ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Γλώσσες Προγραμματισμού

Εισαγωγή στον Προγραμματισμό - Python

Χαρά Πρασσά, chprassa@gmail.com

- *Ένας κώδικας στην Python στην απλούστερη μορφή του είναι μια ακολουθία εντολών οι οποίες εκτελούνται η μια μετά την άλλη, με την σειρά που έχουν πληκτρολογηθεί. Τις περισσότερες φορές όμως, σε πιο σύνθετα προγράμματα, διαφορετικές μεταβλητές εισόδου απαιτούν την υπό συνθήκη εκτέλεση κάποιων τμημάτων εντολών του προγράμματος, ή μπορεί να απαιτείται η επανάληψη μιας ακολουθίας εντολών.*

- *Σχεσιακοί τελεστές*: Συγκρίσεις όπως εάν ένας αριθμός είναι μεγαλύτερος από έναν άλλο ή ένας χαρακτήρας είναι διαφορετικός από κάποιον άλλο. Αν το αποτέλεσμα της σύγκρισης είναι αληθές η Python επιστρέφει το λογικό **True**, σε αντίθετη περίπτωση το λογικό **False**.

Σχεσιακός Τελεστής	Περιγραφή
<	μικρότερο
>	μεγαλύτερο
<=	μικρότερο ή ίσο
>=	μεγαλύτερο ή ίσο
==	ίσο
!=	όχι ίσο

Όταν ελέγχουμε την αλγεβρική ισότητα δύο αριθμών χρησιμοποιούμε το διπλό σύμβολο (==), ενώ το μονό σύμβολο (=), όπως το χρησιμοποιούμε στις μεταβλητές, σημαίνει ανάθεση τιμής σε μια μεταβλητή.

- *Για παράδειγμα:*

```
age=16
can_create_account = age > 18
print(can_create_account)
```

Ελέγχει αν η μεταβλητή age >18 και αποδίδει το λογικό αποτέλεσμα (True ή False) στη μεταβλητή can_create_account

Output

False

```
age=19
can_create_account = age > 18
print(can_create_account)
```

Output

True

- *Λογικοί τελεστές*: Οι λογικοί τελεστές πραγματοποιούν λογικούς υπολογισμούς και επιτρέπουν τη δημιουργία σύνθετων εκφράσεων συνδυάζοντας δύο ή και περισσότερες εκφράσεις σύγκρισης. Το αποτέλεσμα τους είναι το λογικό **True** ή το λογικό **False**.

Λογικός Τελεστής	Περιγραφή
and	Αν τα ορίσματα δεξιά και αριστερά του τελεστή είναι αληθή ο τελεστής επιστρέφει True
or	Αν ένα από τα ορίσματα ή όλα τα ορίσματα δεξιά ή αριστερά του τελεστή είναι αληθή ο τελεστής επιστρέφει True
not	Αν η συνθήκη δεν ισχύει ο τελεστής επιστρέφει True και το αντίστροφο

Για παράδειγμα:

```
>>> a=1
>>> b=1
>>> a==1 and b==1
True
>>> a==1 or b==1
True
>>> not a==1
False
>>> not a==1 and b==3 or a+b==2
True
>>> not a==1 and (b==3 or a+b==2)
False
```

Η σειρά εκτέλεσης των λογικών τελεστών είναι ότι πρώτα εκτελείται η not, μετά η and και τελευταία η or εκτός και αν υπάρχουν παρενθέσεις

- *Δομές ελέγχου:* Εντολές που επιτρέπουν την επιλεκτική εκτέλεση ορισμένων εντολών μέσα στο πρόγραμμα ανάλογα με τις συνθήκες που ικανοποιούνται. Εάν η συνθήκη που περιλαμβάνεται σε μια δομή ελέγχου είναι αληθής, τότε εκτελείται η εντολή ή το σύνολο των εντολών που ακολουθούν τη συνθήκη ενώ, εάν η συνθήκη είναι ψευδής, ο υπολογιστής κατά την εκτέλεση του προγράμματος παραβλέπει την εντολή ή το σύνολο των εντολών που ακολουθούν τη συνθήκη και συνεχίζει με τις υπόλοιπες εντολές.

