μέσα στο μπλοκ εντολών να υπάρχει κατάλληλη εντολή, ώστε κάποια στιγμή η συνθήκη να γίνεται ψευδής και ο βρόχος να τερματίζει.

Σημείωση1: Θα πρέπει μέσα στο μπλοκ εντολών να υπάρχει κατάλληλη εντολή, ώστε να εξασφαλίζεται ότι κάποια στιγμή η συνθήκη θα γίνει ψευδής και θα διακοπεί ο βρόχος. Διαφορετικά, ο βρόχος δε θα τερματίζει.
Σημείωση2: Θα πρέπει αρχικά πριν το βρόχο while να δώσουμε μια τιμή στη μεταβλητή που ελέγχει τη συνθήκη του βρόχου, για να καθορίσουμε αν αυτός να εκτελεστεί ή όχι.

Μια συνήθης εφαρμογή του while βρόχου είναι να ελέγχει την εγκυρότητα των δεδομένων εισόδου από το χρήστη.

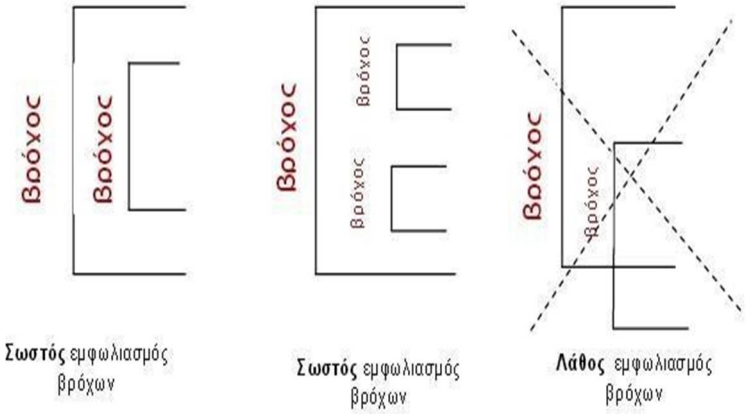
Παράδειγμα

Στο παρακάτω πρόγραμμα εξασφαλίζουμε ότι ο χρήστης θα εισάγει τελικά την επιθυμητή τιμή τοποθετώντας την κατάλληλη συνθήκη :

```
x = input ('Δώσε το βαθμό που πήρε ο μαθητής : ')
while x < 0 or x > 20
        x = input ('Ο βαθμός που έδωσε είναι εκτός των ορίων από 0 ως 20. Ξαναδώσε το βαθμό : ')
```

Κανόνες εμφωλευμένων επαναλήψεων

Μπορούμε να έχουμε και έναν βρόγχο μέσα σε έναν άλλο βρόγχο (**εμφωλευμένος βρόγχος**). Η δομή αυτή αποτελεί συνηθισμένη πρακτική σε αρκετά προγράμματα.
Πρώτα ξεκινάει η εξωτερική επανάληψη και μετά ξεκινάει και ολοκληρώνεται η εσωτερική και συνεχίζει πάλι η εξωτερική.
Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσοτέρων βρόχων που ο ένας είναι στο εσωτερικό του άλλου.



Σύγκριση της for με τη while

Στην εντολή for ο αριθμός των επαναλήψεων είναι προκαθορισμένος, ενώ στην εντολή while δεν είναι πάντα. Επίσης, στην εντολή while υπάρχουν περιπτώσεις που δεν μπορούμε να καταλάβουμε διαβάζοντας απλά τον κώδικα, αν θα έχουμε ή όχι ατέρμονα βρόχο κατά την εκτέλεση. Συνήθως η while μας δίνει πιο πολύπλοκο κώδικα.

Παρατήρηση :

Η σύνταξη	**ισοδυναμεί με**
for μεταβλητή **in range** (αρχή, μέχρι, βήμα): <εντολές>	i = αρχή **while** i < μέχρι : <εντολές> i = i + βήμα

