

# Δομές Επαναληπτικής διαδικασίας

## Επαναληπτικές Διαδικασίες – Βρόχοι - Loops

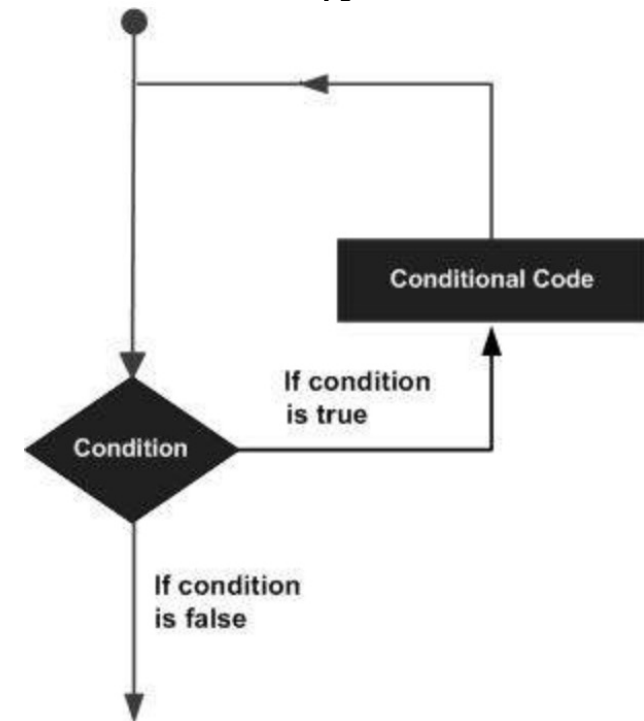
Στους περισσότερους υπολογισμούς θέλουμε να εκτελέσουμε μια σειρά από εντολές αρκετές φορές ώστε να καταλήξουμε στο αποτέλεσμα που θέλουμε.

Φανταστείτε ότι κάποιος μας ζητούσε να υπολογίσουμε τη θέση ενός σώματος συναρτήσει του χρόνου για ένα χρονικό διάστημα 10sec. Θα έπρεπε να χωρίσουμε το χρονικό διάστημα σε μικρότερα υπο-διαστήματα,  $\Delta t$ , και να υπολογίσουμε τη θέση σε κάθε υποδιάστημα. Και την διαδικασία αυτή να την επαναλάβουμε για κάθε  $\Delta t$

Οι γλώσσες προγραμματισμού προσφέρουν διάφορες δομές που επιτρέπουν για περισσότερο πολύπλοκες εκτελέσεις εντολών και ακολουθιών εκτέλεσης εντολών.

Μια δομή βρόχου ή loop μας επιτρέπει την εκτέλεση μιας εντολής ή ομάδας εντολών πολλές φορές, σύμφωνα με το ακόλουθο διάγραμμα

Για την PYTHON υπάρχουν οι ακόλουθοι τύποι επαναληπτικών διαδικασιών



# WHILE Loops

Η διαδικασία αυτή επαναλαμβάνει μια ομάδα εντολών καθόλη τη διάρκεια μια συνθήκη παραμένει αληθής.

Το πρόγραμμα εξετάζει πρώτα αν ικανοποιείται η συνθήκη και αν είναι αληθής εκτελεί την ομάδα των εντολών.

Η σύνταξη της δομής αυτής είναι:

**while expression :**  
**statements**

Η ομάδα των εντολών (statements) μπορεί να είναι **μια και μόνο** εντολή **ή μια σειρά** από εντολές.

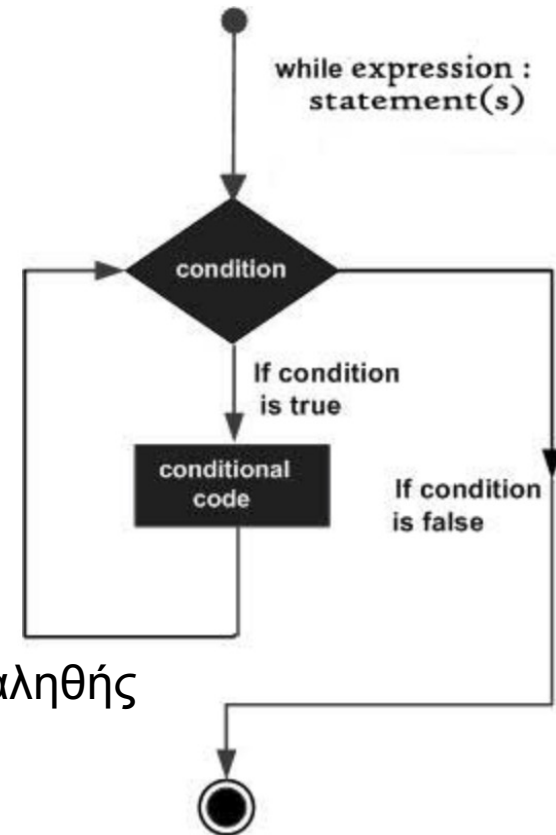
Θα πρέπει όλες να ξεκινούν κάτω από την ίδια στήλη