Η πιο απλή δομή ελέγχου είναι η δομή **if** . Η γενική μορφή είναι:

if συνθήκη**:**
εντολή ή σύνολο εντολών

- *Κάθε **if** statement ξεκινάει με τη λέξη κλειδί **if** και αμέσως μετά ακολουθεί μια συνθήκη. Εάν η συνθήκη είναι **True** τότε θα εκτελεστεί ο κώδικας που ανήκει στο **if,** ειδάλλως εάν η συνθήκη είναι **False** τότε θα αγνοηθεί αυτό το μέρος του κώδικα.*
- Σημαντικό ρόλο παίζει το **κενό κάτω από την if συνθήκη**. Συνήθως το κενό το δημιουργούμε πατώντας το tab στο πληκτρολόγιο μας έτσι ώστε αν γράψουμε πολλαπλές γραμμές κώδικα κάτω από την if να είναι όλες στοιχισμένες για να τις ξεχωρίζει εύκολα ο compiler της Python. Εάν δεν αφήσετε το κενό, τότε ο compiler της Python θα εμφανίσει μήνυμα λάθους ότι δεν αναγνωρίζει την εντολή και το πρόγραμμα μας δεν θα εκτελεστεί. Τέλος, η if συνθήκη ολοκληρώνεται με την προσθήκη των colons (**:**) αμέσως μετά την συνθήκη.
- Τα IDLE, Visual Studio Code και Jupyter όταν γράφετε μια if συνθήκη μεταφέρουν αυτόματα τον κέρσορα στην επόμενη γραμμή στο σωστό διάστημα.

- ***Παράδειγμα:*** Το αρχείο sales.py υπολογίζει το συνολικό κόστος μιας παραγγελίας βάσει των μονάδων πώλησης και της τιμής πώλησης. Εάν η παραγγελία αναφέρεται σε περισσότερες από 40 μονάδες τότε στην αρχική τιμή πώλησης παρέχεται έκπτωση 15%.

sales.py

C: > Users > chpra > OneDrive > Desktop > sales.py > ...

```
1   #Παράδειγμα υπολογισμού συνολικού κόστους παραγγελίας βάσει των μονάδων
2   #πώλησης και της τιμής πώλησης
3   q=int(input('Πληκτρολογείστε τις μονάδες πώλησης: '))
4   p=float(input('Πληκτρολογείστε την τιμή πώλησης: '))
5   c=q*p
6   if q>40:
7       c=c*0.85
8   print('Το κόστος της παραγγελίας είναι: ',format(c,',.2f'),'\u20AC')
```

Output

```
Πληκτρολογήστε τις μονάδες πώλησης: 45
Πληκτρολογήστε την τιμή πώλησης: 64.5
Το κόστος της παραγγελίας είναι: 2,467.12
```

- Η δομή ελέγχου **if-else** χρησιμοποιείται στην περίπτωση που υπάρχουν δύο αμοιβαία αποκλειόμενες περιπτώσεις που καθορίζονται από την ισχύ μιας συνθήκης. Η γενική μορφή είναι:

if συνθήκη**:**

    1° σύνολο εντολών

else:

    2° σύνολο εντολών

Παράδειγμα: Το αρχείο average_grade.py διαβάζει από την οθόνη το ονοματεπώνυμο ενός φοιτητή και τους βαθμούς που έλαβε σε τρείς προόδους. Στη συνέχεια:

- Υπολογίζει τον μέσο όρο
- Εμφανίζει το όνομα του φοιτητή και μήνυμα ‘Επιτυχών’ αν ο μέσος όρος είναι μεγαλύτερος ή ίσος του 5, αλλιώς εμφανίζει ‘Αποτυχών’.

