

# Εισαγωγή στην **Python**

**5**



### *Copyright*

Το παρόν εκπαιδευτικό υλικό προσφέρεται ελεύθερα υπό τους όρους της άδειας Creative Commons:

- *Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Όχι Παράγωγα Έργα 3.0.*

Για να δείτε ένα αντίγραφο της άδειας αυτής επισκεφτείτε τον ιστότοπο

<https://creativecommons.org/licenses/by-nc-nd/3.0/gr/>

Στ. Δημητριάδης, 2015

# Περιεχόμενα

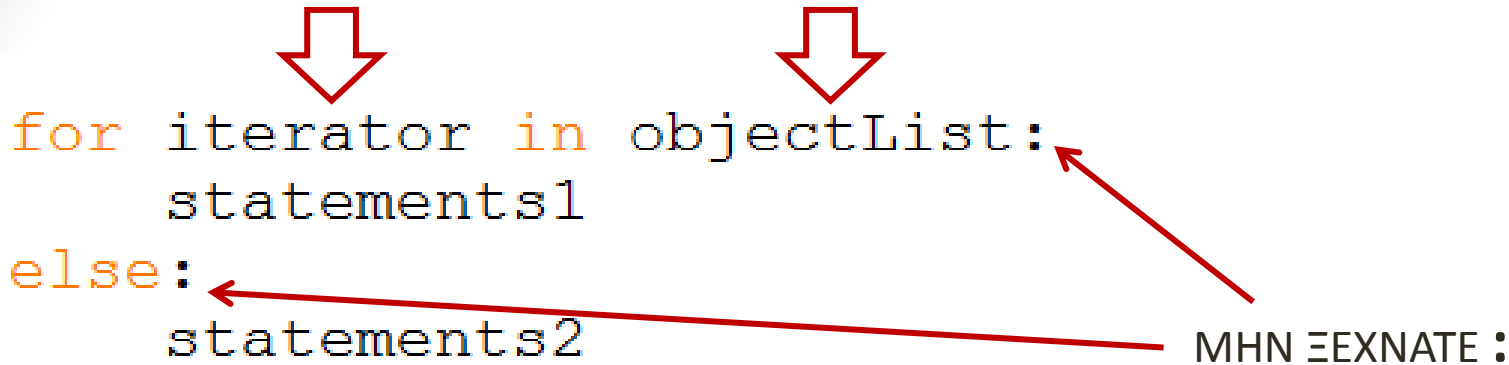
**for**

Δομή επανάληψης

Σύνθετη δήλωση (Compound statement)

# for

## η γενική μορφή



```
for iterator in objectList:  
    statements1  
else:  
    statements2
```

Diagram illustrating the general form of a `for` loop. Red arrows point from the title 'η γενική μορφή' to the components of the loop: 'iterator' (labeled 'ΜΗΝ ΞΕΧΝΑΤΕ :'), 'objectList', and the 'else:' block.

- **iterator**: απαριθμητής (μεταβλητή απαρίθμησης των επαναλήψεων)
- **objectList**: απαριθμήσιμη λίστα (iterable) διακριτών αντικειμένων που καθορίζουν αρχική και τελική τιμή του δείκτη
- **else**: προαιρετικός κλάδος που εκτελείται 1 φορά αφού ολοκληρωθεί η επανάληψη (και δεν έχει γίνει έξοδος με `break`)

# for

## Παραδείγματα

- 1

```
suma = 0
for x in [1, 2, 3, 4]:
    suma = suma + x

print(suma)
```

```
gin = 1
for item in [1, 2, 3, 4]: gin *= item
print(gin)
```

- Η for μπορεί να χρησιμοποιεί μια απλή λίστα τιμών
- Ο απαριθμητής x παίρνει διαδοχικά τις τιμές 1 ως και 4

- Ο απαριθμητής item παίρνει διαδοχικά τις τιμές 1 ως και 4
  - Αν ο βρόχος αποτελείται από 1 δήλωση μπορεί να γραφεί όπως παραπάνω σε 1 σειρά

# for

## Παραδείγματα

- 2

```
for x in ['spam', 'eggs', 'ham']:  
    print(x, end=' ')
```

spam eggs ham

- Μια **λίστα** μπορεί να περιέχει αλφαριθμητικά
- Ο **απαριθμητής** x παίρνει ως τιμές τα αλφαριθμητικά της λίστας

```
message = 'SPAM'  
for x in message:  
    print(x, '/', end=' ')
```

S / P / A / M /

- Ένα **αλφαριθμητικό** μπορεί να μπει στη λίστα της for καθώς θεωρείται **λίστα** χαρακτήρων
- Ο **απαριθμητής** x παίρνει ως τιμές τους χαρακτήρες του αλφαριθμητικού

# for & η συνάρτηση range( )

- Ένας **πρακτικός** και **συνηθισμένος** τρόπος για να γράφουμε την επανάληψη for είναι με χρήση της συνάρτησης **range( )**
- Η **range(start, end)** επιστρέφει ένα απαριθμήσιμο αντικείμενο: μια **λίστα** αριθμών στο διάστημα [start, end)
- Γενική σύνταξη: **range(start, end [, step])**
  - **start**: αρχική τιμή (συμπεριλαμβάνεται)
  - **end**: τιμή άνω ορίου (**δεν** συμπεριλαμβάνεται)
  - **step**: βήμα μεταβολής (προαιρετικό)