Η expression μπορεί να είναι οποιαδήποτε λογική έκφραση

Η διαδικασία επαναλαμβάνεται όσο η λογική έκφραση είναι αληθής

Τη στιγμή που έκφραση γίνεται ψευδής η ροή του προγράμματος περνά στην εντολή ακριβώς μετά το loop

**Προσοχή:** Το loop μπορεί να μην εκτελεστεί ποτέ αν η συνθήκη είναι ψευδής.



## WHILE Loops - Παράδειγμα

```
#!/usr/bin/python3
```

```
count = 0
```

```
while (count < 9):  
    print("The count is:". count)  
    count = count + 1
```

}

Το block των 2 εντολών print και count εκτελείται έως ότου το count γίνει 8

```
print "Good bye!"
```

```
The count is: 0  
The count is: 1  
The count is: 2  
The count is: 3  
The count is: 4  
The count is: 5  
The count is: 6  
The count is: 7  
The count is: 8  
Good bye!
```

## Ατέρμονο loop – Infinite loop

Υπάρχουν περιπτώσεις που η συνθήκη του while loop δεν γίνεται ποτέ ψευδής.

Τότε οι εντολές του loop επαναλαμβάνονται συνεχώς χωρίς να τερματίζεται

Θέλει ιδιαίτερη προσοχή ώστε να αποφεύγονται αυτές οι εσφαλμένες δομές while loop

```
#!/usr/bin/python3

val = 1

while val == 1:      # Auto tha dimiourgisei ena atermono loop
    num = input("Enter a number: ")
    print("You entered:", num)

print("Good bye!")
```

Το αποτέλεσμα του προγράμματος είναι:

```
Enter a number :20
You entered: 20
Enter a number :29
You entered: 29
Enter a number :3
You entered: 3
Enter a number between :Traceback (most recent call last):
  File "test.py", line 5, in <module>
    num = raw_input("Enter a number :")
KeyboardInterrupt
```

Συνεχίζει επ' άπειρο  
έως ότου δώσουμε ctr-c:

## While Loop – Χρήση εντολής else

Θα μπορούσαμε να συνδυάσουμε την εντολή while με μια εντολή else. Στην περίπτωση αυτή, όταν η συνθήκη του while loop γίνεται ψευδής εκτελούνται οι εντολές που βρίσκονται στο block του else

```
#!/usr/bin/python3

count = 0

while count < 5 :
    print(count, " is less than 5")
    count = count + 1
else :
    print(count, " is not less than 5")
```

Το αποτέλεσμα του κώδικα είναι:

```
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
5 is not less than 5
```

## While Loop – Block μιας εντολής

Θα μπορούσαμε να έχουμε μια και μόνο εντολή που να εκτελείται μετά την συνθήκη της δομής του loop.

**Είναι πολύ επικίνδυνη** γιατί οδηγεί σε ατέρμονα loop εφόσον δεν υπάρχει εντολή που να αλλάζει την συνθήκη του loop. **Πρέπει να αποφεύγεται**

```
#!/usr/bin/python3  
  
flag = 1  
  
while (flag): print("Given flag is really true!")  
print("Good bye!")
```

Απαιτείται CTRL-C για να σταματήσει η εκτέλεση αυτού του προγράμματος.

## FOR Loop – Δεύτερη δομή επανάληψης

Η δομή αυτή του loop εκτελεί την ακολουθία των εντολών πολλές φορές και διαχειρίζεται τον κώδικα που ελέγχει τον αριθμό των επαναλήψεων του loop.