average_grade.py

C: > Users > chpra > OneDrive > Desktop > average_grade.py > ...

```
1   #Παράδειγμα average_grade (διαβάζει το ονοματεπώνυμο φοιτητή και βαθμούς που έλαβε σε 3 προόδους)
2   print('Πρόγραμμα υπολογισμού μέσου όρου 3 βαθμών')
3   onoma=input('Ονοματεπώνυμο: ')
4   grade1= float(input('1η βαθμολογία: '))
5   grade2= float(input('2η βαθμολογία: '))
6   grade3= float(input('3η βαθμολογία: '))
7
8   average_grade=(grade1+grade2+grade3)/3
9   if average_grade>=5:
10      print(onoma, format(average_grade,'.2f'),'Επιτυχών')
11  else:
12      print(onoma, format(average_grade,'.2f'),'Αποτυχών')
```

Output

Πρόγραμμα υπολογισμού μέσου όρου 3 βαθμών
Ονοματεπώνυμο: Μάριος Παπαδόπουλος
1η βαθμολογία: 6.5
2η βαθμολογία: 5
3η βαθμολογία: 4.5
Μάριος Παπαδόπουλος 5.33 Επιτυχών

- Μπορούμε να έχουμε και περισσότερες από δύο επιλογές απλά προσθέτοντας πολλαπλά **if-else** statements. Μάλιστα η Python μας επιτρέπει να γράφουμε αυτό τον συνδυασμό σαν **elif** για εξοικονόμηση κώδικα.
- Η δομή ελέγχου **if-elif** χρησιμοποιείται στην περίπτωση που υπάρχουν περισσότερες από δύο αμοιβαία αποκλειόμενες περιπτώσεις που καθορίζονται από την ισχύ μιας συνθήκης. Η γενική μορφή είναι:

if συνθήκη 1η**:**
    1° σύνολο εντολών
elif συνθήκη 2η**:**
    2° σύνολο εντολών
…….
elif συνθήκη nη**:**
    n° σύνολο εντολών
else:
    n°+1 σύνολο εντολών

Παράδειγμα: Το αρχείο **taxpay.py** υπολογίζει το ποσό του φόρου βάσει της ακόλουθης κλίμακας φορολόγησης: 15% επί των πρώτων 10.000 ευρώ, 25% από 10.000-40.000 ευρώ και 40% για ποσά μεγαλύτερα από 40.000 ευρώ.

taxpay.py

C: > Users > chpra > OneDrive > Desktop > taxpay.py > ...

```
1   #Παράδειγμα υπολογισμού φόρου με κλιμακωτή φορολόγηση
2   print('Πρόγραμμα υπολογισμού φόρου με βάση τα εισοδήματα')
3   income=float(input('Εισάγετε το φορολογητέο εισόδημα: '))
4   if income<=10000:
5       tax_pay=0.15*income
6   elif income<=40000:
7       tax_pay=0.15*10000+0.25*(income-10000)
8   else:
9       tax_pay=0.15*10000+0.25*30000+0.4*(income-40000)
10  print('Το εισόδημα είναι \u20AC', format(income,',.2f'), 'και ο φόρος είναι \u20AC', format(tax_pay,',.2f'))
```

Output

Πρόγραμμα υπολογισμού φόρου με βάση τα εισοδήματα
Εισάγετε το φορολογητέο εισόδημα: 45115.50
Το εισόδημα είναι € 45,115.50 και ο φόρος είναι € 11,046.20

- Το παραπάνω παράδειγμα μπορεί να γραφεί ισοδύναμα με εμφωλευμένα **if** ως εξής:

taxpay_nested_if ×

C: > Users > chpra > OneDrive > Desktop > taxpay_nested_if > ...

```
1   #Ισοδύναμος τρόπος γραφής του άνω παραδείγματος με εμφωλευμένα if
2   #Παράδειγμα υπολογισμού φόρου με κλιμακωτή φορολόγηση
3   print('Πρόγραμμα υπολογισμού φόρου με βάση τα εισοδήματα')
4   income=float(input('Εισάγετε το φορολογητέο εισόδημα: '))
5   if income<=10000:
6       tax_pay=0.15*income
7   else:
8       if income<=40000:
9           tax_pay=0.15*10000+0.25*(income-10000)
10      else:
11          tax_pay=0.15*10000+0.25*30000+0.4*(income-40000)
12  print('Το εισόδημα είναι \u20AC', format(income,',.2f'), 'και ο φόρος είναι \u20AC', format(tax_pay,',.2f'))
```

