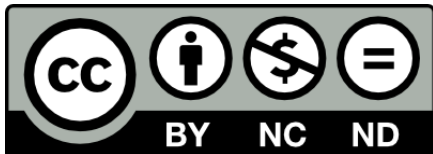


Εισαγωγή στην **Python**

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
- for και η συνάρτηση range()
- Break & Continue στη for
- Σύνοψη: iterator (επαναλήπτης) & iterable (επαναληπτική δομή)

Περιεχόμενα

for

Δομή επανάληψης

Σύνθετη εντολή (Compound statement)



for

η γενική μορφή



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

ΜΗΝ ΞΕΧΝΑΤΕ :

- **iterator**: επαναλήπτης (όνομα που ‘απαριθμεί’ τις επαναλήψεις)
- **iterable**: επαναληπτική δομή (δηλ. επιστρέφει το ένα μετά το άλλο μια σειρά από τα διακριτά αντικείμενα–τιμές που παίρνει ο επαναλήπτης)
- **else**: προαιρετικός κλάδος που εκτελείται 1 φορά αφού ολοκληρωθεί η επανάληψη (και δεν έχει γίνει έξοδος με break)

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

print(suma)
```

Η for μπορεί να χρησιμοποιεί μια απλή λίστα τιμών ως επαναληπτική δομή

- Ο επαναλήπτης x παίρνει διαδοχικά τις τιμές 1 ως και 4

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

- Ο επαναλήπτης 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 **επαναλήπτης** 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 χρησιμοποιηθεί ως επαναλήψιμη δομή της for καθώς θεωρείται **λίστα** χαρακτήρων
- Ο **επαναλήπτης** 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 μια ακολουθία τιμών (επαναληπτική ακολουθιακή δομή) στο διάστημα [start, end)
 - *ΣΗΜΑΝΤΙΚΟ:* Η range() δεν παράγει την ακολουθία τιμών αμέσως αλλά *κάθε φορά που επαναλαμβάνεται ο βρόχος επιστρέφει την επόμενη τιμή*
 - Είναι μια συνάρτηση – **γεννήτορας** (generator)
- 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

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

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

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

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

ΕΞΟΔΟΣ

0 1 2 3 4 5 6 7 8 9

5 6 7 8 9

-5 -4 -3 -2 -1

0 2 4 6 8

for & range()

Παραδείγματα

- 2

ΚΩΔΙΚΑΣ

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

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

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

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

ΕΞΟΔΟΣ

0 1 2 3 4

0 3 6 9

5
8

10
20
30
40
50
60
70
80
90

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 όνομα `x` με τη συνάρτηση-γεννήτορα `range()`
- Στη συνέχεια η `for` λειτουργεί με τη `x` στη θέση της `range()`

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

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

- Ο επαναλήπτης μπορεί να **αλλάξει** τιμή μέσα στο βρόχο
- Η `i += 1` προκαλεί σε κάθε επανάληψη σύνδεση του `i` με τιμή αυξημένη κατά +1
- Έτσι μέσα στο βρόχο το πεδίο τιμών του επαναλήπτη 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', 'dya', 'tria'] x ['ena', 'dya', 'tria']

```
message = 'SPAM'
```

- | | | |
|--------------------------------|--------|--------------|
| • for letter in message | letter | message |
| • for x in range(1, 10) | x | range(1, 10) |

.....

- Η ποικιλία των δομών ακολουθίας (sequences) στην Python είναι μεγάλη και προσφέρει εξαιρετική **ευελιξία** στο γράψιμο κώδικα επανάληψης με εντολή **for**



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

```
for iterator in iterable:
    statements1
    if test1: break
    if test2: continue
    statements2
else:
    statements3
```


- **break**: άμεση έξοδος από το βρόχο for
 - Αν υπάρχει else αγνοείται
 - Στο παράδειγμα: αν εκτελεστεί η break οι statements2 & statements3 δεν εκτελούνται
- **continue**: άμεση μετάβαση στην επόμενη επανάληψη του βρόχου
 - Αν υπάρχουν εντολές μετά την continue αγνοούνται
 - Στο παράδειγμα: αν εκτελεστεί η continue οι statements2 δεν εκτελούνται αλλά ίσως εκτελεστούν σε επόμενη επανάληψη

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 Περιττών: ', 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** επαναλήπτης
- **Iterable** επαναληπτική δομή
 - Γενικά η for δουλεύει με έναν **δείκτη-επαναλήπτη** (iterator) που παίρνει διαδοχικά τιμές από μια **επαναληπτική δομή** (iterable)
 - Μια δομή θεωρείται **‘επαναληπτική’** (iterable) αν αποτελείται από διακριτά αντικείμενα-τιμές τα οποία είναι δυνατόν να επιστρέφει το ένα μετά το άλλο στον επαναλήπτη
 - Ο επαναλήπτης παίρνει **αυτόματα** τον τύπο δεδομένου που συνθέτουν την επαναληπτική δομή