

Python

[About](#)[Downloads](#)[Documentation](#)[Community](#)[Success Stories](#)[News](#)[Events](#)

```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3 # classic division returns a float
5.666666666666667
>>> 17 // 3 # floor division
5
```



Intuitive Interpretation

Calculations are simple with Python, and expression syntax is straightforward: the operators `+`, `-`, `*` and `/` work as expected; parentheses `()` can be used for grouping. [More about simple math functions in Python 3.](#)

[1](#)[2](#)[3](#)[4](#)[5](#)

Διαφορές με την Γλώσσα της ΑΕΠΠ

- Είναι κανονική γλώσσα και μπορεί να εκτελεσθεί. Αυτό σημαίνει ότι ανά πάσα στιγμή ξέρουμε αν και πώς εκτελείται μια εντολή και τι αποτέλεσμα βγάζει
- Οι μεταβλητές δε δηλώνονται καθόλου
- Ισχύουν αριθμητικές πράξεις με string ('α' + 'β')
- Δεν υπάρχει ένδειξη ΑΡΧΗ ούτε ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
- Δεν υπάρχουν ΤΕΛΟΣ_ΑΝ, ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ (Οι εντολές που βρίσκονται μέσα σε κάποια δομή επιλογής ή δομή επανάληψης ΑΝ - ΓΙΑ - ΟΣΟ είναι αυτές που βρίσκονται στη σωστή στοίχιση (δύο θέσεις μέσα)
- Οι μεταβλητές αλλάζουν τύπο κατά τη διάρκεια της εκτέλεσης
- Ο Πίνακας (εδώ ονομάζεται λίστα) δε δηλώνεται και αυτός ως προς το μέγεθος, ενώ και τα στοιχεία του μπορεί να είναι διαφορετικού τύπου και να αλλάζουν κατά τη διάρκεια εκτέλεσης
- Η δημιουργία μιας λίστας (πίνακα) είναι κάτι διαφορετικό από το όνομά της. Η λίστα είναι ένα αντικείμενο στη μνήμη και το όνομα είναι ένας δείκτης που δείχνει σε αυτό (το αντικείμενο)
- Ο τελεστής εκχώρησης είναι το =
- Υπάρχουν έτοιμες συναρτήσεις ταξινόμησης, εύρεσης max min λίστας (πίνακα) κλπ
- Έχει σημασία σε ποιο σημείο γράφεται η εντολή. Στη θέση 1 αν είναι root, στη θέση 3 (δύο κενές) αν είναι μέσα σε for, while, if
- Δεν υπάρχει η δομή Αρχή_Επανάληψης Μέχρις_ότου
- Η συνάρτηση μπορεί να ορισθεί οπουδήποτε μέσα στο πρόγραμμα

Ιστορικά Στοιχεία

- Η ανάπτυξη της python άρχισε το Δεκέμβριο του 1989 στο Άμστερνταμ της Ολλανδίας από τον **Guido van Rossum** (Γκβίντο Βαν Ρόσουμ - Wikipedia), σε μια περίοδο διακοπών επειδή ήθελε να απασχοληθεί με κάτι !!
(υπάλληλος στο *National Research Institute for Mathematics and Computer Science*)
- Το όνομα είναι από τους Monty Pythons



Όπως ο ίδιος έγραψε το 1996:

Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix / Chackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).

Ιστορικά Στοιχεία

- Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού. Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της, η ευκολία χρήσης της και το συντακτικό της τα οποία επιτρέπουν στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα απ' ό τι θα έκαναν σε γλώσσες όπως η C++ και η Java.
Διαθέτει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα τις συνηθισμένες εργασίες και γενικά είναι εύκολη η εκμάθησή της.
- Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python να εκτελείται σε μια ευρεία γκάμα συστημάτων. Χρησιμοποιώντας εργαλεία τρίτων, όπως το Pyzexe ή το Pyinstaller



Looking for Python with a different OS? Python for [Windows](#),
[Linux/UNIX](#), [Mac OS X](#), [Other](#)

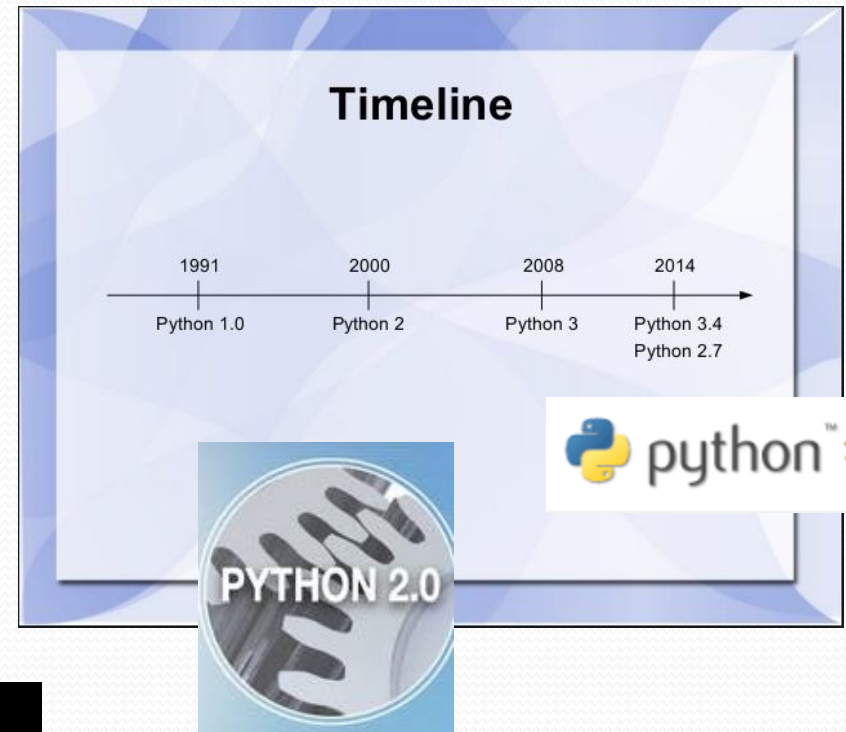


ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python.

Η Python αναπτύσσεται ως ανοιχτό λογισμικό (open source) και η διαχείρισή της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation.

Ιστορικά Στοιχεία

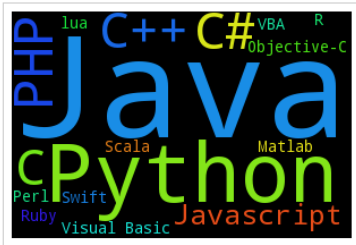
- Εκδόσεις :
 - **1.0** παρουσιάζεται επίσημα το 1991
 - **2.0** παρουσιάστηκε τον Οκτώβριο του 2000 και εγκαινιάζει την κοινότητα υποστήριξης της γλώσσας
 - **3.0** ή py3k παρουσιάστηκε τον Δεκέμβριο του 2008 και είναι μη συμβατή με τις προηγούμενες εκδόσεις. (αργότερα 2014 - πολλά χαρακτηριστικά της μεταφέρθηκαν στην 2.7).



Πόσο δημοφιλής είναι

<http://pypl.github.io/PYPL.html>

Click here : [PYPL Popularity of Programming Language Index](#)



The PYPL Popularity of Programming Language Index is created by analyzing how often language tutorials are searched on Google.

The more a language tutorial is searched, the more popular the language is assumed to be. It is a leading indicator. The raw data comes from Google Trends.

If you believe in collective wisdom, the PYPL Popularity of Programming Language index can help you decide which language to study, or which one to use in a new software project.

- [PYPL index for US](#)
- [PYPL index for India](#)
- [PYPL index for Germany](#)
- [PYPL index for France](#)
- [PYPL index for United Kingdom](#)

US, Jan 2016 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.6 %	+0.5 %
2		Python	17.6 %	+1.5 %
3	↑	Javascript	8.2 %	+0.9 %
4	↑↑	C#	8.0 %	+1.0 %
5	↓↓	C++	7.9 %	-0.7 %
6	↓	Objective-C	6.0 %	-1.5 %
7		C	6.0 %	-0.6 %
8		PHP	5.0 %	-0.4 %

Germany, Jan 2016 compared to a year ago:

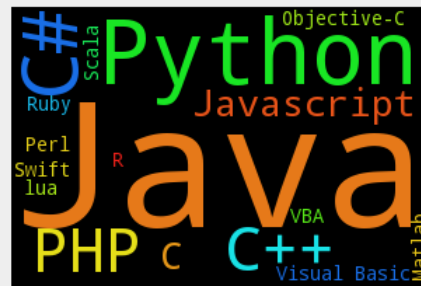
Rank	Change	Language	Share	Trend
1		Java	27.0 %	+0.8 %
2		Python	16.2 %	+2.4 %
3	↑↑	C#	11.9 %	+1.2 %
4	↓	C++	11.7 %	-1.5 %
5	↓	PHP	11.1 %	-0.8 %
6		Javascript	8.8 %	+2.2 %
7		C	7.9 %	+0.9 %
8		Objective-C	5.4 %	-0.6 %

© Pierre Carboneille, 2015

France, Jan 2016 compared to a year ago:

Rank	Change	Language	Share	T
1		Java	45.0 %	+
2		Python	37.7 %	+19.1 %
3		PHP	17.3 %	-6.8 %

© Pierre Carboneille, 2015



British Indian Ocean Territory



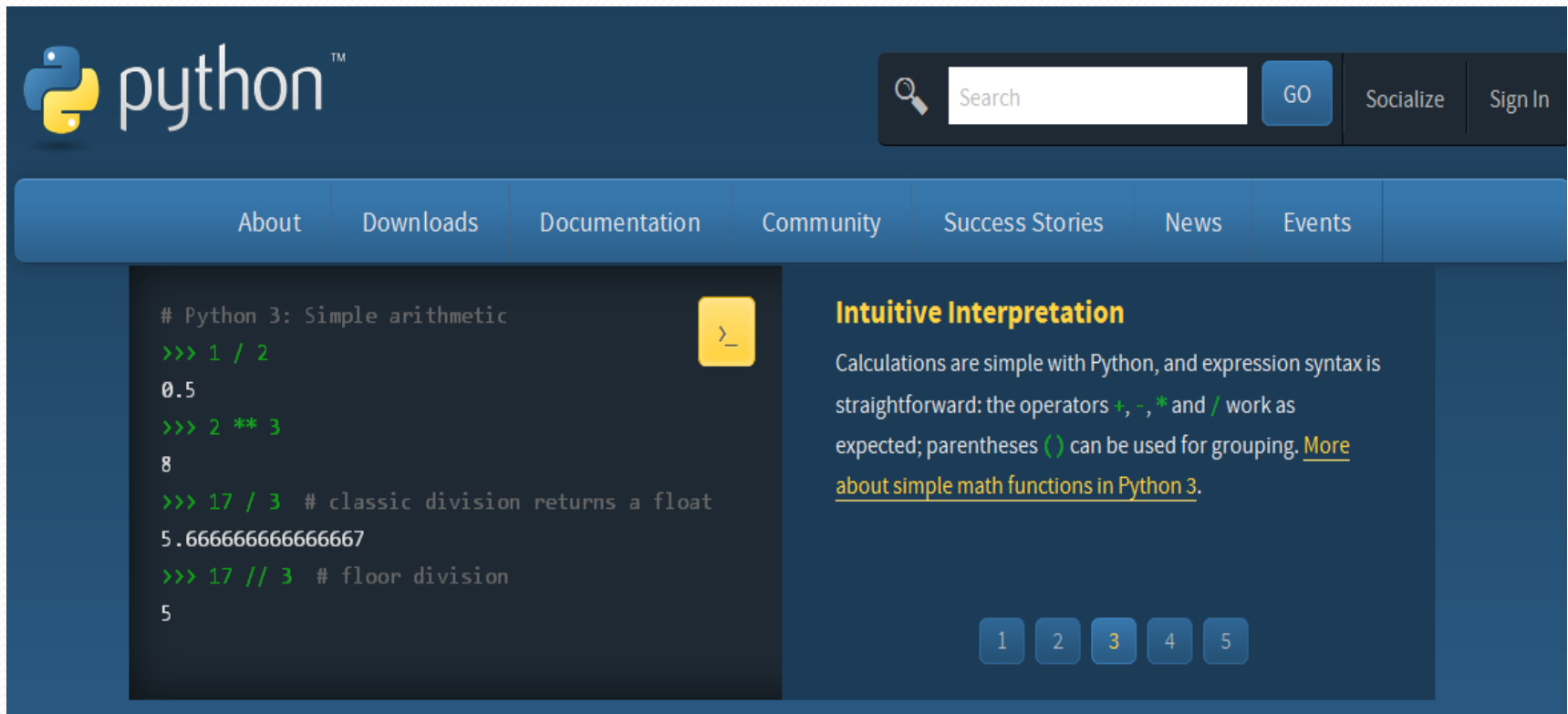
Flag



Coat of arms

Δικτυακός τόπος

<https://www.python.org/>



The screenshot shows the Python.org website interface. At the top left is the Python logo and the word "python" with a trademark symbol. To the right is a search bar with a magnifying glass icon, a "GO" button, and links for "Socialize" and "Sign In". Below this is a navigation menu with links for "About", "Downloads", "Documentation", "Community", "Success Stories", "News", and "Events". The main content area features a code snippet on the left and a text block on the right. The code snippet is titled "# Python 3: Simple arithmetic" and shows several arithmetic operations. The text block is titled "Intuitive Interpretation" and explains that calculations are simple with Python, with operators +, -, *, and / working as expected, and parentheses () used for grouping. A link is provided for "More about simple math functions in Python 3." At the bottom right of the text block are five numbered buttons (1-5).

```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3 # classic division returns a float
5.666666666666667
>>> 17 // 3 # floor division
5
```