Μπορεί να επαναλάβει κάποιες διαδικασίες χρησιμοποιώντας διάφορες ακολουθίες συμπεριλαμβανομένων αντικειμένων σε lists ή γραμματοσειρές (strings)

Η σύνταξη είναι η ακόλουθη:

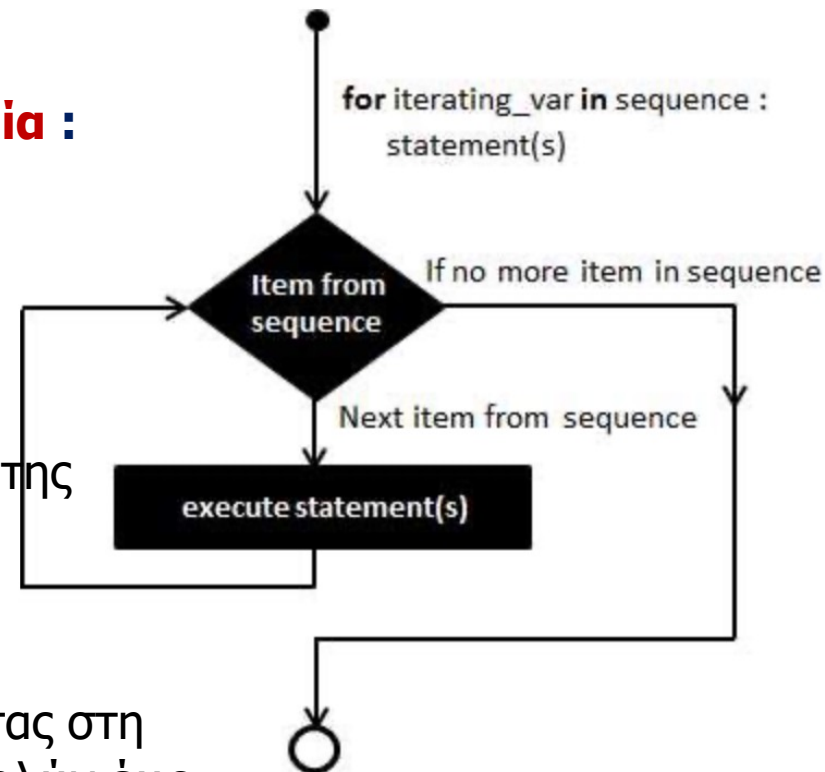
**for** μεταβλητή\_επανάληψης **in** ακολουθία :  
**statements**

Αν η ακολουθία περιέχει μια λίστα από λογικές εκφράσεις ελέγχεται πρώτη

Ακολουθεί η ανάθεση του πρώτου αντικειμένου της λίστας στην μεταβλητή επανάληψης

Εκτελείται μετά το block των εντολών

Ακολουθεί η ανάθεση των αντικειμένων της λίστας στη μεταβλητή επανάληψης και η εκτέλεση των εντολών έως ότου εξαντληθούν τα αντικείμενα της λίστας.





## FOR Loop – Παράδειγμα

```
#!/usr/bin/python3

for letter in "Python" :                # 1st example
    print("Current letter is : ",letter)

fruits = ["banana", "apple", "mango"]  # 2nd example
for fruit in fruits:
    print("Current fruit in the list is :",fruit)

print("Good bye!")
```

Εκτέλεση του διπλανού κώδικα δίνει:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

## FOR Loop – Χρήση του δείκτη θέσης της ακολουθίας

Θα μπορούσαμε να επαναλάβουμε μια διαδικασία εντολών χρησιμοποιώντας τον δείκτη της θέσης μέσα στην ακολουθία

```
#!/usr/bin/python3
```

```
fruits = ["banana", "apple", "mango"]  
for index in range(len(fruits)) :  
    print("Current fruit in the list is :", fruits[index])  
  
print("Good bye!")
```

Το πρόγραμμα θα δώσει:

```
Current fruit : banana  
Current fruit : apple  
Current fruit : mango  
Good bye!
```

Χρησιμοποιήσαμε την μέθοδο `len()` για να βρούμε το πλήθος των στοιχείων της λίστας και κατόπιν τη μέθοδο `range()` για να έχουμε την πραγματική ακολουθία των αριθμών της επανάληψης.

Η μέθοδος **`range()`** δημιουργεί μία ακολουθία αριθμών ξεκινώντας από το 0 οι οποίοι αυξάνονται κατά 1 και σταματά πριν κάποιο αριθμό N που δίνουμε ως όρισμα.

**Το εύρος των αριθμών είναι επομένως `[0,N)`**

Η πλήρης σύνταξη της `range()` είναι: **`range (start, stop, step)`**

**Start:** προαιρετικός ακέραιος για το που ξεκινά. Αν δεν καθοριστεί είναι 0

**Stop:** υποχρεωτικός ακέραιος για το που σταματά.

**Step:** προαιρετικός ακέραιος που δηλώνει πως αυξάνεται η ακολουθία. Κανονικά 1.

**Αποτέλεσμα:** Ακολουθία ακεραίων στο διάστημα **`[start, stop)`**

## FOR Loop – Χρησιμοποίηση else

Θα μπορούσαμε να έχουμε την εντολή else με μια δομή loop.

