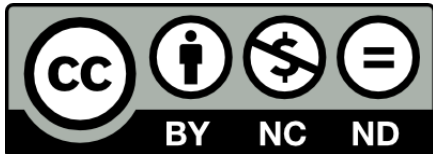


Εισαγωγή στην **Python**

4





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

- Η εννοιολογική ιεραρχία της Python
 - Πρόγραμμα, Τμήμα, Εντολές, Εκφράσεις, Αντικείμενα
- Η εντολή import
- Η συνάρτηση random.randint()
 - Ψευδοτυχαίοι αριθμοί
- while – Εντολή επανάληψης
- if – Εντολή ελέγχου
- Συνθήκες (Conditions)
- Τελεστές σύγκρισης (Comparison operators)
- Μπλοκ & Εσοχές/Στοίχιση (Indentation)
- Η εντολή break

- While/else – η Πλήρης μορφή
- If/elif/else – η Πλήρης μορφή
- break – continue - pass

Η εννοιολογική ιεραρχία της Python

Python Conceptual Hierarchy

1. Ένα **Πρόγραμμα** (program) συντίθεται από κώδικα που μπορεί να βρίσκεται σε διάφορα **Τμήματα** (modules)
 - Μια κλασσική οργάνωση/σύνθεση είναι:
 - (1) Κώδικας **κύριου** προγράμματος
 - (2) Κώδικας σε/από **βιβλιοθήκες**
2. Τα Τμήματα (modules) περιέχουν **Εντολές** (statements/commands)
3. Οι Εντολές περιλαμβάνουν **Εκφράσεις** (expressions)
4. Οι Εκφράσεις δημιουργούν και επεξεργάζονται **Αντικείμενα** (Objects)



Εντολές – Εκφράσεις – Τμήματα

- **Εντολή** – (Command/Statement)
 - Οι εντολές είναι οδηγίες/εντολές προς τον διερμηνέα που εκτελούν κάποια **ενέργεια**, Πχ. Import, for, while, ...
 - Μια εντολή δεν αφορά οπωσδήποτε τον υπολογισμό μιας τιμής (όπως συμβαίνει σε μια έκφραση)
- **Έκφραση** – Expression
 - Οι εκφράσεις δημιουργούνται από συνδυασμό μεταβλητών, τελεστών και τιμών και **περιγράφουν τον υπολογισμό μιας νέας τιμής**
 - Πχ. $a * x ** 2 + b * x + c$
 - Κάθε νόμιμη έκφραση στην Python μπορεί να χρησιμοποιηθεί ως εντολή ("expression statement"). Κατ' αυτή την έννοια οι εκφράσεις «είναι» και εντολές.
- **Τμήμα** - Module
 - Τα Τμήματα (ή αρθρώματα) είναι **κώδικας Python που περιλαμβάνουν πρόσθετες προγραμματιστικές δομές** (συναρτήσεις, κλάσεις, κλπ.). Μπορείτε να χρησιμοποιήσετε τα περιεχόμενα ενός τμήματος εισάγοντάς τα στο πρόγραμμά σας με χρήση της εντολής **import**

Import

Εντολή (Statement)

while

Δομή επανάληψης

Σύνθετη εντολή (Compound statement)

if

Δομή ελέγχου

Σύνθετη εντολή (Compound statement)



Παιχνίδι: Μάντεψε τον αριθμό

```
# Guess the number game -- Μάντεψε τον αριθμό

import random

tries = 0

my_name = input('Γειά σου! Πώς σε λενε; ')
number = random.randint(1, 20)
print('Λοιπόν, ' + my_name + ',\n      σκέφτομαι έναν αριθμό ανάμεσα σε 1 και 20. Μπορείς να τον βρείς;')

while tries < 6:
    # Υπάρχουν 4 κενά πριν από την παρακάτω print
    guess = int(input('Μάντεψε: '))
    tries = tries + 1

    # Υπάρχουν 8 κενά πριν από τις παρακάτω print & break -- Γιατί;
    if guess < number:
        print('Μάντεψες χαμηλά.')

    if guess > number:
        print('Μάντεψες ψηλά.')

    if guess == number:
        break

if guess == number:
    tries = str(tries)
    print('Ωραία, ' + my_name + '! Βρήκες τον αριθμό σε ' + tries + ' προσπάθειες!')
else:
    number = str(number)
    print('Κριμα! Ο αριθμός που σκέφτηκα ήταν ' + number)
```

Προσαρμογή από: Al Sweigart, [Invent Your Own Computer Games with Python](#), 2nd Ed.