Intuitive Interpretation

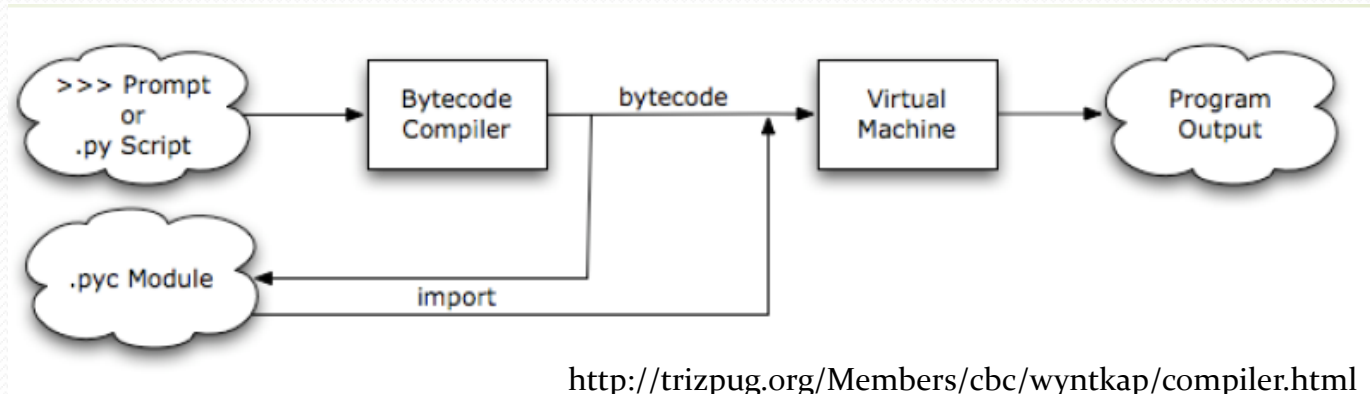
Calculations are simple with Python, and expression syntax is straightforward: the operators `+`, `-`, `*` and `/` work as expected; parentheses `()` can be used for grouping. [More about simple math functions in Python 3.](#)

1 2 3 4 5

Χρήσιμα Εργαλεία

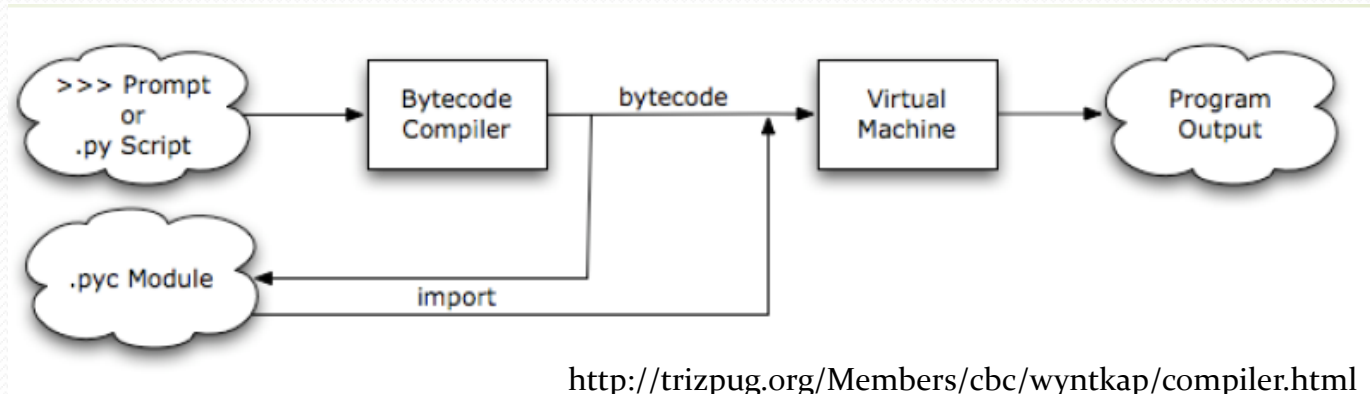
- PyCharmCommunity edition 5.0.1
- PyCharmEdu2.0.1
- <http://www.pythontutor.com/>
- <http://interactivepython.org/runestone/static/pythonds/index.html>

Είναι η Python Interpreted ? ή Compiled ?



- Δεν είναι και τόσο ξεκάθαρα σαφές εάν η Python είναι μια γλώσσα που διερμηνεύεται ή μεταγλωττίζεται. Στην πραγματικότητα, όταν εκτελείται ένας κώδικας από τη γραμμή εντολών (command line) ή μέσω ενός πηγαίου αρχείου, δημιουργεί ένα ενδιάμεσο αρχείο .pyc το οποίο είναι σε bytecode (μια γλώσσα μεταξύ assembly και γλώσσας μηχανής) ο οποίος bytecode, εκτελείται από μια εικονική μηχανή. Έτσι θα μπορούσαμε να πούμε ότι έχει συμπεριφορά μεταγλωττιζόμενης γλώσσας, όχι όμως ξεκάθαρη, αφού η μεταγλώττιση απαιτεί μετάφραση του πηγαίου κώδικα σε κάποιο ενδιάμεσο αρχείο γλώσσας μηχανής.
- Επιπλέον, όταν εκτελείται ένα py script από τη γραμμή εντολών (./somefile.py) δε δημιουργείται κανένα ενδιάμεσο αρχείο και δείχνει ως καθαρά διερμηνευόμενη γλώσσα.
- Στο διαδίκτυο υπάρχει μεγάλη συζήτηση και οι απόψεις δίστανται.

Το μοντέλο εκτέλεσης της Python



- Η τυπική διαδικασία είναι αυτή που φαίνεται στο παραπάνω σχήμα.
- Ο πηγαίος κώδικας αποθηκεύεται σε αρχείο με επέκταση **.py**
- Αρχικά ο κώδικας μεταφράζεται (compiled) σε μορφή **“bytecode”**. **.pyc** στο φάκελο **__pycache__** (εκεί που είναι τα αρχεία .py)
- Στη συνέχεια εκτελείται από μία “εικονική μηχανή” (**PVM**).

Το μοντέλο εκτέλεσης της Python

- Ο κώδικας εκτελείται αμέσως μόλις τον γράψετε.
 - Δεν υπάρχει κάποια χρονοβόρα φάση όπως πχ. build, make, link κλπ.
- Ο bytecode ΔΕΝ είναι δυαδικός κώδικας μηχανής αλλά είναι μια ενδιάμεση αναπαράσταση που δημιουργεί η Python
- Κατά την εκτέλεση η PVM εκτελεί τον bytecode
- Το τελικό αποτέλεσμα είναι πως ο κώδικας Python μπορεί να εκτελείται με ταχύτητες μεταξύ της ταχύτητας μιας καθαρά μεταφραζόμενης (compiled) γλώσσας και μιας καθαρά διερμηνευόμενης (interpreted) γλώσσας

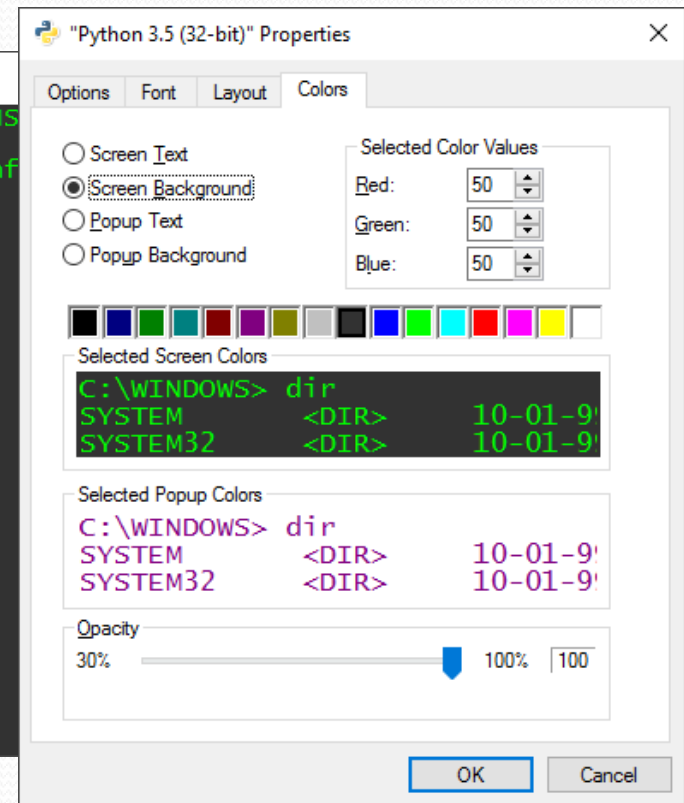
Το περιβάλλον εργασίας της Python



- **Python shell.** Στα win 8 βρίσκεται στη διαδρομή
- `C:\Users\aaa\AppData\Local\Programs\Python\Python35-32\python.exe`

A screenshot of the Python 3.5 (32-bit) shell. The window title is "Python 3.5 (32-bit)". The text inside the shell is green on a black background. It shows the version information: "Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MS tel)] on win32" and a prompt "Type 'help', 'copyright', 'credits' or 'license' for more info". Below that is a prompt ">>>" followed by a cursor and a space character.

```
Python 3.5 (32-bit)
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MS
tel)] on win32
Type "help", "copyright", "credits" or "license" for more inf
>>> _
```



Τελεστές

- Αριθμητικοί

- + - * / **//** **%**
- ** (δύναμη)
- +, * και για τα strings

```
>>> 2+3
5
>>> 2-3
-1
>>> 2*3
6
>>> 2/3
0.6666666666666666
>>> 2//3
0
>>> 5//3
1
>>> 5%3
2
>>> 2 * "sos"
'sossos'
>>> 3 * "sos"
'sossossos'
>>> 'a' + 'b'
'ab'
>>> 'a' - 'b'
Traceback (most recent call
  File "<stdin>", line 1, in
TypeError: unsupported opera
>>>
```

- Συγκριτικοί

- ==, !=, <, >, >=, <=

- Λογικοί

- and, or, not

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

Εκχώρηση τιμής με =
(αντί για το ← της ΑΕΠΠ)

Τελεστές

- Αριθμητικοί
 - συντομεύσεις πράξεων

`b = b + 1`

`b = b * 2`

`b = b - 3`

```
>>>
>>> b = 2
>>> b += 1
>>> b
3
>>> b *= 2
>>> b
6
>>> b -= 3
>>> b
3
>>>
```

Τύποι Δεδομένων

- Αριθμοί (Numbers)

- Ακέραιοι (integers)

- Η Python δεσμεύει δυναμικά τον απαραίτητο χώρο μνήμης για την αναπαράσταση οσοδήποτε μεγάλων ακεραίων

- Στην Python 3.x οι ακέραιοι είναι θεωρητικά άπειρης ακρίβειας

- Πολλαπλή ανάθεση τιμών

```
Python 3.5 (32-bit)
>>> a, b, c = 1, 2, 3
>>> a
1
>>> b
2
>>> c
3
>>> _
```

```
Python 3.5 (32-bit)
>>>
>>> 15+32
47
>>> 2.5*5
12.5
>>> 2**3
8
>>> len(str(2**1000000))
301030
>>> 16/3
5.333333333333333
>>> 16//3
5
>>> 16%3
1
>>>
```

Συνάρτηση `len`,
η οποία δίνει το
μήκος ενός
string
(αργεί
ο υπολογισμός)

Python Clear screen

```
Python 3.5 (32-bit)  
>>>  
>>>  
>>> import os  
>>> os.system('cls')
```

Ενσωμάτωση
βιβλιοθήκης **OS**
και κλήση της
συνάρτησης **cls**

Τύποι Δεδομένων

- Αριθμοί (Numbers)
 - Κινητής υποδιαστολής (πραγματικοί –float)
 - μαθηματικές σταθερές (πχ. 'π') και συναρτήσεις είναι διαθέσιμες μέσω της βιβλιοθήκης **math** ή τρίτων πχ. **SciPy**

Ενσωμάτωση
βιβλιοθήκης
math

και κλήση των
συναρτήσεων
sqrt και **pi**

```
Python 3.5 (32-bit)
0
>>> 3.1415 * 2
6.283
>>> 34.78 + 95.25
130.03
>>>
>>> sqrt(16)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
>>>
>>> import math
>>>
>>> math.sqrt(16)
4.0
>>>
>>> math.pi
3.141592653589793
>>>
```

Τύποι Δεδομένων

- Αριθμοί (Numbers)
 - Άλλοι τύποι όπως:
 - Μιγαδικοί (complex)
 - Δεκαδικοί καθορισμένης ακρίβειας (fixed precision)
 - Κλασματικοί (Fractions implements a rational number object)
 - Σύνολα αριθμών (sets)

```
>>> complex (2,-3)
(2-3j)
>>>
```

```
Python 3.5 (32-bit)
0
>>> from fractions import Fraction
>>> Fraction (16, -10)
Fraction(-8, 5)
>>> Fraction (123)
Fraction(123, 1)
>>> a = Fraction (6,10)
>>> a
Fraction(3, 5)
>>> b = a + Fraction (1,2)
>>> b
Fraction(11, 10)
>>> Fraction(' -3/7 ')
Fraction(-3, 7)
>>> Fraction('-.125')
Fraction(-1, 8)
>>> Fraction('7e-6')
Fraction(7, 1000000)
>>>
```

Τύποι Δεδομένων

```
Python 3.5 (32-bit)
0
>>> s1 = 'Καλημέρα'
>>> s2 = ' σε όλους'
>>> s = s1 + s2
>>> s
'Καλημέρα σε όλους'
>>>
```

- Αλφαριθμητικά (String)
 - Χρησιμοποιούνται για την αποθήκευση **κειμένου** (πχ. ονόματα) ή σαν **συλλογή bytes** (πχ. περιεχόμενα αρχείου)
 - Περικλείονται σε **μονά** ή **διπλά** εισαγωγικά
 - Τελεστής συνένωσης (concatenation) +

Τύποι Δεδομένων

- Αλφαριθμητικά (String)
το string λειτουργεί ως πίνακας. Πρώτο στοιχείο το 0 και τελευταίο το N-1
Στο παράδειγμα της εικόνας μήκος string είναι 17. πρώτο στοιχείο το 0, τελευταίο το 16. Το 17 δεν υπάρχει

```
Python 3.5 (32-bit)
0
>>> s = 'Καλημέρα σε όλους'
>>> len(s)
17
>>> s[0]
'Κ'
>>> s[1]
'α'
>>> s[16]
'ς'
>>> s[17]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> _
```

Τύποι Δεδομένων

- Λογικές (Boolean) εισήχθηκαν στην python από την έκδοση 2.3 και μετά.