Παράδειγμα για εξάσκηση 1: Να γραφεί πρόγραμμα σε Python το οποίο να επιλύει την πρωτοβάθμια εξίσωση ax+b=0. Σύμφωνα με τα μαθηματικά για να λυθεί η εξίσωση εξετάζονται τα παρακάτω:

- Αν a≠0 τότε x=-b/a
- Αν a=0 και b=0 τότε αόριστη εξίσωση
- Αν a=0 και b≠0 τότε αδύνατη εξίσωση

- 1ος τρόπος (*χρήση της if*)

exiswsi_1.py ●

C: > PythonCodes > exiswsi_1.py > ...

```
1   print("Επίλυση εξίσωσης ax+b=0")
2   a=float(input("Εισάγετε το a="))# ανάγνωση του a και μετατροπή σε πραγματικό αριθμό
3   b=float(input("Εισάγετε το b="))# ανάγνωση του b και μετατροπή σε πραγματικό αριθμό
4   if a!=0:
5       x=-b/a
6       print("x=",x)
7   if a==0 and b==0:
8      print("Αόριστη εξίσωση")
9   if a==0 and b!=0:
10      print("Αδύνατη εξίσωση")
```

Output

Επίλυση εξίσωσης ax+b=0
Εισάγετε το a=5
Εισάγετε το b=2.5
x= -0.5

- 2ος τρόπος (*χρήση της elif*)

exiswsi_2.py ●

C: > PythonCodes > exiswsi_2.py > ...

```
1   print("Επίλυση εξίσωσης ax+b=0")
2   a=float(input("Εισάγετε το a="))# ανάγνωση του a και μετατροπή σε πραγματικό αριθμό
3   b=float(input("Εισάγετε το b="))# ανάγνωση του b και μετατροπή σε πραγματικό αριθμό
4   if a!=0:
5       x=-b/a
6       print("x=",x)
7   elif b==0:
8     print("Αόριστη εξίσωση")
9   else:
10      print("Αδύνατη εξίσωση")
```

Output

Επίλυση εξίσωσης ax+b=0
Εισάγετε το a=0
Εισάγετε το b=0
Αόριστη εξίσωση

- 3ος τρόπος (*χρήση εμφωλευμένων if*)

exiswsi_3.py

C: > PythonCodes > exiswsi_3.py > ...

```
1   print("Επίλυση εξίσωσης ax+b=0")
2   a=float(input("Εισάγετε το a="))# ανάγνωση του a και μετατροπή σε πραγματικό αριθμό
3   b=float(input("Εισάγετε το b="))# ανάγνωση του b και μετατροπή σε πραγματικό αριθμό
4   if a!=0:
5       x=-b/a
6       print("x=",x)
7   else:
8       if b==0:
9           print("Αόριστη εξίσωση")
10      else:
11          print("Αδύνατη εξίσωση")
```

Output

Επίλυση εξίσωσης ax+b=0
Εισάγετε το a=0
Εισάγετε το b=2
Αδύνατη εξίσωση

Παράδειγμα για εξάσκηση 2: Να γραφεί πρόγραμμα σε Python το οποίο να εμφανίζει σε μήνυμα οθόνης την έγκριση ή όχι ενός καταναλωτικού δανείου σε πιθανό δανειολήπτη. Για να εγκριθεί το δάνειο θα πρέπει είτε ο δανειολήπτης να έχει ετήσιο εισόδημα μεγαλύτερο από 30000 ευρώ ή να έχει προϋπηρεσία πάνω από 5 έτη.

loan.py

C: > Users > chpra > OneDrive > Desktop > loan.py > ...

```
1   #Παράδειγμα για εξάσκηση 2
2   print('Αλγόριθμος λήψης δανείου')
3   min_income,min_years=30000,5
4   income=float(input('Εισάγετε το ετήσιο εισόδημα: '))
5   years_of_work_experience=int(input('Εισάγετε τα έτη προϋπηρεσίας: '))
6   if income >= min_income or years_of_work_experience>= min_years:
7       print('Έγκριση δανείου')
8   else:
9       print('Απόρριψη δανείου')
```