Η εντολή **import**

```
import random
```

- Η εντολή **import** συνδέει τον κώδικα ενός τμήματος (module)* με το κυρίως πρόγραμμα
- Η **import** εκτελεί δύο βασικές λειτουργίες:
 - α) εκτελεί αναζήτηση για να βρει το module που έχει ζητηθεί
 - β) συνδέει (binds) το αποτέλεσμα της αναζήτησης σε έναν αναγνωριστή (δηλ. όνομα) τοπικής εμβέλειας (local scope)**
- Στο παράδειγμα το τμήμα **random** συνδέεται ώστε να μπορούμε να χρησιμοποιήσουμε τα στοιχεία που περιέχει (πχ. συναρτήσεις, μεθόδους, αντικείμενα, κλπ.)

* Δείτε πλήρη λίστα των διαθέσιμων modules δίνοντας >>> **help('modules')**

** Η έννοια της τοπικής εμβέλειας εξηγείται στην ενότητα των συναρτήσεων

Η συνάρτηση random.randint()

```
number = random.randint(1, 20)
```

- Η συνάρτηση randint περιλαμβάνεται στο module με όνομα random
- Για να την καταλάβει ο διερμηνέας θα πρέπει να κληθεί με χρήση της **σημειογραφίας τελείας** (dot notation) που βλέπετε
- Γενικά: Κάθε στοιχείο που καλείται από ένα module θα πρέπει να γραφεί με τη σημειογραφία τελείας, ως εξής:
- **Όνομα Τμήματος + Τελεία + Όνομα Στοιχείου**

Και στον κανόνα αυτόν υπάρχουν εξαιρέσεις που θα μάθετε αργότερα

Πέρασμα Ορισμάτων στη συνάρτηση

```
number = random.randint(1, 20)
```

- Οι ακέραιες τιμές (1, 20) είναι τα **ορίσματα** (arguments) που περνάμε στη συνάρτηση για να καθορίσουμε ένα διάστημα τιμών: 1 ως 20
- Η randint επιστρέφει ένα **ψευδοτυχαίο ακέραιο** στο διάστημα [1, 20]
- Η γενική μορφή είναι: **random.randint(a, b)**
- Επιστρέφει έναν ψευδοτυχαίο ακέραιο N τέτοιο ώστε: $a \leq N \leq b$
- Περισσότερα για τη random:
- <https://docs.python.org/3.3/library/random.html#module-random>

Εξηγήστε τον κώδικα

```
import random

tries = 0

my_name = input('Γεια σου! Πώς σε λενε; ')
number = random.randint(1, 20)
print('Λοιπόν, ' + my_name + ',\n      σκέφτομαι έναν αριθμό ανάμεσα σε 1 και 20. Μπορείς να τον βρείς;')
```

- Με βάση αυτά που γνωρίζετε εξηγήστε τον κώδικα στις πρώτες γραμμές του προγράμματος

while δομή επανάληψης*

- 1

```
while tries < 6:
```

Δεσμευμένη λέξη

Συνθήκη

:

Δηλώνει την έναρξη του
block (ομάδας εντολών)
επανάληψης

* Σημ. Εδώ παρουσιάζονται οι **απλές** μορφές της while & if.