4.2 Συναρτήσεις

Η έννοια των συναρτήσεων αποτελεί ένα από τα πιο σημαντικά δομικά στοιχεία ενός προγράμματος σε όλες τις γλώσσες προγραμματισμού. Οι συναρτήσεις μπορεί να είναι είτε έτοιμες από τη γλώσσα προγραμματισμού, είτε να δημιουργούνται από εμάς.
Η γλώσσα python μας δίνει τη δυνατότητα να γίνουμε και εμείς προγραμματιστές και να δημιουργήσουμε τις δικές μας εντολές, ώστε να μπορούμε να εκτελούμε μία σειρά ενεργειών, με τη χρήση μιας μόνο εντολής (της νέας εντολής που δημιουργήσαμε).
Οι **συναρτήσεις** στον προγραμματισμό είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Μια συνάρτηση μας επιτρέπει να δίνουμε ένα όνομα σε μια ομάδα εντολών και να την εκτελούμε χρησιμοποιώντας το όνομά της οπουδήποτε στο πρόγραμμά μας και για όσες φορές θέλουμε. Επίσης, μια συνάρτηση μπορεί να δεχθεί είσοδο και να παραγάγει έξοδο (επιστρεφόμενη τιμή). Στην περίπτωση που μέσα σε ένα πρόγραμμα χρειάζεται να επαναλάβουμε αρκετές φορές ένα μέρος του κώδικα, καταφεύγουμε στη χρήση των συναρτήσεων.
Δηλώνουμε στο πρόγραμμα μας ξεχωριστά την ύπαρξη της συνάρτησης καθώς και τις παραμέτρους που αυτή δέχεται, και στο μέρος αυτό γράφουμε, μόνο μία φορά, το μέρος του κώδικα που επιθυμούμε να επαναλαμβάνεται στο πρόγραμμα μου. Όταν θέλουμε να χρησιμοποιήσουμε κάπου στο πρόγραμμα μας το κομμάτι του κώδικα που απομονώσαμε, αρκεί να κάνουμε κλήση της συνάρτησης μας με τα κατάλληλα κάθε φορά ορίσματα (τιμές).
Μια συνάρτηση είναι ένα σύνολο εντολών που εκτελούν μια εργασία. Με τον τρόπο αυτό μπορούμε και να δημιουργήσουμε νέες δικές μας εντολές. Γράφοντας τις εντολές ομαδοποιημένες μέσα σε μια συνάρτηση μπορούμε να τις καλούμε απλά με το όνομά τους στο πρόγραμμα μας όποτε τις χρειαζόμαστε, χωρίς να ξαναγράφουμε όλες τις εντολές από την αρχή.
Τις συναρτήσεις μπορούμε και να τις χρησιμοποιήσουμε και για λόγους εύκολης κατανόησης του προγράμματος μας. Μπορούμε να σπάσουμε τον κώδικα σε επιμέρους κομμάτια και να δημιουργήσουμε από μία συνάρτηση για κάθε ένα από αυτά, και ας μην επαναλαμβάνονται κάπου αλλού μέσα στο πρόγραμμα μας.

4.2.1 Δημιουργώντας τις δικές μας συναρτήσεις

Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Μας επιτρέπουν να δίνουμε ένα όνομα σε ένα σύνολο εντολών και να το εκτελούμε καλώντας το όνομά τους, οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε. Αυτή η διαδικασία ονομάζεται κλήση (call) της συνάρτησης. Οι συναρτήσεις μπορούν να αποθηκευτούν και σε αρχεία για μελλοντική χρήση. Γενικότερα, χρήσιμες ομάδες εντολών «αποθηκεύονται» σε κατάλληλες συναρτήσεις. Η Python μάς προσφέρει άριστη υποστήριξη στις συναρτήσεις που γράφουμε και επιπλέον μάς παρέχει και πολλές έτοιμες (ενσωματωμένες συναρτήσεις).

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

4.2.2 Οι εσοχές

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

def print2( ) :
  print ******
  print ******
	def print2( ) :
	  print ******
	print ******

Η κλήση της 1ης θα τυπώσει δύο σειρές από αστεράκια ενώ της 2ης μία μόνο σειρά

4.2.3 Ορισμός Συνάρτησης

Ο ορισμός συναρτήσεων στην Python είναι αρκετά απλός:

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

Επεξηγήσεις

- Οι αγκύλες [ ] σημαίνουν πως ότι περικλείεται μέσα σε αυτές είναι προαιρετικό.
- Η λίστα των παραμέτρων μπορεί να είναι κενή.
- Μία συνάρτηση δεν είναι υποχρεωτικό να επιστρέφει κάποια τιμή.
- Το όνομα συνάρτησης ακολουθεί τους κανόνες των ονομάτων μεταβλητών της Python

ΠΡΟΣΟΧΗ : Στο τέλος της εντολής def τοποθετούμε άνω και κάτω τελεία ‘ : ‘.