# for & range()

## Παραδείγματα

- 1

ΚΩΔΙΚΑΣ

ΕΞΟΔΟΣ

```
for x in range(0,10):  
    print(x, end=' ')
```

```
0 1 2 3 4 5 6 7 8 9
```

```
for x in range(5,10):  
    print(x, end=' ')
```

```
5 6 7 8 9
```

```
for x in range(-5,0):  
    print(x, end=' ')
```

```
-5 -4 -3 -2 -1
```

```
for x in range(0,10,2):  
    print(x, end=' ')
```

```
0 2 4 6 8
```



# for & range()

## Παραδείγματα

- 2

ΚΩΔΙΚΑΣ

ΕΞΟΔΟΣ

```
for x in range(5):  
    print(x, end=' ')
```

0 1 2 3 4

```
for x in range(0,10,3):  
    print(x, end=' ')
```

0 3 6 9

```
a=5; b=10; c=3  
for x in range(a,b,c):  
    print(x)
```

5  
8

10

20

30

40

50

60

70

80

90

```
a=10  
b=100  
c=b//a  
for x in range(a,b,c):  
    print(x)
```

# for & range()

## Παραδείγματα

- 3

### ΚΩΔΙΚΑΣ

```
x = range(1,5)  
print(x)
```

```
for i in x:  
    print(i)
```

### ΕΞΟΔΟΣ

```
range(1, 5)  
1  
2  
3  
4
```

- Η `x=range(1,5)` συνδέει το `x` με μία απαριθμήσιμο δομή
- Στη συνέχεια η `for` λειτουργεί με τη `x` στη θέση της απαριθμήσιμης δομής

```
for i in range(1,10):  
    i += 1  
    print(i)
```

```
2  
3  
4  
5  
6  
7  
8  
9  
10
```

- Η `i += 1` προκαλεί σε κάθε επανάληψη αύξηση της τιμής του `i` κατά 1
- Έτσι τελικά το πεδίο τιμών του απαριθμητή είναι το `[2, 10]`

# for & range()

## Παραδείγματα

- 4

```
for x in range(10, 5, -1):  
    print(x)
```

10  
9  
8  
7  
6

```
for i in range(10, 2, -3):  
    print(i, ' ', end='')
```

10 7 4

```
for ch in range(70, 64, -1):  
    print(chr(ch), ' ', end='')
```

F E D C B A

# for iterator & iterable

```
for iterator in iterable:
```

for	iterator	iterable
• for i in [1,2,3,4]	i	[1,2,3,4]
• for x in ['ena', 'dyo', 'tria']	x	['ena', 'dyo', 'tria']
message = 'SPAM'		
• for letter in message	letter	message
• <b>for x in range(1, 10)</b>	x	range(1, 10)
• .....		

• Η ποικιλία των απαριθμήσιμων δομών στην Python είναι εξαιρετικά μεγάλη και προσφέρει εξαιρετική **ευελιξία** στο γράψιμο κώδικα επανάληψης με εντολή for

# for τι συμβαίνει με τις **break** & **continue**

```
for target in objectList:
    statements1
    if test1: break
    if test2: continue
else:
    statements2
```


- **break**: άμεση έξοδος από το βρόχο for
  - αν υπάρχει else αγνοείται
- **continue**: άμεση μετάβαση στην επόμενη επανάληψη του βρόχου
  - αν υπάρχουν εντολές μετά την continue αγνοούνται

# for & break

## Παράδειγμα

```
x=int(input('Ακέραιος [1, 10]: '))
suma=0

if x<1:
    print('Αριθμός μικρότερος του 1')
else:
    for i in range(0,x+1):
        if i>10: break
        suma += i
    print('Αθροισμα: ', suma)
```



- Η **break** σταματά την εκτέλεση της επανάληψης αν η τιμή του *i* ξεπεράσει το όριο του 10
- Μετά την *break* εκτελείται η τελευταία *print*

# for & continue

## Παράδειγμα

```
x=int(input('Ακέραιος [1, 10]: '))
suma=0
sumper=0

if x<1:
    print('Αριθμός μικρότερος του 1')
else:
    for i in range(0,x+1):
        if i>10: break
        suma += i
        if i%2==0: continue
        sumper += i
    print('Αθροισμα: ', suma)
    print('Αθροισμα Περιπτώων: ', sumper)
```

- Αν το υπόλοιπο (%) της διαίρεσης του  $i$  με το 2 είναι 0 εκτελείται η **continue**
- Η **continue** προκαλεί αμέσως την επόμενη επανάληψη του βρόχου, ενώ η εντολή  $sumper += i$  αγνοείται

# for iterator & iterable *Σύνοψη*

```
for iterator in iterable:
```

- **Iterator**      Απαριθμητής
- **Iterable**      Απαριθμήσιμη δομή (λίστα αντικειμένων)
- Γενικά η for δουλεύει σωστά με έναν **απαριθμητή** (iterator) που παίρνει διαδοχικά τιμές από μια **απαριθμήσιμη δομή** (iterable)
- Μια δομή θεωρείται **‘απαριθμήσιμη’** (iterable) αν έχει τη μορφή **ακολουθίας στοιχείων** (sequence), στην οποία να είναι δυνατό να καθοριστεί με ακρίβεια η σειρά, δηλ. ποιο είναι το **‘επόμενο’** και ποιο το **‘προηγούμενο’** στοιχείο
- Ο απαριθμητής παίρνει **αυτόματα** τον τύπο των δεδομένων που συνθέτουν την απαριθμήσιμη δομή