Output

Αλγόριθμος για τη λήψη δανείου
Εισάγετε το ετήσιο εισόδημα: 22000
Εισάγετε τα έτη προϋπηρεσίας: 6
Έγκριση δανείου

Γενική έκφραση:

value_if_true if [Συνθήκη] else value_if_false

Παραδείγματα:

```
value=int(input("Give me an integer: "))
is_even=True if value % 2 ==0 else False
print(f"Is even: {is_even}")
```

```
value=int(input("Give me an integer: "))
print("Zero") if value==0 else print("One Digit Number") if value<10 else print("Multiple Digits Number")
```

- Η Python δίνει τη δυνατότητα επανάληψης μιας εντολής ή μιας ομάδας εντολών. Οι πιο συχνά χρησιμοποιούμενες δομές επανάληψης είναι ο βρόχος (loop) του **for** και ο βρόχος του **while**.
- Η εντολή **for** είναι μια απλή δομή επανάληψης που επαναλαμβάνει ένα block εντολών ένα συγκεκριμένο πλήθος επαναλήψεων. Η εντολή **for..in..** είναι μία εντολή βρόχου, η οποία είναι σχεδιασμένη να λειτουργεί τόσες φορές όσες προκύπτει από μία αλληλουχία δεδομένων.

Η σύνταξη της δομής επανάληψης for… είναι της μορφής:

for μεταβλητή **in** αλληλουχία_δεδομένων**:**

1° σύνολο εντολών

2° σύνολο εντολών

……..

n° σύνολο εντολών

Η **αλληλουχία_δεδομένων** είναι ένα σύνολο διακριτών δεδομένων που προκύπτουν από:

- Μία συνάρτηση που ορίζει ένα κλειστό διάστημα ακεραίων αριθμών.
- Τους χαρακτήρες ενός string, τα στοιχεία μίας λίστας (list), μίας πλειάδας (tuple), ενός λεξικού (dictionary), ενός συνόλου(set), ή μιας πιο πολύπλοκης δομής.

Η **μεταβλητή** είναι το όνομα μιας μεταβλητής η οποία θα πάρει τιμές μέσα από αυτό που ονομάσθηκε αλληλουχία_δεδομένων.

Παράδειγμα:

```
for letter in "Python":
    print(letter)
```

terminal output

```
P
y
t
h
o
n
```

- Η Python με τη συνάρτηση **range** παρέχει στην εντολή for παραμέτρους για την εκτέλεση ενός block επανάληψης. Η σύνταξη και οι λειτουργίες της **range** είναι:

  - range(n) όπου n ακέραιος αριθμός

Λειτουργία: Δημιουργεί την αλληλουχία διαδοχικών αριθμών 0,1,2,3,4,…,n-1

  - range(from, to) όπου from, to ακέραιοι αριθμοί

Λειτουργία: Δημιουργεί την αλληλουχία διαδοχικών αριθμών στο διάστημα [from, to-1]

  - range(from, to, step) όπου from , to, step ακέραιοι αριθμοί

Λειτουργία: Δημιουργεί την αλληλουχία αριθμών στο διάστημα [from, to-step] ως εξής: from, from+step, from+step+step,……,to-step]

Παράδειγμα: Να υπολογιστεί το άθροισμα των ακεραίων αριθμών από το 1 μέχρι το 10.

```
sum=0
for k in range (1,11):
    sum+=k
print('Το άθροισμα είναι: ',sum)
```

ή

```
for k in range (11):
```

terminal output

Το άθροισμα είναι: 55

- Ποιο θα ήταν το αποτέλεσμα στο τερματικό αν αλλάζαμε τη στοίχιση της print();

```
sum=0
for k in range (1,11):
    sum+=k
    print('Το άθροισμα είναι: ',sum)
```