Αν η εντολή else χρησιμοποιηθεί με for loop, τότε η εντολή εκτελείται όταν το loop έχει εξαντλήσει την λίστα των αριθμών επανάληψης της ακολουθίας.

```
#!/usr/bin/python3
```

```
for num in range(10, 20) :           #Epanalipsi metaksu 10 kai 20
    for i in range(2, num) :         #Epanalipsi ws pros tous pithanoun diairetes
        if num % i == 0:             #tou arithmou num. Tha einai apo 2 ews num
            j = num/i                # % ypologizei to upoloipo tis diairesis 2
                                     # To piliko tis diairesis

            print("%d equals %d * %d"% (num,i,j))

            break                    #Diakopi tis epanaliptikis diadikasias gia
                                     #na metaferthoume ston epomeno num. Stamata
                                     #diladi tin epanalipsi ws pros to i

else:                                # else einai meros tou LOOP
    print(num, "is a prime number")
    break
```

```
10 equals 2 * 5
11 is a prime number
12 equals 2 * 6
13 is a prime number
14 equals 2 * 7
15 equals 3 * 5
16 equals 2 * 8
17 is a prime number
18 equals 2 * 9
19 is a prime number
```

## Nested Loop – Loop μέσα σε loop

Θα μπορούσαμε να έχουμε ένα loop μέσα σε άλλο loop.  
Η σύνταξη είναι η ακόλουθη:

```
for μεταβλητή_επανάληψης in ακολουθία :  
    for μεταβλητή_επανάληψης in ακολουθία :  
        statements  
    statements
```

Η σύνταξη για nested while loop είναι η ακόλουθη:

```
while expression :  
    while expression :  
        statements  
    statements
```

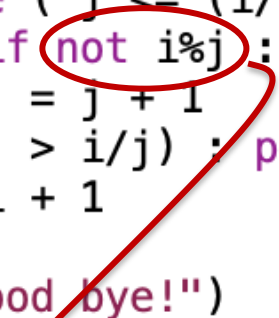
Εν γένει, θα μπορούσε να χρησιμοποιήσει κάποιος οποιοδήποτε είδος loop μέσα σε κάποιο άλλο είδος loop.

## Nested Loop – Παράδειγμα

```
#!/usr/bin/python3
```

```
i = 2
while(i < 100) :
    j = 2
    while ( j <= (i/j) ) :
        if not i%j : break
        j = j + 1
    if (j > i/j) : print(i, " einai prwtos")
    i = i + 1

print("Good bye!")
```



Μια true λογική έκφραση έχει τιμή 1

Μια false λογική έκφραση έχει τιμή 0

Όσο το υπόλοιπο της διαίρεσης του  $i$  με το  $j$  είναι μεγαλύτερο του 0, η έκφραση  $i \% j = \text{true}$  και επομένως η έκφραση:  $\text{not } i \% j = \text{false}$

Η επαναληπτική διαδικασία συνεχίζεται έως ότου  $i \% j = 0 = \text{false}$  (το υπόλοιπο είναι 0) και τότε η έκφραση:  $\text{not } i \% j = \text{not false} = \text{true}$

Το αποτέλεσμα θα είναι:

```
>>> exec(open("test1.py").read())
2  einai prwtos
3  einai prwtos
5  einai prwtos
7  einai prwtos
11 einai prwtos
13 einai prwtos
17 einai prwtos
19 einai prwtos
23 einai prwtos
29 einai prwtos
31 einai prwtos
37 einai prwtos
41 einai prwtos
43 einai prwtos
47 einai prwtos
53 einai prwtos
59 einai prwtos
61 einai prwtos
67 einai prwtos
71 einai prwtos
73 einai prwtos
79 einai prwtos
83 einai prwtos
89 einai prwtos
97 einai prwtos
Good bye!
```

## Εντολές ελέγχου με τα loops - BREAK

Υπάρχουν εντολές οι οποίες αλλάζουν τη ροή του προγράμματος του loop.  
Προσοχή ότι όταν τελειώνει η εκτέλεση όλες οι μεταβλητές που είχαν δημιουργηθεί καταστρέφονται

