

Εισαγωγή στη γλώσσα προγραμματισμού



Code Week 2017 @ POS 4 WORK

Powered by InterMediaKT

\$ whoami

Dennis Rodis

Software engineer / Web developer

Co Founder of [Susurrus](#)

Η Python είναι μια δημοφιλής ψηλού επιπέδου γλώσσα προγραμματισμού με κύρια χαρακτηριστικά:

- Δυναμική (Dynamic, Interpreted)
- Αναγνωσιμότητα
- Ευκολία χρήσης
- Παίζει σχεδόν παντού
- Ευνοεί τη γρήγορη ανάπτυξη εφαρμογών
- Επεκτάσιμη
- Ανοιχτό λογισμικό (open source)

# Γιατί Python - Monty Python



# Guido van Rossum - Δημιουργός της Python



By Doc Searls - 2006oscon\_203.JPG, CC BY-SA 2.0, [Link](#)

# Γιατί Python - Χαρακτηριστικά

- Συντακτική Απλότητα
- Υποχρεωτική Στοίχιση
- Κατάλληλη για πρώτη γλώσσα
- Υποστηρίζει πολλά στυλ προγραμματισμού  
(αντικειμενοστραφή, προστακτικό, συναρτησιακό)
- Μεγάλος αριθμός και ποικιλία σε βιβλιοθήκες
- Batteries Included



```
>>> import this
```

## Zen of Python

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts



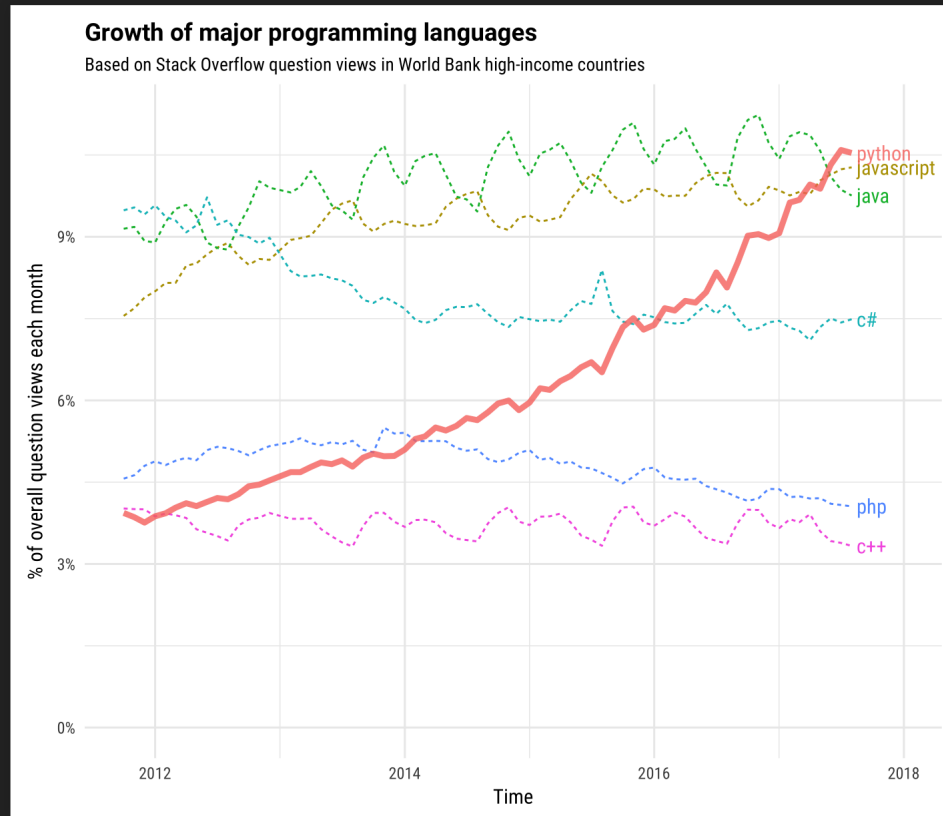
# Γιατί Python - Που χρησιμοποιείται

- Εκπαίδευση
- Ερευνα
- Εφαρμογές στον Παγκόσμιο Ιστο (Web)
- Data Science, Machine Learning
- Ασφάλεια/Information Security
- Internet of Things



- MIT
- NASA
- Google, Youtube
- RedHat
- Instagram, Dropbox,  
Pinterest
- Spotify, Quora, Reddit
- Πολλοί άλλοι!

# Γιατί Python - Δημοφιλής



Hint: Python's popularity in data science and machine learning is probably the main driver of its fast growth.