Παράδειγμα: Να υπολογιστεί το άθροισμα των ακεραίων αριθμών από το 5 μέχρι το 50 με βήμα 5.

```
sum=0
for k in range (5,55,5):
    sum+=k
print('Το άθροισμα είναι: ',sum)
```

terminal output

Το άθροισμα είναι: 275

Παράδειγμα:

```
for number in range(10, 0, -1):
    print('Ο αριθμός είναι: ',number)
```

terminal output

Ο αριθμός είναι: 10
Ο αριθμός είναι: 9
Ο αριθμός είναι: 8
Ο αριθμός είναι: 7
Ο αριθμός είναι: 6
Ο αριθμός είναι: 5
Ο αριθμός είναι: 4
Ο αριθμός είναι: 3
Ο αριθμός είναι: 2
Ο αριθμός είναι: 1

Παράδειγμα: Επανάληψη όπου η αλληλουχία προκύπτει από τα στοιχεία μίας λίστας.

```
names=['Νίκος', 'Γιώργος', 'Μαρία', 'Ελένη']
for name in names:
    print(name)
```

Στο παραπάνω παράδειγμα η μεταβλητή name θα λάβει διαδοχικά τις τιμές:
'Νίκος', 'Γιώργος', 'Μαρία', 'Ελένη'

Εντολή continue: παραβλέπει την τρέχουσα τιμή της δομής επανάληψης βασιζόμενη σε κάποια συνθήκη που έχουμε καθορίσει.

```
for value in [1,4,5,7,9]:
    if value==5:
        continue
    print("The value is: ",value)
```

terminal output

```
The value is:  1
The value is:  4
The value is:  7
The value is:  9
```

Η τιμή 5 παραβλέπεται και η δομή επανάληψης συνεχίζει με την αμέσως επόμενη διαθέσιμη τιμή.

Εντολή break: σταματάει την εκτέλεση της δομής επανάληψης και αναγκάζει την Python να συνεχίσει με τον υπόλοιπο κώδικα του προγράμματος εκτός της for.

```
for value in [1,4,5,7,9]:
    if value==5:
        break
    print("The value is: ",value)
```

terminal output

```
The value is:  1
The value is:  4
```

Παράδειγμα εξάσκησης: Να γραφεί πρόγραμμα το οποίο να διαβάζει από την οθόνη 5 αριθμούς και στο τέλος να εμφανίζει τον μεγαλύτερο.

```
for i in range(1,6):
    print(i,"ος αριθμός",end ='')
    number=float(input(":"))
    if i==1:
        max_num=number
    else:
        if number>max_num:
            max_num=number
print("Ο μεγαλύτερος αριθμός είναι: ", max_num)
```

Παράδειγμα εξάσκησης: Να γραφεί πρόγραμμα το οποίο διαβάζει 10 αριθμούς και υπολογίζει το άθροισμα των αριθμών αυτών. Μετά το τέλος εισαγωγής των στοιχείων το πρόγραμμα εμφανίζει στην οθόνη:

- το άθροισμα των αριθμών αυτών
- τον μεγαλύτερο από τους αριθμούς που δόθηκαν
- τον μικρότερο από τους αριθμούς που δόθηκαν

```
sum_num=0
for i in range(1,11):
    print(i,"ος αριθμός",end ='')
    number=float(input(":"))
    sum_num+=number #εναλλακτικά, sum_num=sum_num+number
    if i==1:
        max_num=number
        min_num=number
    else:
        if number>max_num:
           max_num=number
        elif number<min_num:
           min_num=number
print("Το άθροισμα των αριθμών είναι: ", sum_num)
print("Ο μεγαλύτερος είναι: ", max_num)
print("Ο μικρότερος είναι: ", min_num)
```

- Υπάρχουν κάποιες φορές που όταν εκτελούμε μια εντολή for χρειάζεται να ξέρουμε σε ποια επανάληψη βρισκόμαστε. Για να μπορέσουμε να πάρουμε αυτή την πληροφορία θα πρέπει να συνδυάσουμε την εντολή for με την enumerate.