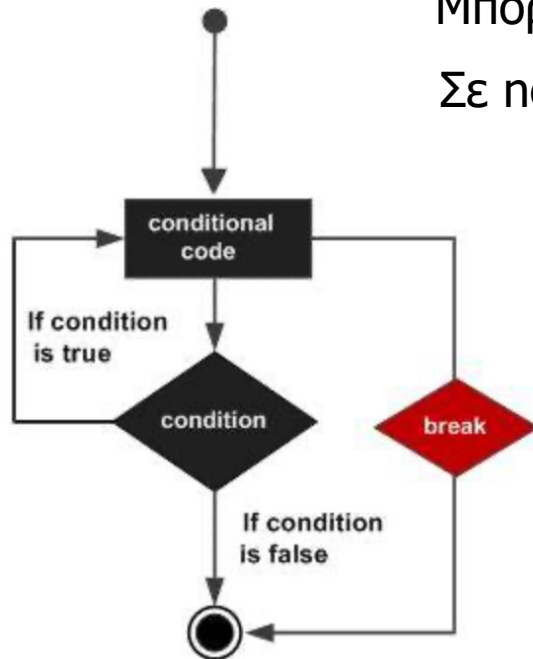
Υπάρχουν 3 εντολές που υποστηρίζει η PYTHON

### **break** εντολή

Σταματά την ακολουθία του loop και μεταφέρει τη ροή στην πρώτη εντολή μετά το loop

Μπορεί να χρησιμοποιηθεί τόσο σε while όσο και σε for loops

Σε nested loops σταματά τη ροή του πιο εσωτερικού loop



## BREAK – Παράδειγμα

```
#!/usr/bin/python3
```

```
for letter in "Python" :      # 1st example
    if letter == 'h' :
        break
    print("Current letter is :", letter)
```

```
var = 10                      # 2nd example
while var > 0:
    print("Current value of variable var = ",var)
    var = var - 1
    if var == 5:
        break

print("Good bye!")
```

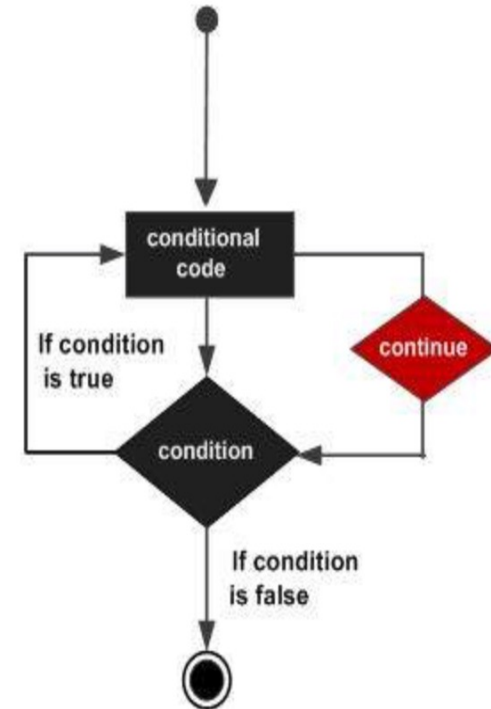
```
Current Letter : P
Current Letter : y
Current Letter : t
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Good bye!
```

# Εντολές ελέγχου με τα loops - **CONTINUE**

Επιστρέφει τη ροή στην αρχή του while loop

Απορρίπτει όλες τις υπόλοιπες εντολές της ακολουθίας του loop στην τρέχουσα επανάληψη και μετακινεί τη ροή στην αρχή του loop

## **continue** εντολή



```
#!/usr/bin/python3
```

```
for letter in "Python" :      # 1st example
    if letter == 'h' :
        continue
    print("Current letter is :", letter)

var = 10                      # 2nd example
while var > 0:
    var = var - 1
    if var == 5:
        continue
    print("Current value of variable var = ",var)

print("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Current variable value : 4
Current variable value : 3
Current variable value : 2
Current variable value : 1
Current variable value : 0
Good bye!
```



## Εντολές ελέγχου με τα loops - Pass

Χρησιμοποιείται όταν κάποια εντολή χρειάζεται συντακτικά αλλά δεν θέλετε να χρησιμοποιήσετε συγκεκριμένη εντολή

Είναι μια μηδενική εντολή **pass** εντολή

```
#!/usr/bin/python3
```

```
for letter in "Python" :  
    if letter == 'h' :  
        pass  
        print(" This is pass block")  
    print("Current letter is :", letter)  
  
print("Good bye!")
```

```
Current Letter : P  
Current Letter : y  
Current Letter : t  
This is pass block  
Current Letter : h  
Current Letter : o  
Current Letter : n  
Good bye!
```