Οι **πλήρεις** μορφές παρουσιάζονται στο τέλος του παρόντος αρχείου διαφανειών

while δομή επανάληψης

- 2

- **Λειτουργία:** ➡
- Ελέγχεται η Λογική τιμή της συνθήκης
- Αν η τιμή είναι **True** επαναλαμβάνεται η εκτέλεση του βρόχου εντολών
- Ελέγχεται πάλι η Λογική τιμή της συνθήκης
- Όταν η τιμή γίνει **False** παύει η επανάληψη του βρόχου και εκτελείται η πρώτη εντολή μετά τον βρόχο while

```
while tries < 6:
```

FALSE

```
# Υπάρχουν 4 κενά πριν από την παρακάτω print
guess = int(input('Μάντεψε: '))
tries = tries + 1
```

```
# Υπάρχουν 8 κενά πριν από τις παρακάτω print & break -- Γιατί;
```

```
if guess < number:
    print('Μάντεψες χαμηλά.')
```

```
if guess > number:
    print('Μάντεψες ψηλά.')
```

```
if guess == number:
    break
```

TRUE

Πρώτη εντολή μετά το Βρόχο While

if δομή ελέγχου

```
if guess < number:
```

Δεσμευμένη λέξη

Συνθήκη

:

Δηλώνει την έναρξη του block
(ομάδας εντολών) ελέγχου

Θα εκτελεστεί εφόσον η
συνθήκη είναι αληθής (True)



Συνθήκες - Conditions

- Μια **συνθήκη** (condition) είναι μια έκφραση η οποία συνδυάζει δύο τουλάχιστον τιμές ή μεταβλητές μέσω ενός **τελεστή σύγκρισης** (comparison operator) και επιστρέφει μια λογική (Boolean) τιμή (True ή False)
- Προσέξτε τη διαφορά μεταξύ του τελεστή ανάθεσης τιμή (=) και του τελεστή σύγκρισης “ίσο με” (==)

```
if guess == number:
```

Τελεστές σύγκρισης στην Python

Comparison Operators

Operator Sign	Operator Name
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Πηγή: Lutz, M. (2013). *Learning Python*, 5th ed., O'Reilly: Cambridge

Συνολικά για τους τελεστές δείτε:

http://www.tutorialspoint.com/python/python_basic_operators.htm



Εξηγήστε τον παρακάτω κώδικα

- Πώς εκτελούνται οι παρακάτω εντολές if ;

```
if guess < number:  
    print('Μάντεψες χαμηλά.')  
if guess > number:  
    print('Μάντεψες ψηλά.')  
if guess == number:  
    break
```

Blocks εντολών & Στοιίχιση (Indentation) - 1

- ΠΡΟΣΟΧΗ: Στην Python ο κενός χώρος (**στοίχιση** των γραμμών κώδικα) μεταφέρει πληροφορία!

```
while tries < 6:
    # Υπάρχουν 4 κενά πριν από την παρακάτω print
    guess = int(input('Μάντεψε: '))
    tries = tries + 1

    # Υπάρχουν 8 κενά πριν από τις παρακάτω print & break -- Γιατί;
    if guess < number:
        print('Μάντεψες χαμηλά.')

    if guess > number:
        print('Μάντεψες ψηλά.')

    if guess == number:
        break
```

- Παρατηρήστε την **στοίχιση** του block εντολών του βρόχου while! Είναι **υποχρεωτική** !
- Όστε ο διερμηνέας να καταλαβαίνει ότι πρόκειται για το block της while

Blocks εντολών & Στοιίχιση (Indentation) - 2

```
while tries < 6:
```

1

```
# Υπάρχουν 4 κενά πριν από την παρακάτω print
```

```
guess = int(input('Μάντεψε: '))
```

```
tries = tries + 1
```

```
# Υπάρχουν 8 κενά πριν από τις παρακάτω print & break -- Γιατί;
```

```
if guess < number:
```

```
    print('Μάντεψες χαμηλά.')
```

2

```
if guess > number:
```

```
    print('Μάντεψες ψηλά.')
```

3

```
if guess == number:
```

```
    break
```

4

- Πόσα block εντολών υπάρχουν στον κώδικα;
- Το block 1 είναι ο κώδικας του βρόχου επανάληψης while
- Τα block 2, 3 και 4 είναι ο κώδικας που αντιστοιχεί στις εντολές if

Επομένως...