## The Incredible Growth of Python

Source: Stack Overflow Blog

# Χρησιμοποιώντας Python - Εγκατάσταση

- Linux\Unix
  - Προεγκαταστημένο στις περισσότερες διανομές
  - Debian/Ubuntu:

```
$ sudo apt-get install python3
```

- Fedora/Red hat:

```
$ sudo yum install python3
```



- Mac Os
  - Binaries από το επίσημο site

```
https://www.python.org/downloads/mac-osx/
```

- Via Homebrew (package manager):

```
$ brew install python3
```

Συνιστώμενος τρόπος εγκατάστασης για Windows  
(υπάρχουν και εκδόσεις για macOS, Linux)

## Anaconda Distribution

(most popular Python data science platform)

<https://www.anaconda.com/download>

- Easy Installation
- Includes many popular packages
- All Data Science tools



## Python2 vs Python3

Short version: Python 2.x is legacy, Python 3.x is the present and future of the language

Χρησιμοποιούμε Python3 (Python2 μόνο για λόγους συμβατότητας)

# Χρησιμοποιώντας Python - Εργαλεία

## Python interpreter - Interactive mode

```
$ python3
Python 3.5.2 (default, Sep 14 2017, 22:51:06)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more info
>>>
>>> print('hello world')
hello world
>>>
```



# IPython - A powerful interactive shell

Προσφέρει περισσότερες δυνατότητες (περιλαμβάνεται στο Anaconda)

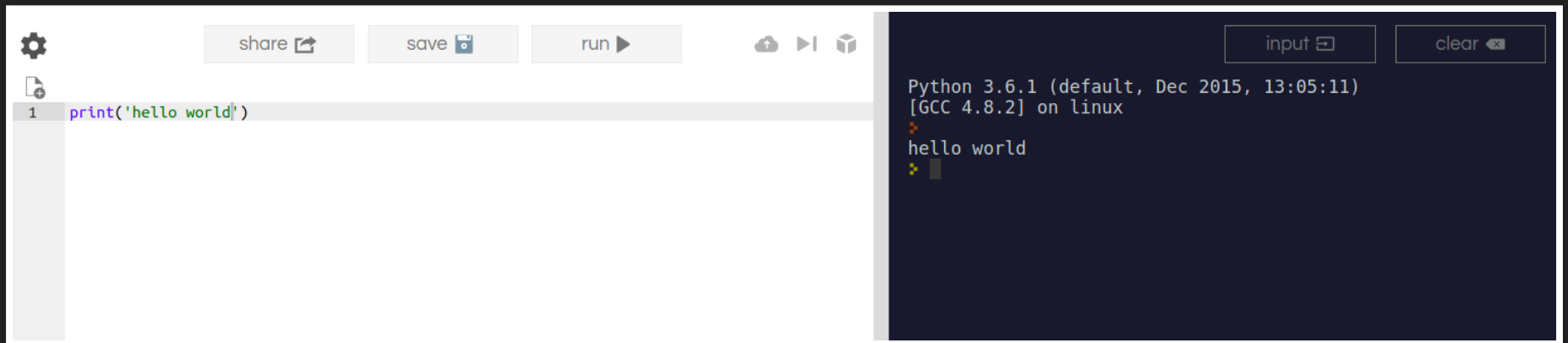
```
$ ipython
Python 3.5.2 (default, Sep 14 2017, 22:51:06)
IPython 5.1.0 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra

In [1]: print('hello world')
hello world

In [2]:
```

# Online Python IDE

<https://repl.it/languages/Python3>



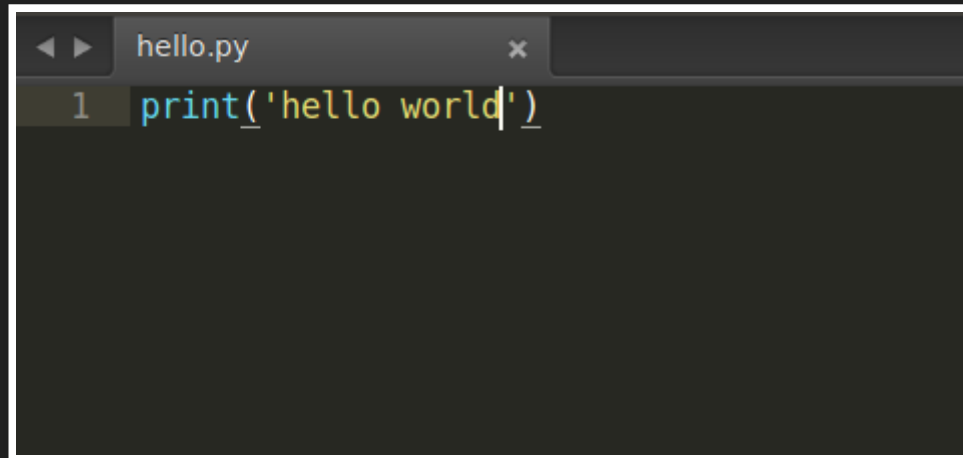
The screenshot displays the Online Python IDE interface. The top bar contains a settings gear icon, a 'share' button with an external link icon, a 'save' button with a floppy disk icon, a 'run' button with a play icon, and a cloud icon with a plus sign. The code editor on the left shows a single line of Python code: `1 print('hello world')`. The terminal on the right shows the output of the code execution: `Python 3.6.1 (default, Dec 2015, 13:05:11)`, `[GCC 4.8.2] on linux`, and `hello world`. The terminal also includes an 'input' button and a 'clear' button with an 'x' icon.