Δεν υπάρχει η λογική
 $\alpha = \text{True}$

Υπάρχει όμως η λογική
`if $\alpha == \text{True}$:`

```
>>> a=bool(1)
>>> a
True
>>> b=bool(0)
>>> b
False
>>> a and b
False
>>> a or b
True
```

Προσοχή **True** και όχι **true**
(case sensitive)

Άλλοι Τύποι Δεδομένων

- Λίστες (lists)
- Λεξικά(dictionaries)
- Πλειάδες (tuples)
- Αρχεία(files)
- Σύνολα (sets)
- None (is one way to reset one parameter to its original, empty state)

Ενσωματωμένες συναρτήσεις

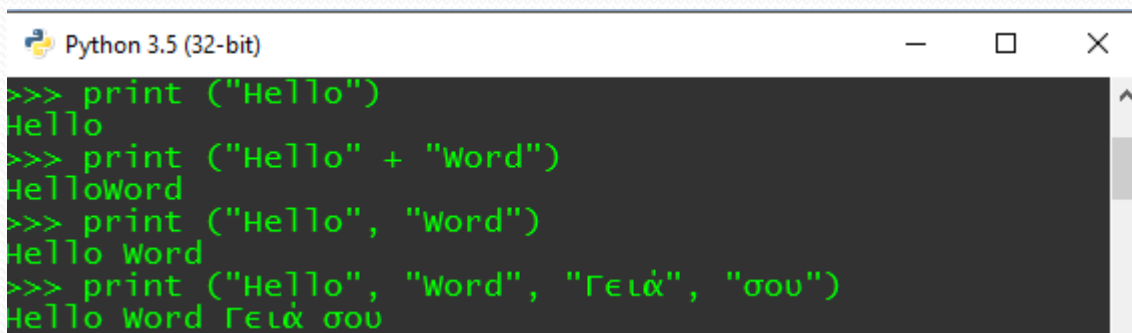
Built-in Functions				
<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

<https://docs.python.org/>

```
>>> help(__builtins__)
```

Είσοδος & Έξοδος

- `print(objects, sep=' ', end='\n', file=sys.stdout, flush=False)`



```
Python 3.5 (32-bit)
>>> print ("Hello")
Hello
>>> print ("Hello" + "Word")
HelloWord
>>> print ("Hello", "Word")
Hello Word
>>> print ("Hello", "Word", "Γειά", "σου")
Hello Word Γειά σου
```

Object: λίστα παραμέτρων για εμφάνιση

Sep: αλφαριθμητικό διαχωριστής μεταξύ των παραμέτρων

End: αλφαριθμητικό που καθορίζει τον τρόπο μετακίνησης του δείκτη μετά το τέλος της print

File: η μονάδα στην οποία κατευθύνεται η έξοδος της print

sys.stdout είναι η εξ ορισμού έξοδος ('κονσόλα')

Flush: λογική παράμετρος που καθορίζει αν η έξοδος που παράγει θα εμφανιστεί άμεσα στη μονάδα εξόδου χωρίς buffering

\ (backslash): επέκταση εντολής σε περισσότερες γραμμές

Είσοδος & Έξοδος

προαιρετικό

- μεταβλητή = `input('μήνυμα')`
- περιμένει μέχρι να πατηθεί Enter
- διαβάζει τα δεδομένα εισόδου ως **αλφαριθμητικό**

```
>>> ans = input('Τι κατοικίδιο έχετε ; Σκύλο (σ) ή Γάτα (γ)')
Τι κατοικίδιο έχετε ; Σκύλο (σ) ή Γάτα (γ)σ
>>> ans
'σ'
```

- Συνήθως περιλαμβάνει μήνυμα προς τον χρήστη
- Ένα συνηθισμένο λάθος για τους αρχάριους είναι να μην μετατρέπουν τον τύπο της `input()`
- Χρήσιμες συναρτήσεις μετατροπής **`int()`**, **`float()`**, **`str()`**

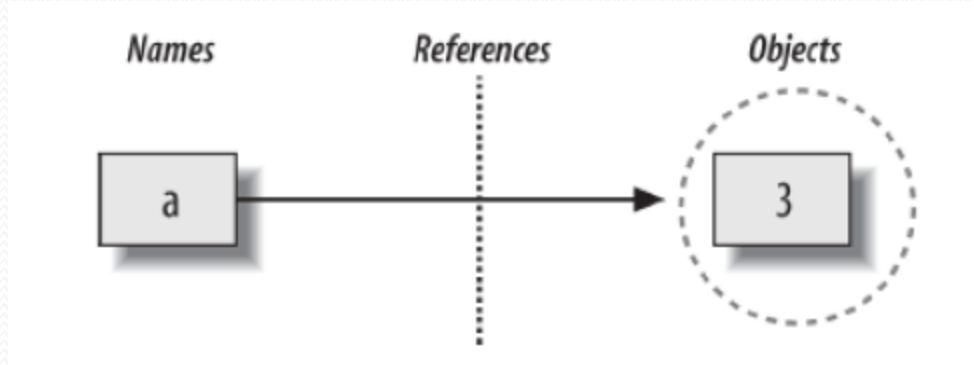
```
Python 3.5 (32-bit)
0
>>> input(b)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'b' is not defined
>>> b=input()
23
>>> b
'23'
>>> a=input()
32
>>> a+b
'3223'
>>> int(a)+int(b)
55
>>> _
```

Μεταβλητές - Ονοματολογία

- Στην Python μια μεταβλητή είναι ένα όνομα για ένα αντικείμενο
- `_spam`, `spam`, `Spam_1`: αποδεκτά ονόματα
- `1_Spam`, `spam$`, `@#!`: μη αποδεκτά ονόματα
- Τα ονόματα είναι **'Case sensitive'**
- Οι ελληνικοί χαρακτήρες επιτρέπονται και η **μεταβλητή α** είναι διαφορετική από τη **μεταβλητή a**
- Υπάρχουν **δεσμευμένες λέξεις** με **διπλό underscore** στην αρχή και το τέλος που έχουν ιδιαίτερη σημασία

Μεταβλητές

- Τι συμβαίνει όταν γράφουμε: $\alpha = 3$
- Δημιουργείται στη μνήμη ένα αντικείμενο τύπου `int` με τιμή 3
- Το όνομα 'α' συνδέεται (*binding*) με το αντικείμενο που έχει τιμή '3'
- Η αναφορά υλοποιείται με τη μορφή ενός δείκτη στη μνήμη



Μεταβλητές

- $\alpha = 3$

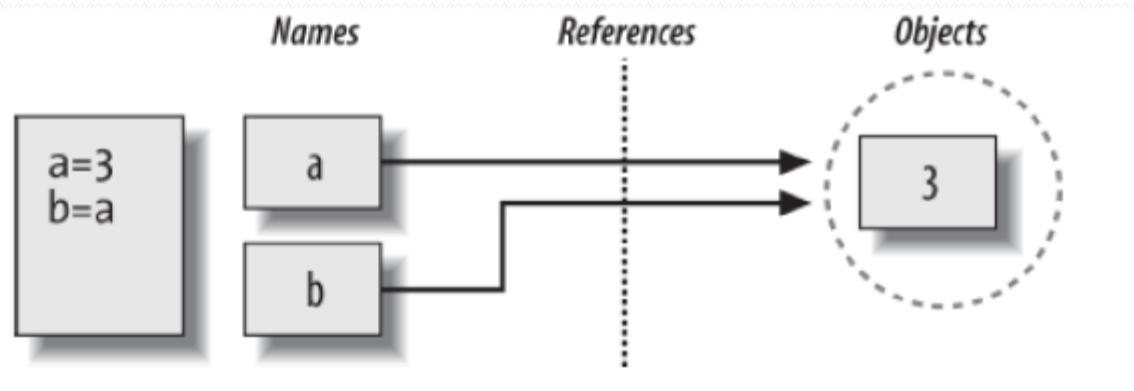
CPython: This is the address of the object in memory

- $b = \alpha$

- Στην περίπτωση αυτή και τα δύο ονόματα 'δείχνουν' προς το ίδιο αντικείμενο (ίδιο identification – id())

- "διαμοιρασμένη αναφορά" (shared reference)

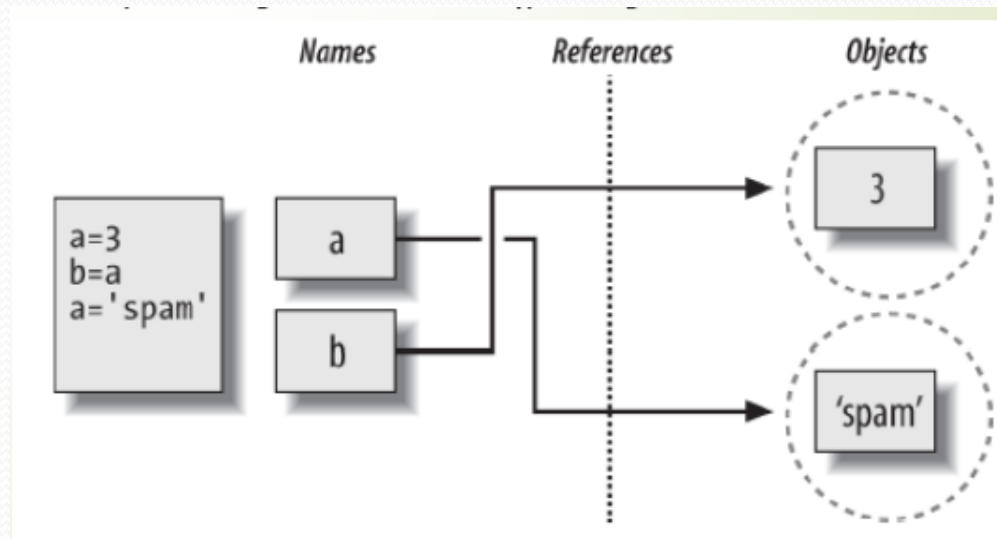
```
Python 3.5 (32-bit)
0
>>> α = 3
>>> id(α)
1734752048
>>> b = α
>>> b
3
>>> id(b)
1734752048
>>> _
```



Μεταβλητές

- $\alpha = 3$
- $b = a$
- $a = \text{'spam'}$
- Ανακατεύθυνση του ονόματος a στο αλφαριθμητικό αντικείμενο με τιμή 'spam'
- Είναι προφανές ότι το `identification - id()` ανήκει στο **Αντικείμενο - Object** και όχι στη μεταβλητή
- Η αναφορά του ονόματος b δεν επηρεάζεται

```
Python 3.5 (32-bit)
0
>>> α = 3
>>> id(α)
1734752048
>>> b = α
>>> α = 'spam'
>>> id(b)
1734752048
>>> id(α)
19785760
>>>
```



Μεταβλητές

- Τα ονόματα των μεταβλητών καταλαμβάνουν τη δική τους περιοχή στη μνήμη (**namespace**) και αποτελούν αναφορές προς τα αντικείμενα με τα οποία είναι διασυνδεδεμένα
- Τα αντικείμενα δεσμεύουν τον απαραίτητο χώρο στη μνήμη για αποθήκευση των δεδομένων που τα συνθέτουν
- Οι αναφορές είναι δείκτες που δημιουργούνται αυτόματα κατά τη στιγμή του ορισμού της αντιστοίχισης της μεταβλητής προς το αντικείμενο **με την εντολή ανάθεσης** (πχ `a=3`)

Ανάθεση Τιμών

- Μια εντολή ανάθεσης δημιουργεί αναφορά σε αντικείμενα στη μνήμη
- Τα ονόματα μεταβλητών δημιουργούνται την πρώτη φορά που θα γίνει ανάθεση τιμής στο όνομα αυτό
- Πριν χρησιμοποιηθεί ένα όνομα μεταβλητής σε μία έκφραση πρέπει να έχει αποκτήσει τιμή (αλλιώς κατά τα γνωστά θα έχουμε παραβίαση της αποτελεσματικότητας σύμφωνα με την ΑΕΠΠ)
- Ορισμένες λειτουργίες εκτελούν αναθέσεις τιμής σιωπηρά (πχ. στη μεταβλητή ενός βρόχου)

Ανάθεση Τιμών

Operation

```
spam = 'Spam'
```

```
spam, ham = 'yum', 'YUM'
```

```
[spam, ham] = ['yum', 'YUM']
```

```
a, b, c, d = 'spam'
```

```
a, *b = 'spam'
```

```
spam = ham = 'lunch'
```

```
spams += 42
```

Python 3.5 (32-bit)

```
0
>>> spam = 'Hello'
>>> spam
'Hello'
>>> spam = "Hello"
>>> spam
'Hello'
>>> spam, ham = 'yum', 'YUM'
>>> spam
'yum'
>>> ham
'YUM'
>>> [spam, ham] = ['yum', 'YUM']
>>> spam
'yum'
>>> ham
'YUM'
>>> a,b,c,d = 'spam'
>>> a
's'
>>> b
'p'
>>> c
'a'
>>> d
'm'
>>> _
```

Ανάθεση Τιμών

Operation

```
spam = 'Spam'
```

```
spam, ham = 'yum', 'YUM'
```

```
[spam, ham] = ['yum', 'YUM']
```

```
a, b, c, d = 'spam'
```

```
a, *b = 'spam'
```

```
spam = ham = 'lunch'
```

```
spams += 42
```

Python 3.5 (32-bit)

```
0
>>> a,b = 'Hello'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack (expected 2)
>>> a,b = 'He'
>>> a
'H'
>>> b
'e'
>>> a,*b = 'Hello'
>>> a
'H'
>>> b
['e', 'l', 'l', 'o']
>>> _
```