Παράδειγμα:

```
word='Python'
for index, character in enumerate(word):
    print(index, character)
```

```
0 P
1 y
2 t
3 h
4 o
5 n
```

Μετράει την επανάληψη ξεκινώντας από το 0

```
word='Python'
for index, character in enumerate(word, start=1):
    print(index,character)
```

Μετράει την επανάληψη ξεκινώντας από το 1

- Ο βρόχος **while** χρησιμοποιείται για την εκτέλεση μιας εντολής ή μιας ομάδας εντολών όταν δεν είναι προκαθορισμένο το πλήθος των επαναλήψεων αλλά εξαρτάται από την ισχύ μιας συγκεκριμένης συνθήκης. Η γενική μορφή του βρόχου **while** είναι:

```
while λογική_πρόταση:
    1° σύνολο εντολών
    2° σύνολο εντολών
    ……..
    n° σύνολο εντολών
```

Σε αυτή τη μορφή επανάληψης, οι εντολές που βρίσκονται στο block επανάληψης θα εκτελεστούν για μη προκαθορισμένο αριθμό επαναλήψεων. Η επανάληψη τερματίζει με βάση την τιμή της συνθήκης.

Με το όρο λογική_πρόταση θεωρείται ότι μπορεί να είναι:
- Μία μεταβλητή που μπορεί να πάρει τιμές True ή False ή
- μία απλή συνθήκη με χρήση σχεσιακών τελεστών ή
- μία σύνθετη συνθήκη με χρήση λογικών τελεστών ή
- η σταθερά True.

Ο βρόχος while λειτουργεί με τον εξής τρόπο:

- Εάν η συνθήκη είναι **False** (ψευδής) τότε η Python αγνοεί όλες τις εντολές που βρίσκονται στο εσωτερικό του βρόχου **while** και συνεχίζει την επόμενη εντολή εκτός while η οποία πρέπει να βρίσκεται στοιχισμένη με το while.

- Εάν η λογική πρόταση είναι **True** (αληθής) τότε εκτελούνται οι εντολές που βρίσκονται στο εσωτερικό του βρόχου while (τουλάχιστον μια φορά). Όταν η συνθήκη είναι πάντα αληθής τότε οι εντολές του βρόχου επαναλαμβάνονται επ' άπειρον (ατέρμονας βρόχος). Προκειμένου να τερματιστεί η επανάληψη πρέπει, με κάποιο τρόπο, να αλλάξουν κάποιες τιμές εντός της επανάληψης οι οποίες θα επιδράσουν στην λογική πρόταση προκειμένου αυτή να γίνει False ή να χρησιμοποιηθεί η εντολή break.

Παράδειγμα εξάσκησης: Να γραφεί πρόγραμμα το οποίο να υπολογίζει και να εμφανίζει στην οθόνη το άθροισμα των περιττών αριθμών από 1 έως και 100.

```
k=1 # αρχική τιμή του μετρητή
sum=0 # μηδενισμό αθροιστή
while k<=100: # οσο ο μετρητής είναι <=100
    sum+=k # προσθέτει στον αθροιστή το περιεχόμενο του μετρητή
    k+=2 # αυξηση του μετρητή κατά 2
print(sum) # εκτύπωση του αθροίσματος
```

- Ο βρόχος while είναι ιδιαίτερα χρήσιμος στις περιπτώσεις ελέγχου ενός κριτηρίου σύγκλισης. Πολλές φορές μαζί με τη συνθήκη χρησιμοποιείται και ένας μετρητής για να αποφευχθεί η περίπτωση των άπειρων επαναλήψεων.

```
x=1.5
i=0 # μετρητής για τη μέτρηση του αριθμού των επαναλήψεων
while x>1 and i<100:
    x=x*2
    i=i+1
print(x,i)
```

- Περίπτωση ατέρμονα βρόγχου

```
value=int(input("Give me a number (zero to exit): "))
while value!=0:
    print(f"You gave the number: {value}")
```

Αν δώσετε μια τιμή διάφορη του μηδενός, ο βρόγχος θα είναι ατέρμονας καθώς πάντα η while είναι True.