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
hello world
```

## Editor/IDE

- IDLE
- PyCharm (discounts for students)
- Sublime Text
- VS Code

# hello.py file

A screenshot of a code editor window. The title bar at the top shows a left arrow, a right arrow, the filename 'hello.py', and a close button 'x'. The editor area has a dark background. Line 1 is highlighted with a light green background. The code on line 1 is 'print('hello world')' with syntax highlighting: 'print' is blue, the opening quote is yellow, 'hello world' is green, and the closing quote is yellow. A white cursor is positioned at the end of the line, after the closing quote.

```
< > hello.py x
1 print('hello world')
```

```
$ python hello.py
hello world
```

# ΣΥΝΤΑΚΤΙΚΟ - Μεταβλητές

```
# integers
two = 2
some_number = 10000

# booleans
true_boolean = True
false_boolean = False

# string
my_name = "Dennis"
greek_name = "Διονύσης"

# float
book_price = 15.80
```

Στην Python δεν δηλώνουμε ρητά τι τύπος δεδομένων χρησιμοποιείται - γίνεται δυναμικά (dynamic typing)





# int, float, bool, str

```
type(some_number)  
<class 'int'>
```

```
type(true_boolean)  
<class 'bool'>
```

```
type(my_name)  
<class 'str'>
```

```
type(book_price)  
<class 'float'>
```



```
# είσοδος από τη γραμμή εντολών  
name = input("Όνομα; ")
```

```
# εκτύπωση στη γραμμή εντολών  
print("Καλημέρα", name)
```

```
# μετατροπή τύπου  
x = '2'  
x = int(2)  
type(x) # int
```

```
num = input("X: ")  
num ** 2 # throws error  
num = int(num)  
num ** 2 # correct
```

# Python ως κομπιουτεράκι

## Αριθμητικοί τελεστές, + - \* / \*\* // %

```
2 + 2          # 4
50 - 5*6       # 20
(50 - 5*6) / 4  # 5.0
8 / 5          # 1.6

17 / 3         # 5.666666666666667
17 // 3        # 5 - κρατά μόνο το ακέραιο μέρος
17 % 3         # υπόλοιπο 2

5 * 3 + 2      # 17

5 ** 2         # 5 στο τετράγωνο = 25
```

[gist link](#)



# Χρησιμοποιώντας τις μεταβλητές σε πράξεις

```
width = 20
height = 5 * 9
width * height      # 900

result = width * height
print(result)
```

# Συγκριτικοί τελεστές, > >= < <= == != Λογικοί τελεστές, and or not

```
a = 5
b = 4
a > b           # True
not a > b        # False
a == 5          # True
a != 5           # False
a >= 5           # True

c = 6
a > b and a > c  # False
a > b and not a > c # True

a > b or a > c    # True
not a > b or not a > c # True
not ( a > b ) or a > c # False
```



# Έλεγχος Ροής

## Δομή ελέγχου if

```
if συνθήκη:  
    εντολές  
  
elif συνθήκη:  
    εντολές  
  
else:  
    εντολές
```

Προσοχή στην στοίχιση! Δεν είναι προαιρετική!  
Μην παραλείπετε το σύμβολο :

[gist link](#)











# Δομές Επανάληψης

## Βρόγχος while

```
while συνθήκη:  
    εντολές
```

Βρόγχοι while θα τρέχουν συνεχώς όσο πληρείται μια συγκεκριμένη συνθήκη.

[gist link](#)







# break

```
# έξοδος με break
while True:
    num = int(input("X: "))

    if num == 0:
        break

    print(1/num)
```

```
# Fibonacci sum

def fib(a, b):
    sum = 0;

    while a < 4 * (10 ** 6):
        sum += a
        a, b = b, a + b

    return sum

a, b = 0, 1
print(fib(a, b))
```

Υπολογίζει το άθροισμα όλων των αριθμών της ακολουθίας Fibonacci οι οποίοι είναι μικρότεροι από το  $4 \cdot 10^6$

## Βρόγχος for

```
for μεταβλητή in ακολουθία:  
    εντολές
```

Η for διατρέχει τα στοιχεία μιας ακολουθίας (π.χ. μιας συμβολοσειράς ή μιας λίστας)

## for και ακολουθία range, συμβολοσειρά