Ανάθεση Τιμών

Operation

```
spam = 'Spam'
```

```
spam, ham = 'yum', 'YUM'
```


```
[spam, ham] = ['yum', 'YUM']
```

```
a, b, c, d = 'spam'
```

```
a, *b = 'spam'
```

```
spam = ham = 'lunch'
```

```
spams += 42
```

 Python 3.5 (32-bit)

```
0
>>> spam = ham = 'lunch'
>>> spam
'lunch'
>>> ham
'lunch'
>>> spams = 3
>>> spams += 42
>>> spams
45
>>> (δηλαδή spams = spams + 42)
```

Ανάθεση Τιμών

```
>>> a, b, c, d = spam = foo, *boo = [10, 20, 'ham', 'eggs']
```

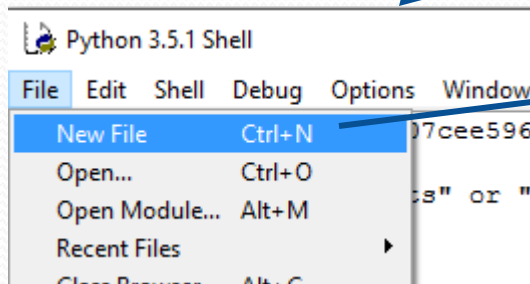
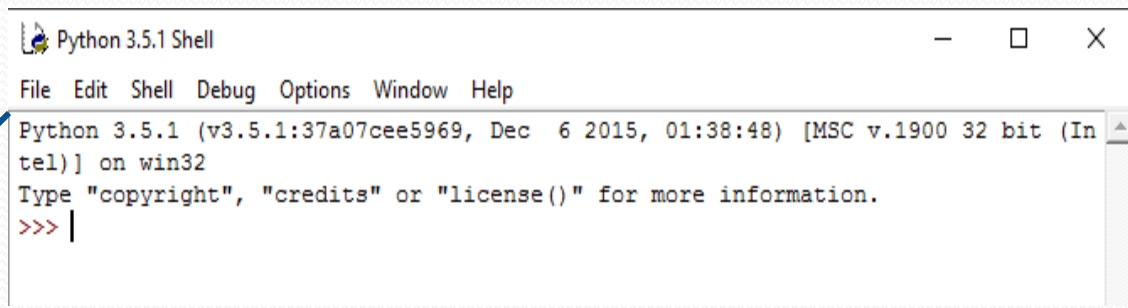
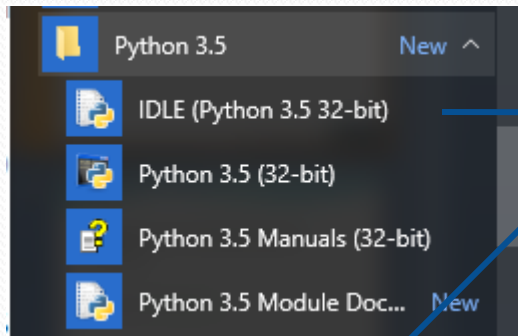
Πως γίνεται η αντιμετάθεση τιμών **ΛΙΣΤΑΣ**

```
Python 3.5 (32-bit)
0
>>> boo = [10, 20, 'ham', 'eggs']
>>> boo
[10, 20, 'ham', 'eggs']
>>>
>>> a,b,c,d = spam = boo
>>> a
10
>>> b
20
>>> c
'ham'
>>> d
'eggs'
>>> spam
[10, 20, 'ham', 'eggs']
>>>
```

Περιβάλλον Python IDLE



Έρχεται προεγκατεστημένο και αποτελεί ένα περιβάλλον εργασίας με την python το οποίο περιλαμβάνει και το shell. Αλλά και έναν editor από τον οποίο μπορεί να γίνει και εκτέλεση (run) των προγραμμάτων.



Περιβάλλον Python IDLE

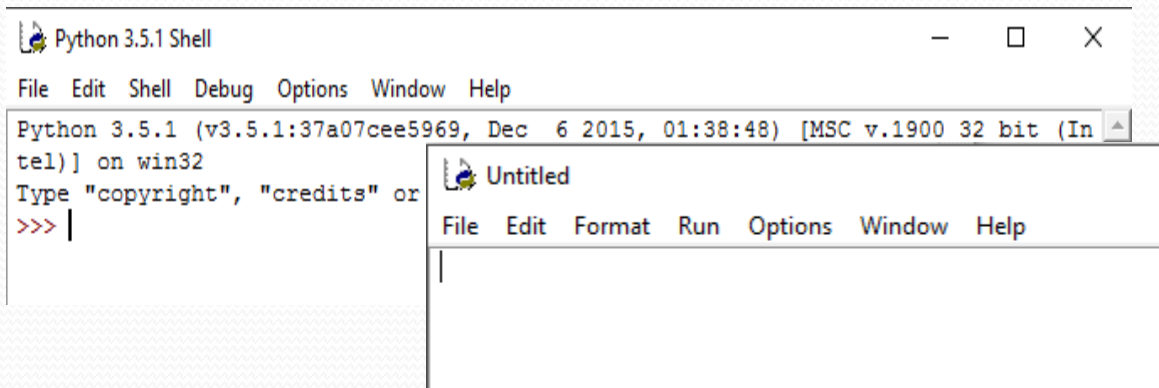


Μειονέκτημα :

- 1) Δε δουλεύει το history των εντολών με το πάνω βελάκι στο shell
- 2) Στον editor δε δουλεύει το copy paste όταν είναι γυρισμένο στα Ελληνικά

Πλεονέκτημα :

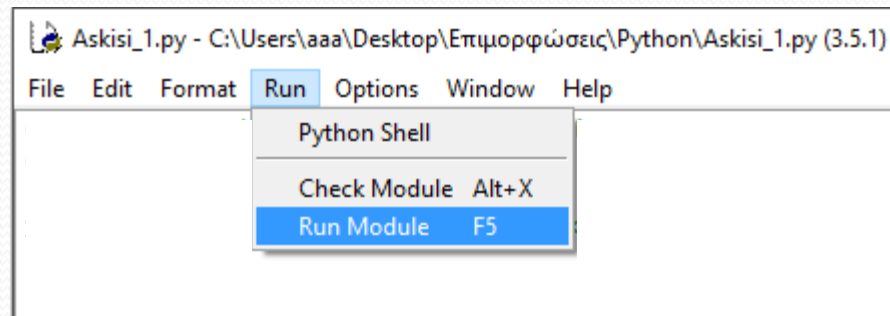
- 1) Χρωματισμός των εντολών και των δεσμευμένων λέξεων
- 2) Δυνατότητα εκτέλεσης των προγραμμάτων με το πάτημα του «run»



The screenshot shows two overlapping windows from the Python IDLE environment. The top window is titled 'Python 3.5.1 Shell' and contains the following text: 'Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32', 'Type "copyright", "credits" or "help()" to see more help text.', and a prompt '>>> |'. The bottom window is titled 'Untitled' and is currently empty, showing only a cursor at the beginning of the line. Both windows have standard Windows-style title bars and menus.

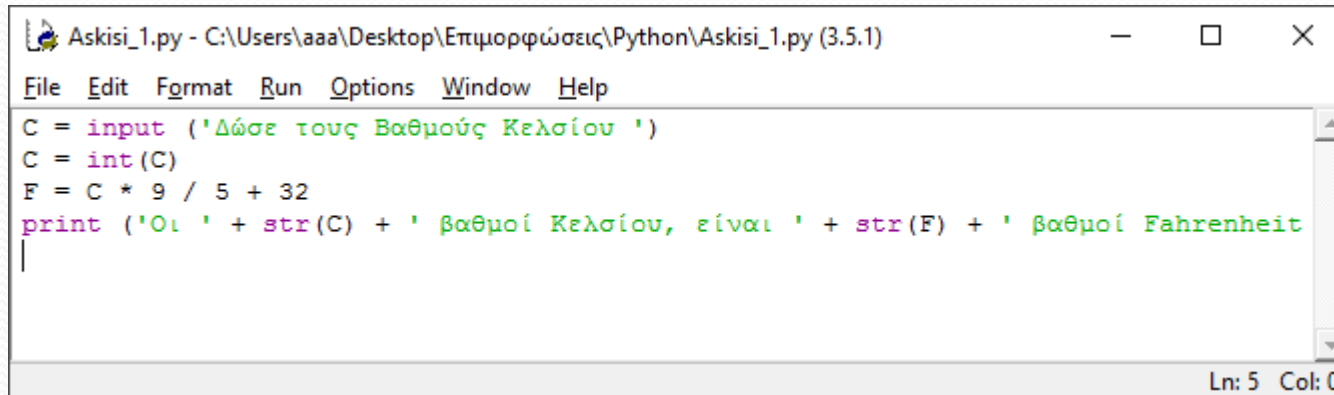
Άσκηση 1

- Μετατροπή βαθμών Κελσίου σε Fahrenheit εμφανίζοντας κατάλληλα μηνύματα
- $T_{(^{\circ}\text{F})} = T_{(^{\circ}\text{C})} * 9/5 + 32$

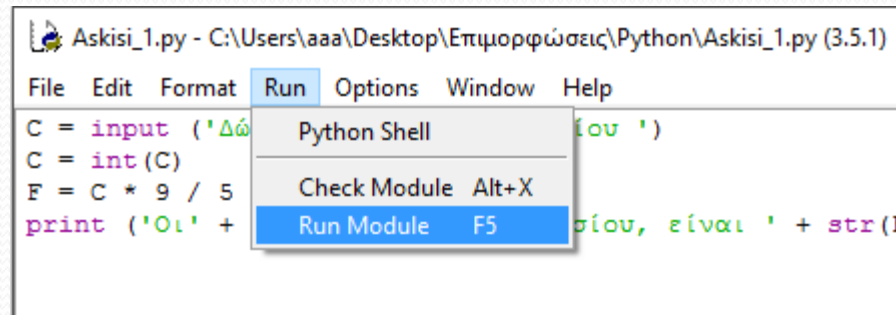


Άσκηση 1

- Μετατροπή βαθμών Κελσίου σε Fahrenheit εμφανίζοντας κατάλληλα μηνύματα
- $T_{(^{\circ}F)} = T_{(^{\circ}C)} * 9/5 + 32$



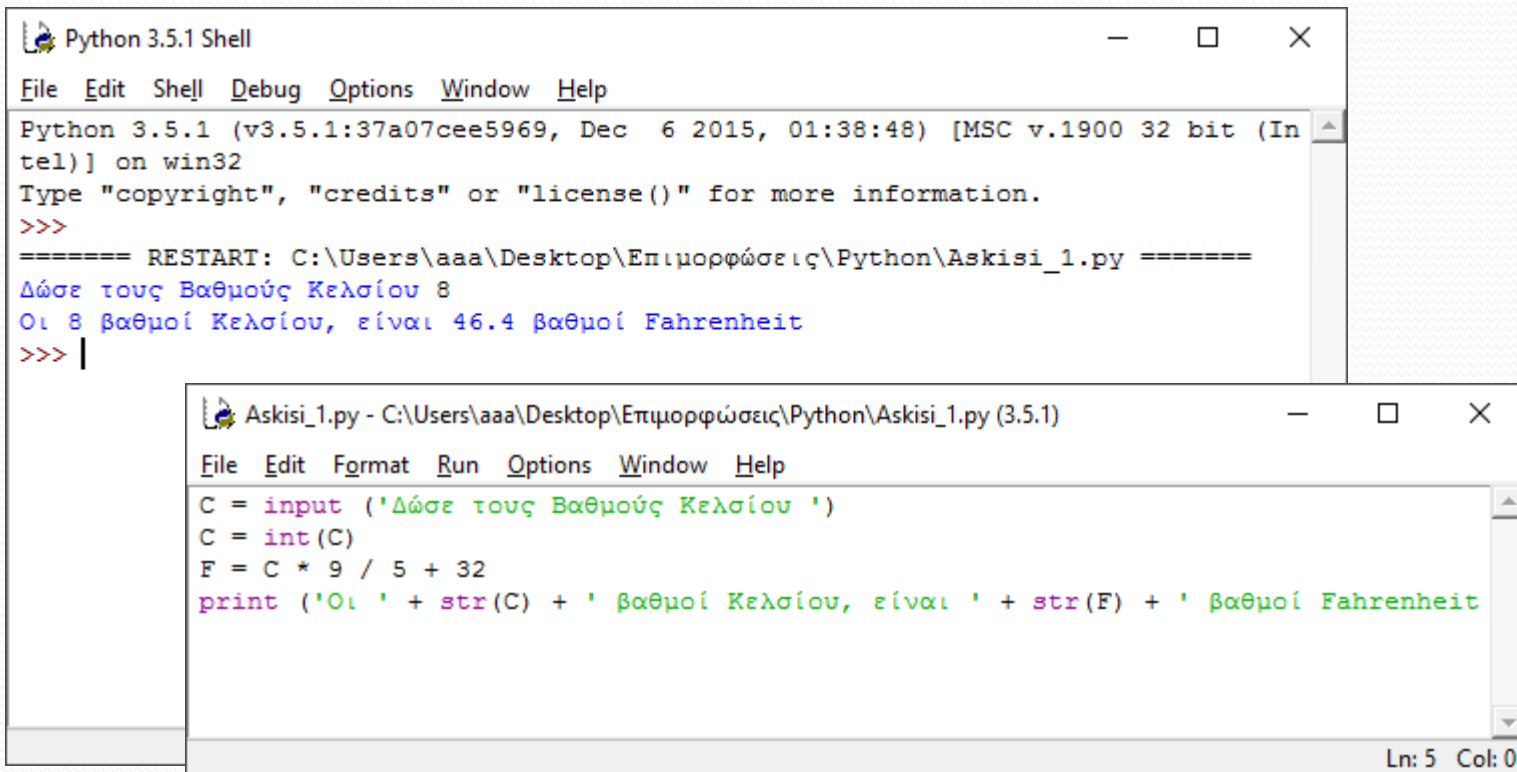
```
Askisi_1.py - C:\Users\aaa\Desktop\Επιμορφώσεις\Python\Askisi_1.py (3.5.1)
File Edit Format Run Options Window Help
C = input ('Δώσε τους Βαθμούς Κελσίου ')
C = int(C)
F = C * 9 / 5 + 32
print ('Οι ' + str(C) + ' βαθμοί Κελσίου, είναι ' + str(F) + ' βαθμοί Fahrenheit
|
Ln: 5 Col: 0
```



```
Askisi_1.py - C:\Users\aaa\Desktop\Επιμορφώσεις\Python\Askisi_1.py (3.5.1)
File Edit Format Run Options Window Help
C = input ('Δώ
C = int(C)
F = C * 9 / 5
print ('Οι' +
Python Shell
Check Module Alt+X
Run Module F5
ίου ' )
σίου, είναι ' + str(F
```

Άσκηση 1

...και η εκτέλεση...