Για να ορίσουμε μια δική μας συνάρτηση

1. χρησιμοποιούμε τη χαρακτηριστική λέξη def, (από την λέξη define=ορίζω)
2. ακολουθεί ένα όνομα που ταυτοποιεί την εκάστοτε συνάρτηση
3. και ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν ονόματα μεταβλητών,
4. ενώ η γραμμή τελειώνει με άνω και κάτω τελεία ‘ : ‘.
5. Ακολουθούν οι εντολές που θέλουμε να εκτελεστούν
6. Αν θέλουμε η συνάρτηση να μας επιστρέφει κάποια τιμή χρησιμοποιούμε τη λέξη return.

Η πρώτη γραμμή του ορισμού μιας συνάρτησης είναι η επικεφαλίδα της συνάρτησης. Η επικεφαλίδα ξεκινάει με τη δεσμευμένη λέξη def ακολουθούμενη από το όνομα της συνάρτησης, ένα ζευγάρι παρενθέσεων που προαιρετικά περικλείει μια διαχωριζόμενη με κόμματα λίστα παραμέτρων (parameters) και τελειώνει με μια άνω κάτω τελεία. Κάτω από την επικεφαλίδα ακολουθεί το σώμα της συνάρτησης (οι εντολές της συνάρτησης) σε εσοχή. Δεν υπάρχει κάποια ειδική εντολή που να υποδηλώνει το τέλος του μπλοκ εντολών της συνάρτησης. Όλα εξαρτώνται από τη στοίχιση των εντολών, **άρα πρέπει να είμαστε ιδιαίτερα προσεκτικοί με τη στοίχιση των εντολών** και την εσοχή πριν από κάθε εντολή, ώστε να εξασφαλίσουμε πως ανήκει στο σωστό μπλοκ εντολών.

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

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

Οι συναρτήσεις ορίζονται συνήθως στην αρχή των προγραμμάτων. Είναι προφανές όμως ότι μια συνάρτηση πρέπει να οριστεί πρώτα πριν χρησιμοποιηθεί (κληθεί).
Κατά την κλήση μιας συνάρτησης εκτελούνται οι εντολές που περιέχονται στο σώμα της.
Μπορούμε να καλούμε μία συνάρτηση όσες φορές απαιτούνται για τη λύση του προβλήματος που αντιμετωπίζει το πρόγραμμά μας. Μάλιστα κάτι τέτοιο είναι πολύ συνηθισμένο και χρήσιμο.

4.2.4 Παράμετροι συναρτήσεων

Μια συνάρτηση δέχεται δεδομένα μέσω των **παραμέτρων (parameters)** και επιστρέφει τα αποτελέσματα μέσω άλλων ή και των ίδιων παραμέτρων, στο πρόγραμμα ή σε μία άλλη συνάρτηση.
Οι παράμετροι καθορίζονται μέσα στο ζευγάρι των παρενθέσεων στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα. Όταν καλούμε τη συνάρτηση, δίνουμε και τις τιμές με τον ίδιο τρόπο, οι οποίες τιμές ονομάζονται **ορίσματα (arguments)**.
Κάποιες από τις ενσωματωμένες συναρτήσεις, που έχουμε ήδη συναντήσει, δεν απαιτούν ορίσματα, όπως για παράδειγμα, όταν καλούμε την math.pi, όπου δεν έχουμε κάποιο όρισμα. Σε άλλες όμως συναρτήσεις απαιτούνται ένα ή και περισσότερα ορίσματα, όπως στην math.pow όπου απαιτούνται δύο, ένα για τη βάση και ένα για τον εκθέτη.