- Ένα μπλοκ εντολών (ή στα ελληνικά **πλοκάδα***) είναι μια ή περισσότερες γραμμές κώδικα που ομαδοποιούνται γιατί έχουν **το ίδιο επίπεδο στοίχισης**
- Μπορείτε άμεσα να ελέγχετε την αρχή και το τέλος ενός μπλοκ ελέγχοντας το επίπεδο στοίχισης (δηλ. το **πλήθος των κενών** πριν από τις γραμμές κώδικα)

```
while tries < 6:
    # Υπάρχουν 4 κενά πριν από την παρακάτω print
    guess = int(input('Μάντεψε: '))
    tries = tries + 1

    # Υπάρχουν 8 κενά πριν από τις παρακάτω print & break -- Γιατί;
    if guess < number:
        print('Μάντεψες χαμηλά.')

    if guess > number:
        print('Μάντεψες ψηλά.')

    if guess == number:
        break
```

Η στοίχιση δείχνει το block

* Για τον όρο «πλοκάδα» δείτε [εδώ](#) και [εδώ](#)

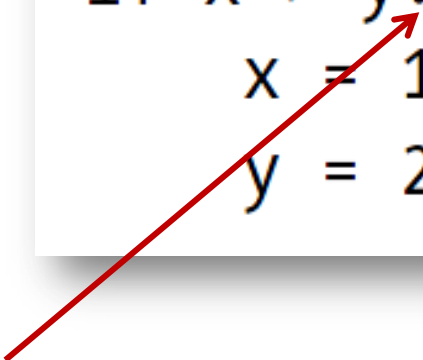
Μια σύγκριση

C-like σύνταξη

```
if (x > y) {  
    x = 1;  
    y = 2;  
}
```

Python

```
if x > y:  
    x = 1  
    y = 2
```



- Η Python **προσθέτει** το **':'** ως σύμβολο έναρξης ενός μπλοκ
- Η Python **αφαιρεί**:
 - Τις παρενθέσεις **()**
 - Το ερωτηματικό **;**
 - Τα άγκιστρα **{ }**

- Γενικά θα πρέπει να ασκηθείτε ώστε να διακρίνετε τα μπλοκ εντολών (τις πλοκάδες!) μέσα σε κώδικα Python (όπως στο σχήμα δεξιά)

```
from pylab import *  
  
x=0  
data=[]  
t=0  
  
dt=0.1  
v=0  
a=2  
  
while t<3:  
    x=x+v*dt  
    v=v+a*dt  
    t=t+dt  
    data=data+[[x,t]]  
  
savetxt('temp_data.txt',data)
```

Block 1

Block 2

Block 3

Block 2, continuation

Block 1, continuation

- Ποιό είναι το μπλοκ της while στον κώδικα αριστερά;
- Ποια εντολή εκτελείται μετά το βρόχο while;

- Πόσα επίπεδα στοίχισης διακρίνετε στον κώδικα δεξιά;
 - Μην ανησυχείτε αν δεν καταλαβαίνετε τις εντολές
- Αντιστοιχίστε τις **if** με τις **else** με βάση απλά το επίπεδο στοίχισης

```
def add5(x):  
    return x+5  
  
def dotwrite(ast):  
    nodename = getNodeName()  
    label=symbol.sym_name.get(int(ast[0]),ast[0])  
    print ' %s [label="%s' % (nodename, label),  
    if isinstance(ast[1], str):  
        if ast[1].strip():  
            print '= %s";' % ast[1]  
        else:  
            print '"]'  
    else:  
        print '"]';'  
        children = []  
        for n, child in enumerate(ast[1:]):  
            children.append(dotwrite(child))  
        print ' %s -> {' % nodename,  
        for name in children:  
            print '%s' % name,
```