The image shows two overlapping windows from a Windows environment. The top window is titled "Python 3.5.1 Shell" and displays the following text:

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\aaa\Desktop\Επιμορφώσεις\Python\Askisi_1.py =====
Δώσε τους Βαθμούς Κελσίου 8
Οι 8 βαθμοί Κελσίου, είναι 46.4 βαθμοί Fahrenheit
>>> |
```

The bottom window is titled "Askisi_1.py - C:\Users\aaa\Desktop\Επιμορφώσεις\Python\Askisi_1.py (3.5.1)" and displays the following Python code:

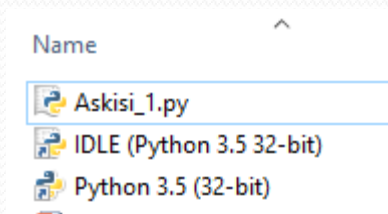
```
File Edit Format Run Options Window Help
C = input ('Δώσε τους Βαθμούς Κελσίου ')
C = int(C)
F = C * 9 / 5 + 32
print ('Οι ' + str(C) + ' βαθμοί Κελσίου, είναι ' + str(F) + ' βαθμοί Fahrenheit
```

The status bar at the bottom right of the second window shows "Ln: 5 Col: 0".

Άσκηση 1

- Μετατροπή βαθμών Κελσίου σε Fahrenheit εμφανίζοντας κατάλληλα μηνύματα
- $T_{(^{\circ}F)} = T_{(^{\circ}C)} * 9/5 + 32$

Επίσης Εκτέλεση κατευθείαν με διπλό κλικ στο όνομα της Άσκησης από τον explorer



Έλεγχος της ροής του προγράμματος – Εντολή if

- Η πλήρης **if/elif/else** μπορεί να έχει πολλούς κλάδους

elif (προαιρετικά όσα θέλουμε) αλλά έναν κλάδο else (προαιρετικά)

```
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
```

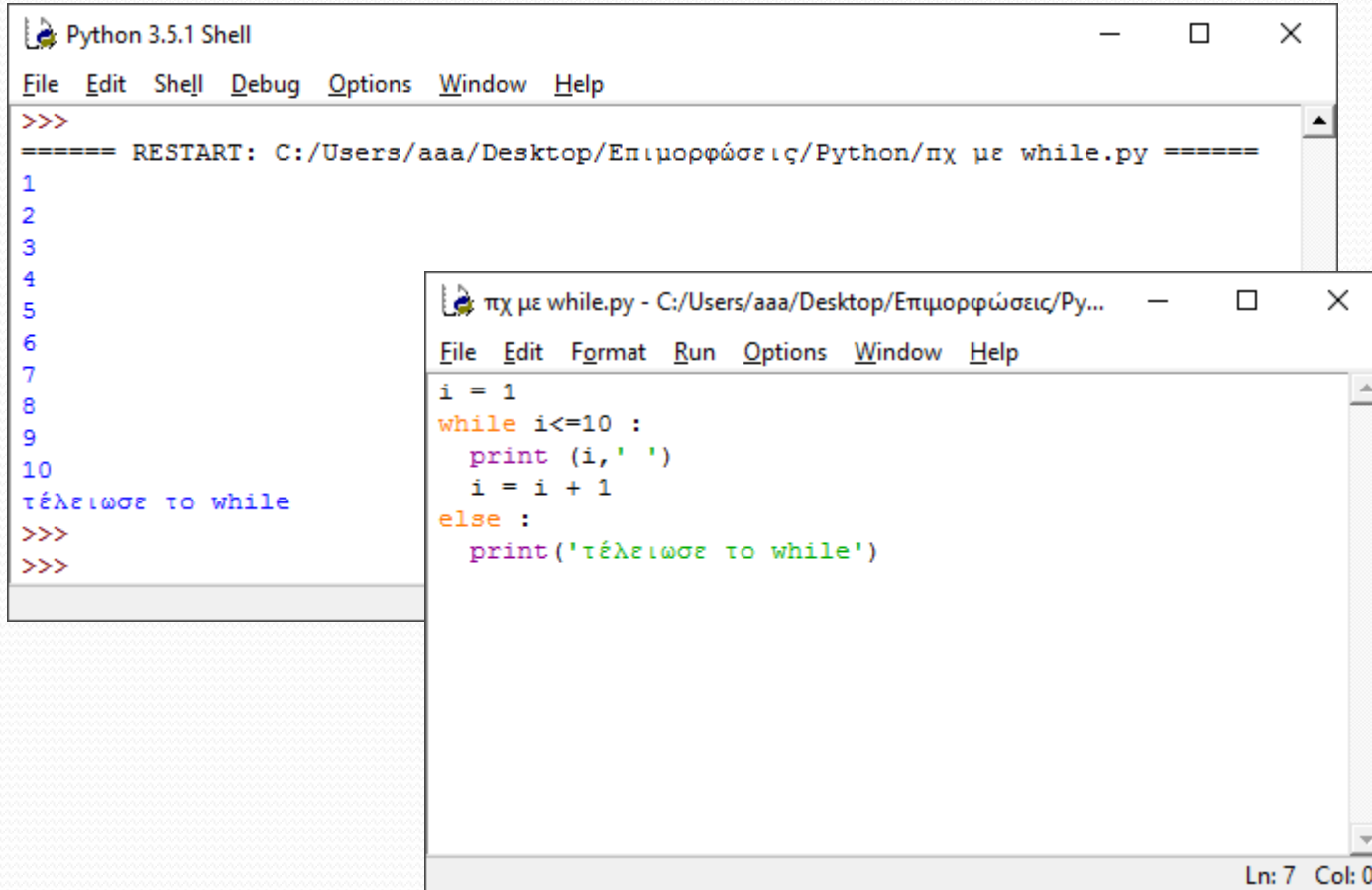
Εσοχή : η εσοχή είναι απαραίτητη όταν θέλουμε να δηλώσουμε ότι οι εντολές είναι μπλοκ μέσα σε κάποιο if ή σε κάποιο while ή σε κάποιο for. Δουλεύει από 2 και πάνω κενά. Ο IDLE τοποθετεί τέσσερα κενά. Επίσης ο IDLE μετατρέπει το TAB σε τέσσερα κενά

Η εντολή While

- Όσο η συνθήκη test είναι **Αληθής** εκτελείται το «κόκκινο» block
- Όταν η συνθήκη test γίνει **Ψευδής** εκτελείται το «γαλάζιο» block και στη συνέχεια η ροή βγαίνει από το βρόχο
- Η else είναι προαιρετική. Αν δεν υπάρχει τότε όταν test γίνει **Ψευδής** η ροή βγαίνει από το βρόχο
- Εάν το «κόκκινο» block περιλαμβάνει και την **break** τότε γίνεται έξοδος από το βρόχο χωρίς να εκτελεστεί ο κλάδος else

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

Η εντολή While παράδειγμα



The image shows two overlapping windows. The top window is a 'Python 3.5.1 Shell' with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a command prompt. It displays a restart message and a list of line numbers from 1 to 10. Line 10 contains the text 'τέλειωσε το while', and lines 11 and 12 show '>>>'.

The bottom window is a Python IDE titled 'πχ με while.py - C:/Users/aaa/Desktop/Επιμορφώσεις/Py...'. It has a menu bar (File, Edit, Format, Run, Options, Window, Help) and contains the following Python code:

```
i = 1
while i<=10 :
    print (i, ' ')
    i = i + 1
else :
    print('τέλειωσε το while')
```

The status bar at the bottom right of the IDE window shows 'Ln: 7 Col: 0'.

break – continue – pass

- Η **break** προκαλεί άμεση έξοδο από τον ‘πλησιέστερο’ βρόχο χωρίς την εκτέλεση του else του
- Η **continue** οδηγεί άμεσα την εκτέλεση στην αρχή του βρόχου παραλείποντας άλλες εντολές (αν υπάρχουν)
- Η **pass** είναι απλά μια ‘κενή’ δήλωση που μπαίνει για να είναι τυπικά συντακτικά σωστός ο βρόχος

```
while test-1:  
    statements-1  
    if test-2: break  
    if test-3: continue  
else:  
    statements-2
```

Η εντολή for

- **iterator**: απαριθμητής των επαναλήψεων
- **objectList**: απαριθμήσιμη(iterable) λίστα διακριτών αντικειμένων
 - An object capable of returning its members one at a time
- **else**: προαιρετικός κλάδος που εκτελείται 1 φορά αφού ολοκληρωθεί η επανάληψη (και δεν έχει γίνει έξοδος με break)

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


Η εντολή for & range & step

- και για όσους έχουν απορία τι γίνεται αν το βήμα είναι αρνητικό και αρχή < τέλος

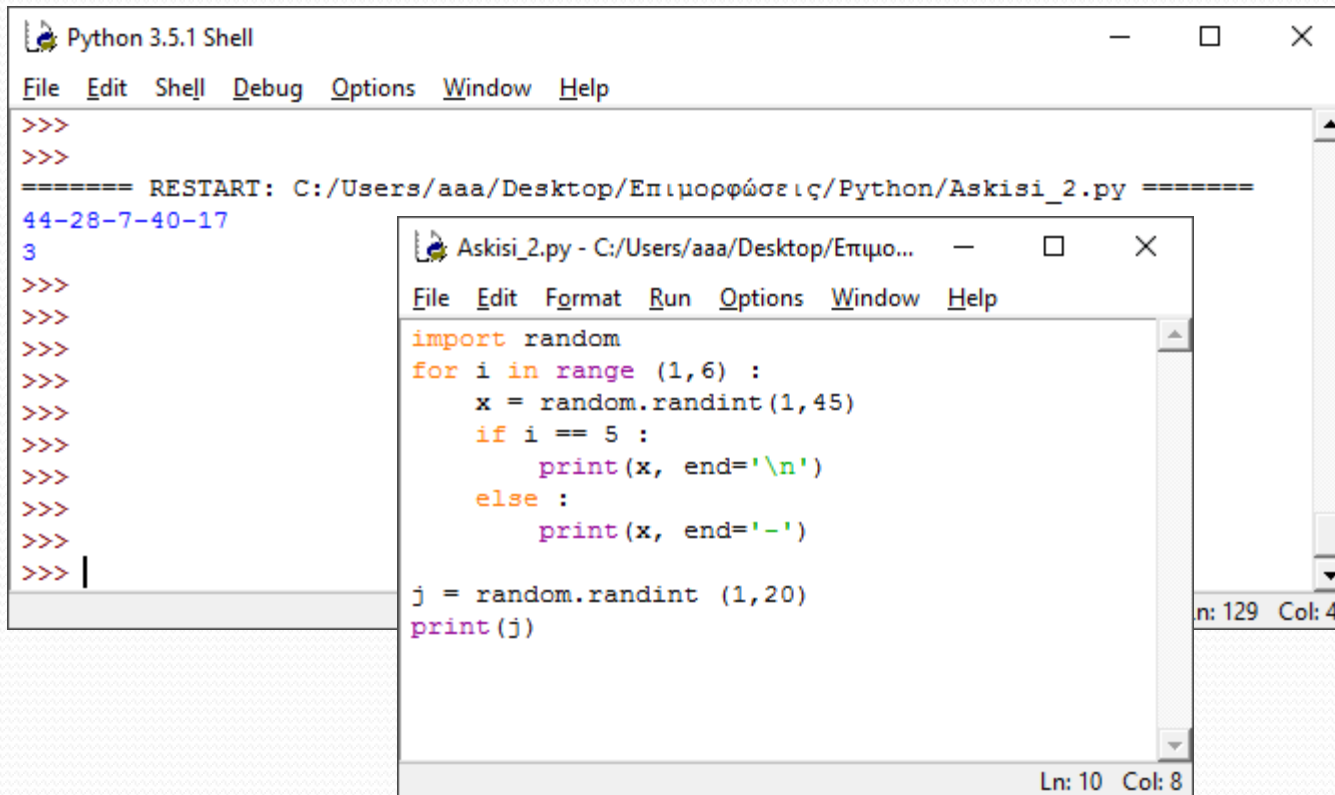
```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\aaa\Desktop\Επιμορφώσεις\Python\πχ με for.py =====
τέλειωσε το for
>>> |
```

```
πχ με for.py - C:\Users\aaa\Desktop\Επιμορφώσεις\Python\πχ με for.py (3.5.1)
File Edit Format Run Options Window Help
for i in range (1, 10, -2) :
| print (i, ' ')
else :
    print('τέλειωσε το for')
```

Ln: 2 Col: 0

Άσκηση 2

- ΝΓΠ που να τυπώνει τυχαίους αριθμούς για ένα δελτίο Joker. (5 αριθμοί στο διάστημα [1, 45] και ένας στο διάστημα [1-20])



The image shows a Python 3.5.1 Shell window and a code editor window. The shell window displays the output of a Python script, including a restart message and the number 3. The code editor window shows the source code of the script, which generates 5 random numbers between 1 and 45, and one random number between 1 and 20.

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
===== RESTART: C:/Users/aaa/Desktop/Επιμορφώσεις/Python/Askisi_2.py =====
44-28-7-40-17
3
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> |

Askisi_2.py - C:/Users/aaa/Desktop/Επιμο...
File Edit Format Run Options Window Help
import random
for i in range (1,6) :
    x = random.randint(1,45)
    if i == 5 :
        print(x, end='\n')
    else :
        print(x, end='-')

j = random.randint (1,20)
print(j)
Ln: 10 Col: 8
n: 129 Col: 4
```