Παραδείγματα

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

print ginomeno(5, 10)          # θα μας εμφανίσει το αποτέλεσμα 50
```

Σημείωση: Η μεταβίβαση παραμέτρων **κατά τιμή (call by value**), στην Python λειτουργεί με τέτοιο τρόπο, ώστε οποιαδήποτε αλλαγή στις τιμές των παραμέτρων εντός της συνάρτησης δεν έχει καμία επίδραση στα ορίσματα-μεταβλητές που έχουν οριστεί εκτός της συνάρτησης, όπως φαίνεται παρακάτω:

```
def increment(a, b):
   a = a + 1
   b = b + 1
   return a+b                    # τέλος συνάρτησης

z = 1
w = 2
q = increment( z, w )
print z, w, q                    #  θα εμφανιστούν στην οθόνη οι τιμές 1, 2, 5
```

Τα ορίσματα z, w δεν αλλάζουν τιμή, παρόλο που οι αντίστοιχες παράμετροι αυξάνονται εντός της συνάρτησης increment, όπως φαίνεται και από την τιμή που επιστρέφει η συνάρτηση και καταχωρείται στην q.

Η μεταβίβαση παραμέτρων στην Python γίνεται **κατά τιμή (call-by-value),** δηλαδή δημιουργούνται αντίγραφα των ορισμάτων, οπότε δεν γίνεται καμία αλλαγή στα ορίσματα-μεταβλητές που έχουν οριστεί εκτός της συνάρτησης.

- Οι μεταβλητές a, b δεν υφίστανται παρά μόνο όταν κληθεί η συνάρτηση increment,
- Οι τιμές των μεταβλητών z και w αντιγράφονται στις αντίστοιχες a και b όταν εκτελείται η συνάρτηση,
- Οι μεταβλητές a και b καταστρέφονται, ΧΩΡΙΣ να επιστρέψουν πίσω στις αντίστοιχες z και w τίποτα.

Μία συνάρτηση μπορεί να επιστρέφει και τιμή, η οποία μπορεί να εκχωρηθεί σε μία μεταβλητή ή ακόμη και να χρησιμοποιηθεί ως μέρος μιας έκφρασης. Η επιστροφή τιμής γίνεται με την εντολή return η οποία ως εξής :
Η εντολή return επιστρέφει αμέσως τον έλεγχο ροής από τη συνάρτηση στο σημείο του προγράμματος που αυτή κλήθηκε και χρησιμοποιεί την τιμή της έκφρασης που ακολουθεί ως επιστρεφόμενη τιμή.

Το όνομα της συνάρτησης, πρέπει να υπακούει σε κάποιους κανόνες;
Ακριβώς τους ίδιους με τα ονόματα των μεταβλητών και των σταθερών.

Τι είναι οι παράμετροι μίας συνάρτησης;
Οι συναρτήσεις τις περισσότερες φορές χρειάζονται δεδομένα πάνω στα οποία θα ενεργήσουν. Τα δεδομένα αυτά διοχετεύονται μέσα στην συνάρτηση μέσω μεταβλητών που τοποθετούνται μέσα στις παρενθέσεις της συνάρτησης

Που συνήθως γράφονται οι συναρτήσεις;
Οι συναρτήσεις συνήθως γράφονται στην αρχή του προγράμματος και σε κάθε περίπτωση πριν από το σημείο που θα κληθούν (χρησιμοποιηθούν).

Πόσες φορές μπορεί να κληθεί μία συνάρτηση;
Δεν υπάρχει κανένας απολύτως περιορισμός.

Κάθε φορά που καλώ μία συνάρτηση, πρέπει να χρησιμοποιώ τα ίδια ορίσματα;
Όχι φυσικά.

Συναρτήσεις - Σπουδαιότητα

- Μας επιτρέπουν να δώσουμε όνομα σε ένα σύνολο εντολών και να το επαναχρησιμοποιούμε εύκολα και γρήγορα, απλά αναφέροντας το όνομα του. Η χρησιμοποίηση μιας συνάρτησης λέγεται κλήση της συνάρτησης.
- Ο ορισμός μίας νέας συνάρτησης κάνει το πρόγραμμα μικρότερο, απομακρύνοντας τμήματα κώδικα που επαναλαμβάνονται.
- Οι τυχόν αλλαγές– διορθώσεις γίνονται σε ένα μόνο μέρος του προγράμματος.
- Γενικά διευκολύνουν στη συγγραφή, ανάγνωση, κατανόηση και διόρθωση του προγράμματος.

Ειδικά στην Python:

- Οι συναρτήσεις μπορούν να αποθηκευτούν και σε ξεχωριστά αρχεία, εμπλουτίζοντας το ρεπερτόριο εντολών της γλώσσας.
- Υπάρχει άριστη υποστήριξη και ευελιξία.
- Υπάρχει μεγάλη γκάμα έτοιμων συναρτήσεων για ότι σχεδόν επιθυμούμε.