- Επίλυση προβλήματος

```
value=int(input("Give me a number (zero to exit): "))
while value!=0:
    print(f"You gave the number: {value}")
    value=int(input("Give me a number (zero to exit): "))
```

```
Give me a number (zero to exit): 3
You gave the number: 3
Give me a number (zero to exit): 3
You gave the number: 3
Give me a number (zero to exit): 4
You gave the number: 4
Give me a number (zero to exit): 0
```

- Αν δώσετε ως τιμή το μηδέν, ο βρόγχος θα είναι ατέρμονας καθώς πάντα η while είναι True. Γιατί;

```
value=input("Give me a number (zero to exit): ")
while value!=0:
    print(f"You gave the number: {value}")
```

```
You gave the number: 0
You gave the number: 0
You gave the number: 0
You gave the number: 0
…
```

- Επίλυση προβλήματος

```
value=input("Give me a number (zero to exit): ")
while value!='0':
    print(f"You gave the number: {value}")
```

```
Give me a number (zero to exit): 0
```

- *Παράδειγμα εξάσκησης*: Να γραφεί πρόγραμμα το οποίο να δέχεται λέξεις από το χρήστη και να επιστρέφει το πλήθος των μικρών λέξεων (αριθμός χαρακτήρων <5) και το πλήθος των μεγάλων λέξεων (αριθμός χαρακτήρων >=5). Το πρόγραμμα σταματάει όταν ο χρήστης εισάγει τη λέξη 'stop'.

```
small_words, big_words=0, 0
while True:
    word=input("Give a word (type 'stop' to exit): ")
    if word=="stop":
        break
    if len(word)<5:
        small_words+=1 #small_words=small_words+1
    else:
        big_words +=1
print(f"Μικρές λέξεις: {small_words}\nΜεγάλες λέξεις: {big_words}")
```

Η while εκτελείται πάντα αλλά η χρήση της break αποκλείει την περίπτωση ατέρμονα βρόγχου

Η εκτέλεση της break προκαλεί την έξοδο από τη δομή επανάληψης (while)

```
Give a word (type 'stop' to exit): and
Give a word (type 'stop' to exit): for
Give a word (type 'stop' to exit): Python
Give a word (type 'stop' to exit): Student
Give a word (type 'stop' to exit): stop
Μικρές λέξεις: 2
Μεγάλες λέξεις: 2
```

Παράδειγμα εξάσκησης: Να γραφεί πρόγραμμα με το οποίο θα χρησιμοποιήσετε την εντολή while για να συγκρίνετε τον αριθμό που εισάγει ο χρήστης με έναν τυχαίο αριθμό που δημιουργείται από τη γεννήτρια τυχαίων αριθμών random.randint( ) function.

```
import random
random_number = random.randint(1,10)
guess = None
while guess!= random_number:
    guess = int(input('Εισάγετε έναν αριθμό από το 1 έως το 10: '))
    if guess < random_number:
        print("Επιλέξτε ένα μεγαλύτερο αριθμό")
    elif guess > random_number:
        print("Επιλέξτε ένα μικρότερο αριθμό")
    else:
        print("Ο αριθμός είναι σωστός!")
```

Η εντολή continue προκαλεί τερματισμό της τρέχουσας επανάληψης σε ένα βρόγχο και στέλνει τη ροή του προγράμματος ξανά στην επόμενη επανάληψη του βρόγχου.

Παράδειγμα εξάσκησης: Θέλουμε να καταγράψουμε από το σύνολο των αριθμών που πληκτρολογεί ο χρήστης πόσοι είναι μονοψήφιοι αριθμοί και το πρόγραμμα να τερματίζει όταν ο χρήστης πληκτρολογεί το μηδέν.

```
total_numbers=0
one_digit_numbers=0
while True:
    word=input("Give me a number (type 0 to exit) :")
    if word=="0":
        break
    total_numbers+=1
    if len(word)>1:
        continue
    one_digit_numbers+=1
print(f"Total numbers: {total_numbers}\n One digit numbers: {one_digit_numbers}")
```

```
Give me a number (type 0 to exit) :9
Give me a number (type 0 to exit) :888
Give me a number (type 0 to exit) :777
Give me a number (type 0 to exit) :4
Give me a number (type 0 to exit) :333
Give me a number (type 0 to exit) :0
```

Total numbers: 5

One digit numbers: 2