Λίστες (Lists) – Μονοδιάστατοι Πίνακες

- Η Λίστα είναι η πιο γενική και ευέλικτη δομή ακολουθίας της Python
- **Ordered: Διατεταγμένη**
 - Είναι μια ακολουθιακή δεικτοδοτημένη συλλογή αντικειμένων, πχ. ακέραιοι, αλφαριθμητικά, κλάσεις, άλλες λίστες, λεξικά, κλπ.
- **Indexed: Δεικτοδοτημένη**
 - η θέση ενός αντικειμένου σε μια Λίστα προσδιορίζεται με χρήση δείκτη μέσα σε αγκύλες που γράφεται δίπλα στο όνομα της λίστας, πχ. `lista[2]`
 - **Zero indexed:** ο δείκτης `[0]` προσδιορίζει το πρώτο στοιχείο / δεν έχει κάποιο προκαθορισμένο περιορισμό στο μέγεθός της

```
Python 3.5 (32-bit)
Python 3.5.1 (v3.5.1:37a07cee
tel)] on win32
Type "help", "copyright", "cr
>>> lista = [9, 15, 26, 45]
>>> lista
[9, 15, 26, 45]
>>> lista[2]
26
>>> lista[0]
9
>>>
```

Λίστες (Lists) – Μονοδιάστατοι Πίνακες

- **Iterable:** Απαριθμήσιμη
 - Μια απαριθμήσιμη δομή (iterable) μπορεί να χρησιμοποιείται σε μια δομή επανάληψης
- **Mutable:** Μεταλλάξιμη
 - Σε μια μεταλλάξιμη δομή (mutable) τα δεδομένα της μπορούν να μεταβληθούν στη θέση μνήμης ('in place') χωρίς να δημιουργηθεί νέα λίστα
- **Heterogeneous:** Ανομοιογενής
 - Μπορεί να περιλαμβάνει δεδομένα διαφορετικού τύπου

```
>>>  
>>> for i in [1, 3, 4, 5]:  
...     print(i)  
...  
1  
3  
4  
5
```

Ένα for που λειτουργεί με λίστα αντί για range

```
Python 3.5 (32-bit)  
0  
>>> check = 1  
>>> lista = [ 10 , 2.45 , 'hello' ,check]  
>>> lista  
[10, 2.45, 'hello', 1]  
>>> lista[0]  
10  
>>> lista[2]  
'hello'  
>>> lista[3]  
1  
>>>
```

Λίστα - Δεικτοδότηση

```
Python 3.5 (32-bit)
0
>>> check = 1
>>> lista = [ 10 , 2.45 , 'hello' ,check]
>>> lista
[10, 2.45, 'hello', 1]
>>> lista[0]
10
>>> lista[2]
'hello'
>>> lista[3]
1
>>> lista[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'type' object is not subscriptable
>>>
```

Λίστα - Δεικτοδότηση

`lista [len (lista) - 1] ...` το τελευταίο στοιχείο της λίστας

```
Python 3.5 (32-bit)
0
>>> check = 1
>>> lista = [ 10 , 2.45 , 'hello' ,check]
>>> lista
[10, 2.45, 'hello', 1]
>>> lista[0]
10
>>> lista[2]
'hello'
>>> lista[3]
1
>>> lista[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'type' object is not subscriptable
>>>
>>>
>>> len(lista)
4
>>> lista[len(lista)-1]
1
>>>
```

Λίστα – πράξεις

```
Python 3.5 (32-bit)
>>> A = ['ένα', 'δύο', 'τρία']
>>>
>>> A[0]
'ένα'
>>>
>>> A[0] = 1
>>> A
[1, 'δύο', 'τρία']
>>> len(A)
3
>>> A[3] = 4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
>>>
>>> A
[1, 'δύο', 'τρία']
>>> A[1] = A[1] + A[2]
>>> A
[1, 'δύοτρία', 'τρία']
>>> A[0] = A[1] + A[2]
>>> A
['δύοτρίατρία', 'δύοτρία', 'τρία']
>>>
>>>
>>>
```

Λίστα – πράξεις

```
Python 3.5 (32-bit)
0
>>> B = [ 1, 2, 3, 4]
>>>
>>> G = [ 5, 6, 7, 8]
>>>
>>> D = B + G
>>>
>>> D
[1, 2, 3, 4, 5, 6, 7, 8]
>>>
>>> D[1] = B[2] + G[3]
>>> D
[1, 11, 3, 4, 5, 6, 7, 8]
>>>
>>>
>>> B2 = B * 2
>>>
>>> B2
[1, 2, 3, 4, 1, 2, 3, 4]
>>>
>>> _
```

For & Λίστα

Ο τελεστή **in** επιστρέφει True αν ένα αντικείμενο είναι μέλος μιας ακολουθιακής δομής (sequence).

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window
>>>
>>>
== RESTART: C:/Users/aaa/Desktop/
1
2
3
>>>
>>>
>>>
>>>
>>>
== RESTART: C:/Users/aaa/Desktop/
1
2
3
4
5
>>>
>>>
RESTART: C:/Users/a
elephant
tiger
dragon
>>> |
```

```
πχ με for_kai_lista.py - C:/Users/aaa/Desktop/Επιμο...
File Edit Format Run Options Window Help
for i in (1, 2, 3) :
    print (i, ' ')
```

```
πχ με for_kai_lista.py - C:/Users/aaa/Desktop/Επιμο...
File Edit Format Run Options Window Help
lista = [1, 2, 3, 4, 5]
for i in lista |:
    print (i, ' ')
```

```
πχ με for_kai_lista_string.py - C:/Users/aaa/Desktop/...
File Edit Format Run Options Window Help
zoo = ['elephant', 'tiger', 'dragon']
for i in zoo :
    print (i, ' ')
```

Ln: 2 Col: 13

Χρήσεις του in

```
zoo=['elephant', 'tiger', 'lion']

for i in range(3):
    print(zoo[i])

for animal in zoo:
    print(animal)

if 'dog' in zoo:
    '<do something-1>'
else:
    '<do something-2>|'

if 'dog' in zoo:
    pass
else:
    pass

print('tiger' in zoo)
print('cat' in zoo)
```

Ο τελεστή **in** επιστρέφει True αν ένα αντικείμενο είναι μέλος μιας ακολουθιακής δομής (sequence).

Λίστα – list () - append

```
Python 3.5 (32-bit)
0
>>> s = 'TEST'
>>> s
'TEST'
>>> lista = list(s)
>>> lista
['T', 'E', 'S', 'T']
>>>
>>> a = list(range(10))
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```

Η **list()** μετατρέπει μια
απαριθμήσιμη δομή σε
Λίστα

```
Python 3.5 (32-bit)
0
>>> lista
['T', 'E', 'S', 'T']
>>> lista.append(2)
>>> lista
['T', 'E', 'S', 'T', 2]
>>> lista.append('R')
>>> lista
['T', 'E', 'S', 'T', 2, 'R']
>>>
```

Η **append** είναι μέθοδος της δομής
λίστας και προσθέτει ένα στοιχείο
στην τελευταία θέση της λίστας.

Λίστα – append & del – list comprehension

```
Python 3.5 (32-bit)
>>>
>>> lista
['T', 'E', 'S', 'T']
>>> del lista[2]
>>> lista
['T', 'E', 'T']
>>> lista[2]
'T'
>>>
```

Η δήλωση **del**
αφαιρεί ένα στοιχείο
της λίστας

```
Python 3.5 (32-bit)
0
>>> λιστα = [i for i in range(10)]
>>> λιστα
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> _
```

Άλλος τρόπος
δημιουργίας λίστας
ονομάζεται
«list comprehension»

Εκκίνηση από αυτό το νούμερο

List comprehension-πχ (1)

```
listnum = [i-1 for i in range(10)]
```

```
[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8]
```

```
listnum = [2*i for i in range(10)]
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
listnum = [(2*(i**2)+4*i) for i in range(10)]
```

```
[0, 6, 16, 30, 48, 70, 96, 126, 160, 198]
```

```
lista = ['Λέξη' for i in range(5)]
```

```
['Λέξη', 'Λέξη', 'Λέξη', 'Λέξη', 'Λέξη']
```

```
lista = ['Λέξη'+ str(i) for i in range(5)]
```

```
['Λέξη0', 'Λέξη1', 'Λέξη2', 'Λέξη3', 'Λέξη4']
```

List comprehension-πχ (2)

```
lista = ['1' if i<3 else '2' for i in range(1,6)]  
        ['1', '1', '2', '2', '2']
```

```
lista = [0 if i%2==0 else 1 for i in range(10)]  
        [0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
```

```
lista = [1 if i in [0, 1] else (i-1)+(i-2) for i in range(10)]  
        [1, 1, 1, 3, 5, 7, 9, 11, 13, 15]
```

```
lista = [1 if i in [0,1] else lista[i-1]+lista[i-2] for i in range(20)]
```

```
lista = [chr(i) for i in range(65, 70)]  
        ['A', 'B', 'C', 'D', 'E']
```

```
lista = [2**i if i<5 else chr(65+2*i) for i in range(10)]  
        [1, 2, 4, 8, 16, 'K', 'M', 'O', 'Q', 'S']
```

Λίστα – Μετάλλαξη

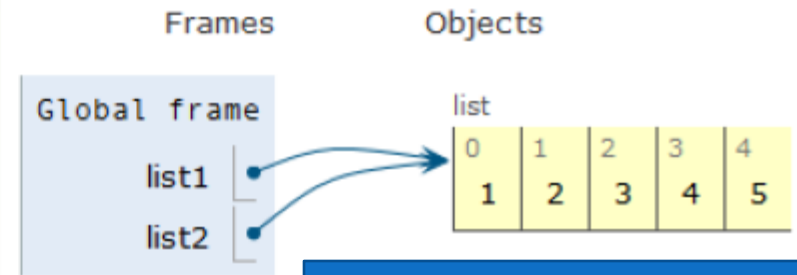
```
list1 = [1, 2, 3, 4, 5]
```

```
list2 = list1
```

```
list2[0] = 'x'
```

```
print(list1, '\n', list2)
```

```
['x', 2, 3, 4, 5]  
['x', 2, 3, 4, 5]
```



Το Object παραμένει το ίδιο

- Η εντολή `list2 = list1` απλά συνδέει το νέο όνομα `list2` με την υπάρχουσα δομή λίστας με την οποία συνδέεται ήδη το όνομα `list1`
- Τα ονόματα `list1` & `list2` αναφέρονται στην ίδια δομή, δηλαδή δημιουργούν μια *shared reference*
- Επομένως κάθε αλλαγή τιμής στη δομή συνδέεται και με τα δύο ονόματα

Λίστα – Τομή

```
list1 = [1,2,3,4,5,6,7,8,9,10]
```

```
list2 = list1[2:5]
```

```
print(list2)
```

```
[3, 4, 5]
```

Δημιουργείται νέο Object

Global frame

list1

list2

list

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10

list

0	1	2
3	4	5

όχι το 5^ο
στοιχείο

Η list2 θα έχει τα στοιχεία της list1 που βρίσκονται στο διάστημα [2, 5)

Python 3.5 (32-bit)

```
>>> A = [1,2,3,4,5]
>>> B = A[2:4]
>>> B
[3, 4]
>>> A[2] = 'T'
>>> A
[1, 2, 'T', 4, 5]
>>> B
[3, 4]
>>> -
```

Δημιουργείται νέο Object.....πχ

δημιουργώ την A. η B είναι τομή της A. Αλλάζω ένα από τα κοινά στοιχεία και ενώ αλλάζει για την A, δεν αλλάζει για την B

Λίστα – Slicing

- Τεχνική slicing, με την οποία δημιουργούνται τομές της αρχικής λίστας
- `list2 = list1[start : end : step]`

```
spam=['A', 'B', 'C', 'D', 'E']  
foo = spam[2:4]    ['C', 'D']  
foo2 = spam[1:]   ['B', 'C', 'D', 'E']  
foo3 = spam[:3]   ['A', 'B', 'C']  
foo4 = spam[:]    ['A', 'B', 'C', 'D', 'E']
```

Λίστα – τομή (slicing) – πχ (1)

και κάποιες τεχνικές για αντικατάσταση ή εισαγωγή νέων στοιχείων

```
L = [1, 2, 3]
```

```
L[1:2] = [4, 5]
```

```
[1, 4, 5, 3]
```

```
L[1:1] = [6, 7]
```

```
[1, 6, 7, 4, 5, 3]
```

```
L[1:2] = []
```

```
[1, 7, 4, 5, 3]
```

- Όταν προσδιορίζουμε μια **τομή** της λίστας τότε οι νέες τιμές **αντικαθιστούν** τις επιλεγμένες τιμές.