- Στον κώδικα δεξιά αν θέλετε να γράψετε μια εντολή **if μέσα στο μπλοκ της while** σε ποιο επίπεδο στοίχισης θα τη γράφατε;
- Αν η εντολή **if** θα έπρεπε να εκτελείται **μετά τη while** σε ποιο επίπεδο στοίχισης θα τη γράφατε;

```
n = 1
sum1toN = 0
while n <= 5:
    sum1toN += n
    print('{0:3d}: {1:5d}'.format(n, sum1toN))
    n += 1
```


- Τι θα συμβεί κατά την εκτέλεση του κώδικα στις δύο διαφορετικές μορφές δεξιά;

A

```
lines = 0
while lines < 50:
    lines = lines + 1
    print("I love to write code!")
```

B

```
lines = 0
while lines < 50:
    lines = lines + 1
print("I love to write code!")
```

Εντολή **break**

- **Έξοδος** από τον βρόχο επανάληψης **while**
- Η **break** "σπάζει" την εκτέλεση του μπλοκ επανάληψης μέσα στο οποίο βρίσκεται
- Η εκτέλεση συνεχίζεται με την πρώτη γραμμή μετά το βρόχο **while**

```
while tries < 6:
    # Υπάρχουν 4 κενά πριν από την παρακάτ
    guess = int(input('Μάντεψε: '))
    tries = tries + 1

    # Υπάρχουν 8 κενά πριν από τις παρακάτ
    if guess < number:
        print('Μάντεψες χαμηλά.')

    if guess > number:
        print('Μάντεψες ψηλά.')

    if guess == number:
        break

if guess == number:
    tries = str(tries)
    print('Ωραία, ' + my_name + '! Βρήκες')
else:
    number = str(number)
    print('Κριμα! Ο αριθμός που σκέφτηκα ή')
```

Εξηγήστε τον κώδικα

```
if guess == number:
    tries = str(tries)
    print('Ωραία, ' + my_name + '! Βρήκες τον αριθμό σε ' + tries + ' προσπάθειες!')
else:
    number = str(number)
    print('Κριμα! Ο αριθμός που σκέφτηκα ήταν ' + number)
```

- Με βάση αυτά που γνωρίζετε εξηγήστε τον κώδικα στις τελευταίες γραμμές του προγράμματος

while & if

οι πλήρεις μορφές



While/else δομή επανάληψης

- 1

```
while test:                # Loop test
    statements-1           # Loop body
else:                      # Optional else
    statements-2           # Run if didn't exit loop with break
```

- Η **πλήρης** μορφή της δομής while/else
- Όσο η συνθήκη *test* είναι Αληθής (**True**) εκτελείται η πλοκάδα statements-1
- Εάν η *test* γίνει Ψευδής (**False**) εκτελείται το block statements -2 (κλάδος else) και στη συνέχεια η ροή βγαίνει από το βρόχο
- Η else είναι **προαιρετική** (optional). Αν δεν υπάρχει τότε αν η *test* γίνει Ψευδής (**False**) η ροή βγαίνει από το βρόχο
- Εάν η πλοκάδα statements-1 περιλαμβάνει την **break** τότε γίνεται έξοδος από το βρόχο χωρίς να εκτελεστεί ο κλάδος else



While/else δομή επανάληψης

- 2

- Παραδείγματα

```
while True:
    print('Type Ctrl-C to stop me!')
```

```
a=0; b=10
while a < b:
    print(a, end=' ')
    a += 1          # ή a = a + 1
```

```
a=0; b=10
while a < b:
    print(a, end=' ')
    a += 1
else:
    print('a >= b')
print('Out of loop')
```

- Πώς θα γίνει η εκτέλεση του κώδικα και τι θα τυπώσει η print σε κάθε παράδειγμα;

While/else δομή επανάληψης

- 3

- Παραδείγματα

```
a=0; b=10
while a < b:
    print(a, end=' ')
    a += 1
    if a>8: break
else:
    print('a >= b')

print('Out of loop')
```

- Πώς θα γίνει τώρα η εκτέλεση του κώδικα και τι θα εμφανιστεί τελικά στην οθόνη;