- Όταν προσδιορίζουμε ένα **σημείο** (όχι τομή) της λίστας τότε οι νέες τιμές **εισάγονται** χωρίς να επηρεάζονται οι άλλες τιμές

```
L = [1]
```

```
L[:0] = [2, 3, 4]
```

```
[2, 3, 4, 1]
```

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΑΡΧΗ

```
L[len(L):] = [5, 6, 7]
```

```
[2, 3, 4, 1, 5, 6, 7]
```

ΕΙΣΑΓΩΓΗ ΣΤΟ ΤΕΛΟΣ

Λίστα – ταξινόμηση

παράμετρος reverse

εντολή sorted

εντολή sort()

```
>>> lista6 = [4, 5, 2, 7, 1, 3, 4, 3, 2]
>>> sorted(lista6)
[1, 2, 2, 3, 3, 4, 4, 5, 7]
>>> sorted(lista6, reverse=True)
[7, 5, 4, 4, 3, 3, 2, 2, 1]
>>>
>>> lista6.sort()
>>> lista6
[1, 2, 2, 3, 3, 4, 4, 5, 7]
>>> lista6.sort(reverse=True)
>>> lista6
[7, 5, 4, 4, 3, 3, 2, 2, 1]
>>> _
```

- Αρχικά είχε επιλεγεί η Quicksort
- Λόγω διαφόρων προβλημάτων επιλέχθηκε για την ενσωματωμένη ταξινόμηση η adaptive merge sort
- Φυσικά μπορούμε να επέμβουμε στον κώδικα και να γράψουμε όποια ταξινόμηση θέλουμε

Λίστα – ταξινόμηση

Δουλεύει και με ταξινόμηση Ονομάτων

```
>>> lista5 = ['Λάζαρος', 'Μανώλης', 'Αντώνης', 'Κώστας', 'Βασίλης']
>>> sorted(lista5)
['Αντώνης', 'Βασίλης', 'Κώστας', 'Λάζαρος', 'Μανώλης']
>>> sorted(lista5, reverse=True)
['Μανώλης', 'Λάζαρος', 'Κώστας', 'Βασίλης', 'Αντώνης']
```

```
>>> lista5.sort()
>>> lista5
['Αντώνης', 'Βασίλης', 'Κώστας', 'Λάζαρος', 'Μανώλης']
>>> lista5.sort(reverse=True)
>>> lista5
['Μανώλης', 'Λάζαρος', 'Κώστας', 'Βασίλης', 'Αντώνης']
```

Άσκηση 3

- ΝΓΠ που να δημιουργεί μια λίστα με 10 τυχαίους ακέραιους και στη συνέχεια να τους **ταξινομεί** με **Bubble sort**

```
Bubble_sort.py - C:/Users/aaa/Desktop/Επιμορφώσεις/Python/Bubble_sort.py (3.5.1)
File Edit Format Run Options Window Help
import random
lista = []
for i in range(1, 11):
    lista.append(random.randint(1,20))

# το (len(lista)) είναι 10
# lista = [random.randint(1,20) for i in range(1,10)]
print (lista)
```

Δημιουργία 10 τυχαίων στο διάστημα 1-20

```
>>>
===== RESTART: C:/Users/aaa/Desktop/En
110 0 11 0 10 11 7 0 20 151
```

Άσκηση 3

- ΝΓΠ που να δημιουργεί μια λίστα με 10 τυχαίους ακέραιους και στη συνέχεια να τους **ταξινομεί** με **Bubble sort**

```
Bubble_sort.py - C:/Users/aaa/Desktop/Επιμορφώσεις/Python/Bubble_sort.py (3.5.1)
File Edit Format Run Options Window Help
import random
lista = []
for i in range(1, 11):
    lista.append(random.randint(1,20))

# το (len(lista)) είναι 10
# lista = [random.randint(1,20) for i in range(1,10)]

print (lista)

for k in range(1, len(lista)):
    for i in range(len(lista)-1, k-1, -1):
        if lista[i-1] > lista[i]:
            temp = lista[i]
            lista[i] = lista[i-1]
            lista[i-1] = temp
            # lista[i], lista[i-1] = lista[i-1], lista[i] ! εντολή swap
            # print(lista)

print(lista)
```

εκτέλεση

```
>>>
===== RESTART: C:/Users/aaa/Desktop/En
[13, 8, 11, 8, 10, 11, 7, 9, 20, 15]
[7, 8, 8, 9, 10, 11, 11, 13, 15, 20]
>>>
```

Άσκηση 4

- ΝΓΠ που να δημιουργεί μια λίστα με 10 τυχαίους ακέραιους και στη συνέχεια να **αναζητά** στοιχείο που δίνει ο χρήστης.

```
import random
lista = []
for i in range(1, 11):
    lista.append(random.randint(1,20))

#lista = [random.randint(1, 20) for i in range(1, 10)]

print(lista)
lista.sort()
print(lista)
```

Διαδική Αναζήτηση

```
import random
lista = []
for i in range(1, 11):
    lista.append(random.randint(1,20))

#lista = [random.randint(1, 20) for i in range(1, 10)]

print(lista)
```

Σειριακή Αναζήτηση

Άσκηση 4

- ΝΓΠ που να δημιουργεί μια λίστα με 10 τυχαίους ακέραιους και στη συνέχεια να **αναζητά** στοιχείο που δίνει ο χρήστης.

```
import random
lista = []
for i in range(1, 11):
    lista.append(random.randint(1,20))

#lista = [random.randint(1, 20) for i in range(1, 10)]

print(lista)
lista.sort()
print(lista)

item = int(input('Δώσε στοιχείο για αναζήτηση: '))
first = 0
last = len(lista)-1
found = False
while first <= last and not found:
    m = (first + last) // 2
    if lista[m] == item:
        found = True
    else:
        if item < lista[m]:
            last = m - 1
        else:
            first = m + 1

print(found)
```

```
import random
lista = []
for i in range(1, 11):
    lista.append(random.randint(1,20))

#lista = [random.randint(1, 20) for i in range(1, 10)]

print(lista)

item = int(input('Δώσε στοιχείο για αναζήτηση: '))
pos = 0
found = False
while pos < len(lista) and not found:
    if lista[pos] == item:
        found = True
    else:
        pos = pos+1

print(found)
```

List of Lists (LoL) – (Δισδιάστατοι Πίνακες)

Τα στοιχεία μιας λίστας μπορεί να είναι τα ίδια λίστες

```
>>> list1=[1,2,3]
>>> list1
[1, 2, 3]
>>>
>>> list2=[[0,5,10],['A','B','C'],[7.3, 20.2]]
>>> list2
[[0, 5, 10], ['A', 'B', 'C'], [7.3, 20.2]]
>>> list2[1][2]
'C'
>>> list2[2][1]
20.2
>>> list2[0][0]
0
```

- Ο **πρώτος δείκτης** αναφέρεται πάντοτε στα στοιχεία της κύριας λίστας
- Ο **δεύτερος δείκτης** προσδιορίζει τα στοιχεία της λίστας-στοιχείου

List of Lists - comprehension

χρειαζόμαστε 2 'εμφωλευμένους' βρόχους for

```
>>> list1 = [[0 for i in range(5)] for k in range(3)]
>>> list1
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
^^^
```

- **for k in range(3):** δημιουργεί τη λίστα list1 η οποία έχει 3 στοιχεία
- **for i in range(5):** δημιουργεί τις 3 λίστες-στοιχεία
- Κάθε λίστα-στοιχείο είναι λίστα με 5 απλά δεδομένα
- Κάθε απλό δεδομένο έχει την τιμή 0

```
for j in range(3):
    print(list1[j])
```

```
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
```

List of Lists - comprehension

```
list1 = [[i+3*k for i in range(3)] for k in range(5)]  
for j in range(5):  
    print(list1[j])
```

```
[0, 1, 2]  
[3, 4, 5]  
[6, 7, 8]  
[9, 10, 11]  
[12, 13, 14]
```

Άσκηση 5

- ΝΓΠ που να δημιουργεί μια LoL3x3 με τυχαίους ακέραιους και να υπολογίζει το άθροισμα κάθε στοιχείου της. Στο τέλος να τυπώνει την LoL και το αντίστοιχο άθροισμα.

```
import random
lista = [[random.randint(1, 10) for i in range(3)] for k in range(3)]
```

Κάντε παραδοσιακή
άθροιση

Άσκηση 5

- ΝΓΠ που να δημιουργεί μια `LoL3x3` με τυχαίους ακέραιους και να υπολογίζει το άθροισμα κάθε στοιχείου της. Στο τέλος να τυπώνει την `LoL` και το αντίστοιχο άθροισμα.

```
import random
lista = [[random.randint(1, 10) for i in range(3)] for k in range(3)]

print(lista)

for i in range(3):
    print(lista[i], sum(lista[i]))
```

Συναρτήσεις Λίστας

- **len(list):** μήκος (πλήθος στοιχείων της λίστας)
- **max(list):** στοιχείο με τη μέγιστη τιμή στη λίστα
- **min(list):** στοιχείο με την ελάχιστη τιμή στη λίστα
- **list(seq):** ακολουθιακή δομή σε λίστα
- **Sum(list):** άθροισμα στοιχείων λίστας

```
>>> a = [78, 23, 52, 47, 65]
>>> len(a)
5
>>> max(a)
78
>>> min(a)
23
>>> sum(a)
265
```

```
>>> b = [[random.randint(1,10) for i in range (5)] for k in range (7)]
>>> b
[[8, 7, 5, 8, 2], [1, 3, 8, 9, 5], [6, 1, 8, 9, 6], [4, 6, 10, 7, 1], [2, 4, 3,
9, 7], [1, 9, 4, 10, 6], [8, 2, 5, 7, 5]]
>>> max(b)
[8, 7, 5, 8, 2]
>>> min(b)
[1, 3, 8, 9, 5]
>>> max(b[0])
8
>>> min(b[1])
1
>>> sum(b[0])
30
>>> sum(b[1])
26
>>>
```

Μέθοδοι Λίστας

- **list.append(x)**: Προσθέτει το x στο τέλος της λίστας
- **list.extend(L)**: Επεκτείνει τη λίστα προσθέτοντας την απαριθμήσιμη δομή L στο τέλος της λίστας
- **list.insert(i, x)**: Εισάγει το x στη θέση i
- **list.remove(x)**: Απομακρύνει το στοιχείο x από τη λίστα
- **list.pop([i])**: Αφαιρεί ΚΑΙ επιστρέφει το στοιχείο που βρίσκεται στη θέση i της λίστας
- **list.clear()**: Αφαιρεί όλα τα στοιχεία της λίστας
- **list.index(x)**: δείκτης θέσης όπου βρίσκεται το x
- **list.count(x)**: πόσες φορές εντοπίζεται το x στη λίστα
- **list.sort()**: Ταξινομεί τα στοιχεία της λίστας ('in place').
- **list.reverse()**: Αντιστρέφει τη σειρά των στοιχείων της λίστας ('in place').
- **list.copy()**: Δημιουργεί αντίγραφο της λίστας

Συναρτήσεις (Functions)

- Η συνάρτηση ομαδοποιεί ένα σύνολο εντολών
- Ενεργοποιείται με την αναφορά του ονόματός της
- Επιτρέπει τον καθορισμό τιμών παραμέτρων που περνούν ως «ορίσματα» από το πρόγραμμα που καλεί προς τη συνάρτηση,
- Υπολογίζει μία ή περισσότερες τιμές που επιστρέφει στο πρόγραμμα που τις καλεί

Δήλωση Συνάρτησης

```
def name(arg1, arg2, ... argN):  
    statements  
    return value
```

- **def:** δεσμευμένη λέξη για τη δήλωση συνάρτησης
- **name:** το όνομα της συνάρτησης
- **(arg1, arg2, ... argN):** η λίστα παραμέτρων ή ορισμάτων
- **statements:** το block εντολών που εκτελεί η συνάρτηση
- **return:** δηλώνει το τέλος της συνάρτησης
- **value:** τιμή (μεταβλητή ή έκφραση) που υπολογίστηκε και επιστρέφεται στον κώδικα κλήσης

Παράδειγμα Συνάρτησης

Προσοχή στην εσοχή των 2 θέσεων (λειτουργεί και με 3 και με 4 θέσεις, δε λειτουργεί αν δεν υπάρχει εσοχή)

```
>>>
>>> def ginomeno(x,y):
...     return x*y
...
>>> a = ginomeno(2,3)
>>> a
6
>>> b = ginomeno('ok',3)
>>> b
'okokok'
```

Μορφές Σύνταξης Συνάρτησης

- Χωρίς παραμέτρους & χωρίς return
- Χωρίς παραμέτρους & με return
- Με παραμέτρους & χωρίς return
- Με παραμέτρους & με return
- Με παραμέτρους προκαθορισμένης τιμής

```
>>>  
>>> def sayHello():  
...     print('Hello')  
...  
>>> sayHello()  
Hello  
>>> _
```

```
>>>  
>>> def ThisIsTheEnd() :  
...     return(bool(0))  
...  
>>> ThisIsTheEnd()  
False
```

```
>>>  
>>> def sayHello(par):  
...     print('Hello ',par)  
...  
>>> sayHello('Bill')  
Hello Bill  
>>> _
```

classic

```
def dynamh(x,y=2):  
    return x**y
```

```
dyn = dynamh(2)  
print(dyn)
```

```
dyn = dynamh(2, 3)  
print(dyn)
```

Άσκηση 5

- ΝΓΠ το οποίο να εμφανίζει τη δύναμη ενός αριθμού, καλώντας κατάλληλη συνάρτηση για τον υπολογισμό
Θα ζητούνται από το χρήστη η βάση και ο εκθέτης και αφού κληθεί η κατάλληλη συνάρτηση θα επιστρέφει τη δύναμη, η οποία και θα εμφανίζεται

Άσκηση 5

- ΝΓΠ το οποίο να εμφανίζει τη δύναμη ενός αριθμού, καλώντας κατάλληλη συνάρτηση για τον υπολογισμό
Θα ζητούνται από το χρήστη η βάση και ο εκθέτης και αφού κληθεί η κατάλληλη συνάρτηση θα επιστρέφει τη δύναμη, η οποία και θα εμφανίζεται

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
===== RESTART: C:/Users/aaa/Desktop/Επιμορφώσεις/Python/Askisi_5.py
Πρόγραμμα Υπολογισμού Δύναμης
Δώσε τη βάση 2
Δώσε τον εκθέτη 3
8
Θες άλλο υπολογισμό ; (ν/ο) n
Δώσε τη βάση 3
Δώσε τον εκθέτη 3
27
Θες άλλο υπολογισμό ; (ν/ο) v
Δώσε τη βάση 2
Δώσε τον εκθέτη 6
64
Θες άλλο υπολογισμό ; (ν/ο) o
Δώσε τη βάση 3
Δώσε τον εκθέτη 4
81
Θες άλλο υπολογισμό ; (ν/ο) o
>>> |
```