While

Ερωτήσεις

- Γράψτε τον κώδικα που παράγει την έξοδο δεξιά

```
  * *
 * * *
* * * *
 * * * * *
  * * * * *
   * * * * *
    * * * * *
     * * * * *
      * * * * *
       * * * * *
        * * * * *
         * * * * *
          * * * * *
           * * * * *
```

```
          *
        * * *
      * * * *
    * * * * *
  * * * * *
 * * * * *
* * * * *
 * * * * *
  * * * * *
   * * * * *
    * * * * *
     * * * * *
      * * * * *
       * * * * *
```

```
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

- ...και τα παραπάνω δέντρα

While

Απαντήσεις

```
*
**
***
****
*****
*****
*****
*****
*****
*****
*****
```

```
lines = 1
maxlines=10
while lines <= maxlines:
    print(lines*'★')
    lines += 1
```

```
  **
 ***
****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```

```
lines=1
maxlines=10
while lines<=maxlines:
    print((maxlines-lines)*' ' +2*lines*'★')
    lines +=1
```

```
  *
 ***
****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```

```
lines=1
maxlines=10
while lines <= maxlines:
    print((maxlines-lines)*' ' + (2*lines-1)*'★')
    lines += 1
```

if /elif/else

δομή ελέγχου

- 1

```
if test1:                # Έλεγχος if
    statements1          # εκτελείται αν test1 Αληθής
elif test2:              # Ενσωματωμένο if (προαιρετικό)
    statements2          # εκτελείται αν test2 Αληθής
else:                    # Κλάδος else (προαιρετικός)
    statements3          # εκτελείται αν test1 Ψευδής
```

- Η **πλήρης** if/elif/else μπορεί να έχει **πολλούς** κλάδους elif (προαιρετικά) αλλά **έναν** κλάδο else (προαιρετικά)

if / elif / else

δομή ελέγχου

- 2

- Παραδείγματα

```
if 1:  
    print('true')
```

```
if not 1:  
    print('true')  
else:  
    print('false')
```

```
x = 'killer rabbit'  
if x == 'roger':  
    print("shave and a haircut")  
elif x == 'bugs':  
    print("what's up doc?")  
else:  
    print('Run away! Run away!')
```

- Πώς θα γίνει η εκτέλεση του κώδικα και τι θα τυπώσει η print σε κάθε παράδειγμα;

```
if choice == 'spam':  
    print(1.25)  
elif choice == 'ham':  
    print(1.99)  
elif choice == 'eggs':  
    print(0.99)  
elif choice == 'bacon':  
    print(1.10)  
else:  
    print('Bad choice')
```

break continue pass

```
while test-1:
    statements-1
    if test-2: break      # Έξοδος από το βρόχο while, η else αγνοείται
    if test-3: continue  # Επιστροφή στην αρχή του βροχου, επανάληψη statements-1
else:
    statements-2          # Εκτελείται 1 φορά εφόσον η έξοδος δεν έγινε από break
```

- Η break προκαλεί **άμεση έξοδο** από τον ‘πλησιέστερο’ (ή εσωτερικότερο) βρόχο
- Η continue οδηγεί άμεσα την εκτέλεση στην **αρχή** του βρόχου **παραλείποντας** άλλες εντολές παρακάτω (αν υπάρχουν)
- Η pass είναι απλά μια ‘**κενή**’ εντολή που μπαίνει για να τυπικά συντακτικά σωστός ο βρόχος -- μέχρι φυσικά να γραφούν οι τελικές εντολές

```
while True:
    pass                # Type Ctrl-C to stop me
```

While/If/Random

- Γράψτε κώδικα που να προσομοιώνει το ρίξιμο 2 ζαριών
- Όταν τρέχει ο κώδικας, κάθε φορά που ο χρήστης πατά Enter θα πρέπει να εμφανίζονται οι αριθμοί των 2 ζαριών
- Η εκτέλεση σταματά όταν πατηθεί το 'q'
 - (ενδεικτική εκτέλεση κώδικα δεξιά)

Άσκηση

```
>>>
2      5

4      5

1      3

1      1

5      2

1      6

6      3

6      2

q
>>> |
```