```
Askisi_5.py - C:/Users/aaa/Desktop/Επιμορφώσεις/Python/Askisi_5.py
File Edit Format Run Options Window Help
print('Πρόγραμμα Υπολογισμού Δύναμης')

telos = bool(0)

def dynamh(x,y):
    return x**y

while telos == False :
    a = input('Δώσε τη βάση ')
    b = input('Δώσε τον εκθέτη ')
    print( dynamh( int(a),int(b) ) )
    c = input('Θες άλλο υπολογισμό ; (ν/ο) ')
    if c == ('o' or 'o') :
        telos = bool(1)
```

Ln: 214 Col: 4

Εκτέλεση Συνάρτησης από την Python

- Η δήλωση της συνάρτησης μπορεί να εμφανίζεται οπουδήποτε αρκεί όταν κληθεί, να έχει ήδη δηλωθεί

Εκτέλεση Συνάρτησης από την Python

- Η συνάρτηση μπορεί να αλλάζει όνομα κατά την εκτέλεση ή να συνδέεται και με άλλα ονόματα

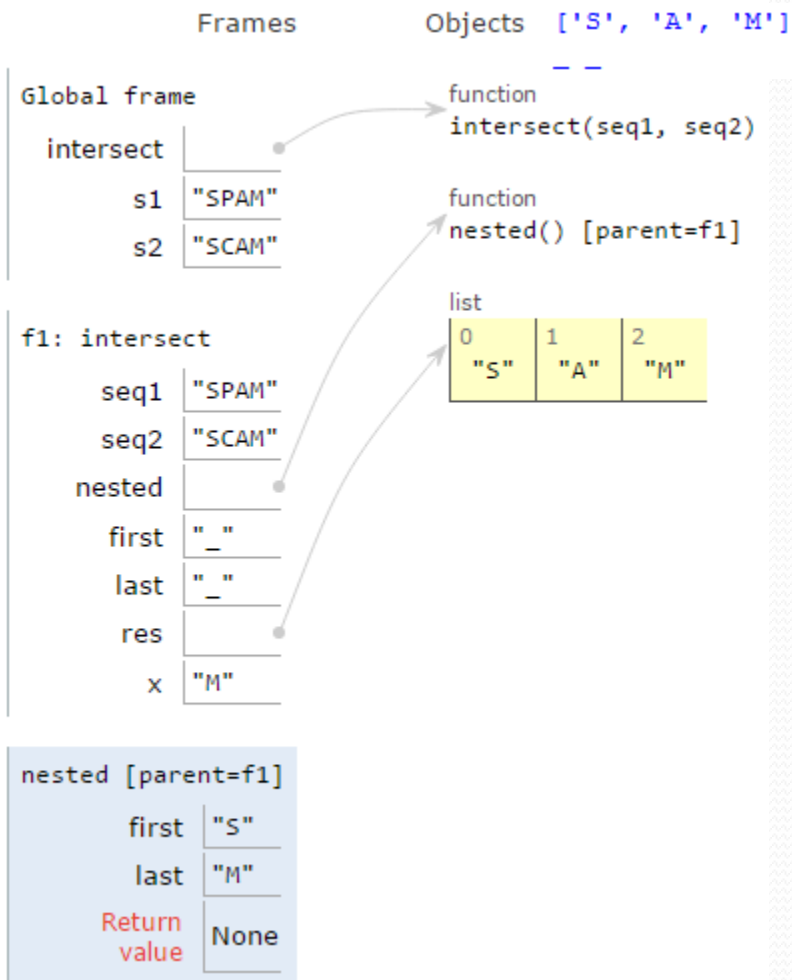
```
def dynamh(x, y=2) :  
    return x**y  
  
power = dynamh  
print(power(2))
```

Μετά την εκτέλεση της εντολής ανάθεσης
power= dynamh
η συνάρτηση μπορεί να κληθεί και με το όνομα 'power'

Εμβέλεια μεταβλητών

- Κάθε μεταβλητή που παίρνει τιμή μέσα σε μια συνάρτηση είναι **τοπική(local)** στη συνάρτηση αυτή
- Μια μεταβλητή που παίρνει τιμή σε επίπεδο κύριου προγράμματος είναι **καθολική(global)** στο πρόγραμμα
- Μια μεταβλητή που παίρνει τιμή σε μια περιβάλλουσα συνάρτηση είναι **μη-τοπική(nonlocal)** για τις φωλιασμένες συναρτήσεις

Εμβέλεια μεταβλητών



```
def intersect(seq1, seq2):
```

```
    def nested():
        #nonlocal first, last
        first = res[0]
        last = res[len(res)-1]
        return nested
```

```
    first = last = '_'
    res = []
    for x in seq1:
        if x in seq2:
            res.append(x)
    print(res)
    nested()
    print(first, last)
```

```
s1 = 'SPAM'
s2 = 'SCAM'
intersect(s1, s2)
```


Εμβέλεια μεταβλητών (επίπεδα διαχείρισης)

- Η Python διαχειρίζεται τα ονόματα σε **4 επίπεδα εμβέλειας**:
- **Local(Τοπική)**: Κάθε απλή συνάρτηση
- **Enclosing(Περικλείουσα)**: Κάθε συνάρτηση που περικλείει φωλιασμένες συναρτήσεις
- **Global(Καθολική)**: Κύριο πρόγραμμα
- **Built-in(Ενσωματωμένη)**: Τα standard ονόματα της γλώσσας / Άλλα πακέτα / Βιβλιοθήκες, κλπ
- **Γενικός κανόνας**: κάθε μεταβλητή είναι local στο επίπεδο της συνάρτησης όπου γίνεται ανάθεση τιμής, εκτός εάν εμφανίζεται σε δηλώσεις global ή nonlocal

Εμβέλεια μεταβλητών - Παραδείγματα

```
>>> x=10
>>> def sum(Y):
...     return x+Y
...
>>> sum(1)
```

```
>>> x = 10
>>> def sum():
...     x=8
...     print(x)
...
>>> sum()
>>> x
```

```
>>> x = 10
>>> def sum():
...     global x
...     x = 8
...
>>> sum()
>>> x
```

Τι αποτέλεσμα θα βγάλει
κάθε πρόγραμμα ;

Εμβέλεια μεταβλητών - Παραδείγματα

```
>>> x=10
>>> def sum(Y):
...     return x+Y
...
>>> sum(1)
11
```

Η x είναι global
επειδή δηλώνεται
στο κυρίως
πρόγραμμα

```
>>> x = 10
>>> def sum():
...     x=8
...     print(x)
...
>>> sum()
8
>>> x
10
>>>
```

Η x είναι global
όμως η local x
υπερτερεί της
global εντός της
συνάρτησης. Η τιμή
εκτός συνάρτησης
παραμένει

```
>>> x = 10
>>> def sum():
...     global x
...     x = 8
...
>>> sum()
>>> x
8
>>>
```

Η x είναι global
όμως η local x η
οποία θα δηλωθεί
ως global θα πάρει
την τιμή της
προηγούμενης
global x

Εμβέλεια μεταβλητών - Παραδείγματα

Ποια είναι η τελική τιμή της X ?

Τι θα βγάλει αν πληκτρολογήσουμε func1 ?

Τι θα βγάλει αν πληκτρολογήσουμε X μετά ?

```
X = 99

def func1():
    global X
    X = 88

def func2():
    global X
    X = 77
```

```
X = 99

def func1():
    X = 88
    def func2():
        print(X)
    func2()

func1()
```

Εμβέλεια μεταβλητών - Παραδείγματα

Η τελική τιμή της X
σχετίζεται με το
ποια συνάρτηση θα
κληθεί τελευταία

Όταν θα κληθεί η
func1 θα τυπώσει 88
Αν μετά ζητήσουμε
την τιμή της X αυτή
θα είναι 99, επειδή
είναι η τελευταία
γνωστή global

```
X = 99

def func1():
    global X
    X = 88

def func2():
    global X
    X = 77
```

```
X = 99

def func1():
    X = 88
    def func2():
        print(X)
    func2()

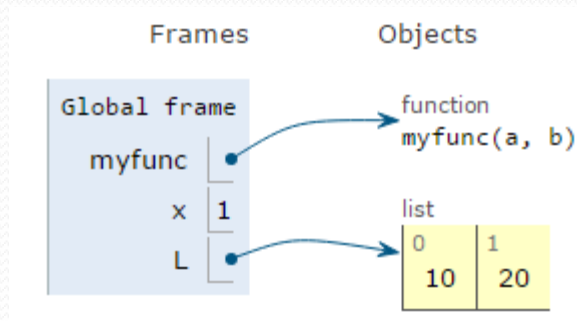
func1()
```

Ορίσματα & Εμβέλεια

- Το πέρασμα ορισμάτων γίνεται με ανάθεση αντικειμένων στα ονόματα των παραμέτρων που είναι τοπικές μεταβλητές της συνάρτησης
- Όμως στην πράξη εξαρτάται από την μεταλλαξιμότητα του αντικειμένου
- **Μη-μεταλλάξιμα (Immutable)** (πχ. ακέραιοι, αλφαριθμητικά – **by value**):
 - Περνούν στη συνάρτηση με δημιουργία αντιγράφου
- **Μεταλλάξιμα (Mutable): (by reference)**
 - Περνούν στη συνάρτηση «με δείκτη»
 - Άρα κάθε μεταβολή τους στην συνάρτηση (τοπική εμβέλεια) επηρεάζει και την καθολική εμβέλεια

Ορίσματα & Εμβέλεια - Παράδειγμα

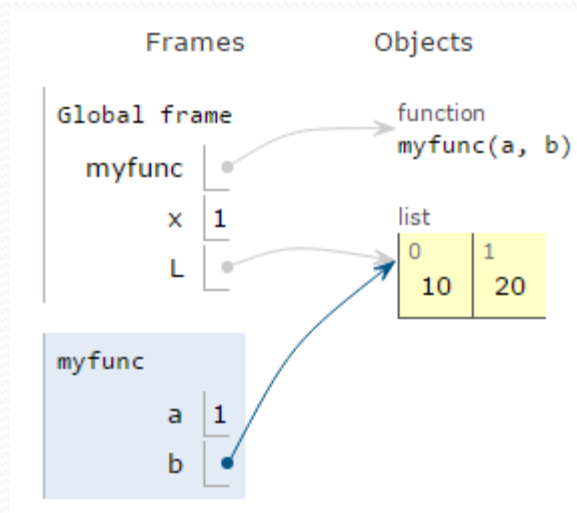
```
def myfunc(a, b):  
    a = 4  
    b[0] = 100  
  
X = 1  
L = [10, 20]  
  
myfunc(X, L)  
  
print(X, L)
```



Πριν την κλήση

Ορίσματα & Εμβέλεια - Παράδειγμα

```
def myfunc(a, b):  
    a = 4  
    b[0] = 100  
  
X = 1  
L = [10, 20]  
  
myfunc(X, L)  
  
print(X, L)
```

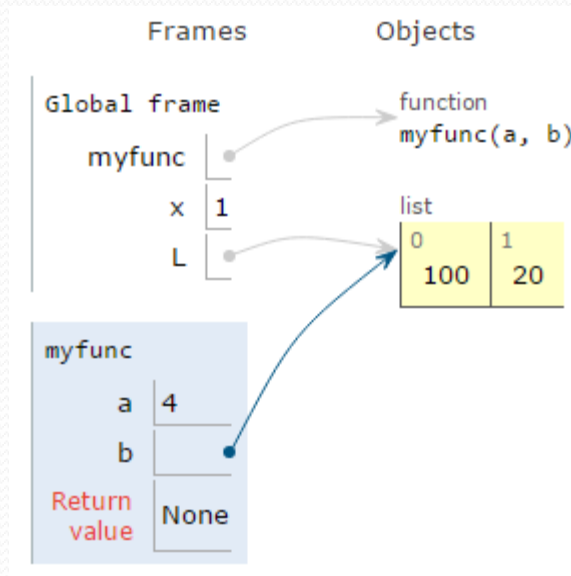


Κατά την κλήση

Ορίσματα & Εμβέλεια - Παράδειγμα

```
def myfunc(a, b):  
    a = 4  
    b[0] = 100  
X = 1  
L = [10, 20]  
  
myfunc(X, L)  
  
print(X, L)
```

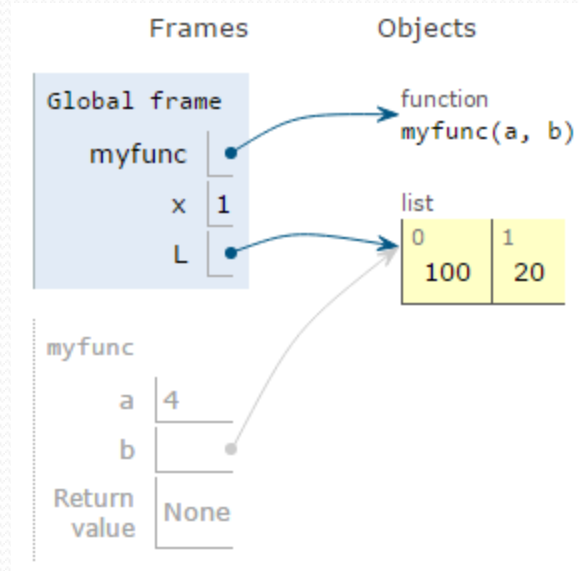
Κατά την εκτέλεση



Ορίσματα & Εμβέλεια - Παράδειγμα

```
def myfunc(a, b):  
    a = 4  
    b[0] = 100  
  
X = 1  
L = [10, 20]  
  
myfunc(X, L)  
  
print(X, L)
```

Μετά την κλήση



Program output:

```
1 [100, 20]
```

Βιβλιογραφία

- Lutz M., (2013) «Learning Python», O' Reilly
- Μπούγια Ι. (2015), «Παρουσίαση Γλώσσας Προγραμματισμού Python», Εκπαιδευτικός ΠΕ19, παρουσίαση επιμορφωτικής δράσης
- Python Software Foundations
<https://docs.python.org/>

Python

Τέλος

Στεφανίδης Βασίλειος

Υπεύθυνος ΚΕΠΛΗΝΕΤ Χαλκιδικής