```
# 0 μέχρι το 9
for i in range(10):
    print(i, end=" ")

# 5 μέχρι 45 (βήμα 10)
for i in range(5,50,10):
    print(i, end=" ")

# Κ-α-λ-η-μ-έ-ρ-α-
for c in "Καλημέρα":
    print(c, end="-")
```

# Εικασία του Κόλατζ / Collatz conjecture

Επιλέξτε έναν θετικό ακέραιο  $x$ .

Αν είναι άρτιος

    διαιρέστε τον με το 2,

ενώ αν είναι περιττός

    πολλαπλασιάστε τον με το 3 και προσθέστε άλλη μια μονάδα.

Επαναλάβετε τη διαδικασία με τον αριθμό που θα προκύψει.

Από οποιονδήποτε αριθμό  $x$  κι αν ξεκινήσετε,  
θα καταλήξετε τελικά στο 1.









# Συναρτήσεις

```
def add(a, b):  
    c = a + b  
  
    return c  
  
print(add(2, 3))
```

Μια συνάρτηση πρέπει να έχει οριστεί πριν χρησιμοποιηθεί. Τα αποτελέσματα επιστρέφονται με την return

[gist link](#)





```
# προεπιλεγμένα ορίσματα

def hello(message='Hello world!'):
    print(message)

hello()
hello('Hello there!')
hello(message='Hello there!')

def greetings(greet="hello", name="there"):
    print(greet + ' ' + name)

greetings() # hello there
greetings(name='dennis', greet='bye') # bye dennis
```

## Δομές δεδομένων

- Συμβολοσειρές (strings)
- Λίστες (lists)
- Πλειάδες (tuples)
- Σύνολα (sets)
- Λεξικά (dicts)

Οι συμβολοσειρές, λίστες, πλειάδες είναι ακολουθίες. Αυτό σημαίνει πως κάθε στοιχείο είναι σε μια αριθμημένη σειρά, με βάση και την οποία μπορεί να προσπελαστεί.

ARRAY

5	7	1	3	4
---	---	---	---	---

INDEX

↑      ↑      ↑      ↑      ↑  
0      1      2      3      4

# Συμβολοσειρές (strings)

```
word = 'Python'

# ο πρώτος χαρακτήρας
print(word[0])

# ο δεύτερος χαρακτήρας
print(word[1])

print(word[2])
print(word[3])
print(word[5])

len(word)    # 6 μήκος, πλήθος χαρακτήρων
print(word[len(word) - 1])
```

[gist link](#)





0	1	2	3	4	5
P	y	t	h	o	n
-6	-5	-4	-3	-2	-1

```
word = 'Python'  
print(word[0])    # P  
print(word[-1])   # n
```

# Επιλογή τμήματος συμβολοσειράς

```
>>> word[1:4]
```

```
'yth'
```

```
>>> word[1:]
```

```
'ython'
```

```
>>> word[:4]
```

```
'Pyth'
```

```
>>> word[::2]
```

```
'Pto'
```

```
>>> word[::-1]
```

```
'nohtyP'      # αντιστροφή
```

## Τελεστές + \* in

```
# ένωση  
greeting = 'Hello ' + 'there'  
print(greeting)    # 'Hello there'
```

```
>>> 'a' * 10  
'aaaaaaaaaa'
```

```
>>> 'y' in 'Python'  
True  
>>> 'f' in 'Monty'  
False
```

# Προσπέλαση - for

```
word = "Python"  
  
for letter in word:  
    print(2 * letter, end="")  
  
# PPyytthhoonn
```

# format

```
"Γειά σου {}! Με λένε {}".format('Γιώργο', 'Διονύση')  
# 'Γειά σου Γιώργο! Με λένε Διονύση'
```

```
"Γειά σου {1}! Με λένε {0}".format('Γιώργο', 'Διονύση')  
# 'Γειά σου Διονύση! Με λένε Γιώργο'
```

# Μέθοδοι

```
'Monty-Python'.split('-')    # ['Monty', 'Python']  
  
'hello world'.replace('hello', 'goodbye')    # 'goodbye world'  
  
'hello world'.count('l')    # 3  
  
'Python'.startswith('P')    # True  
  
'python'.upper()            # 'PYTHON'  
'PYTHON'.lower()            # 'python'
```

# Λίστες (Lists)

```
colors = ['red', 'green', 'blue', 'yellow']  
  
a_list = []      # κενή λίστα  
  
whatever = ['abc', 4.56, [2,3], 'def', 6]  
print(whatever)
```

Περιέχει σε μια συγκεκριμένη σειρά μια συλλογή τιμών. Η ίδια τιμή μπορεί να υπάρχει περισσότερες από μια φορές. Μια λίστα μπορεί να περιέχει οποιουδήποτε τύπου αντικείμενα.

[gist link](#)





0	1	2	3
red	green	blue	yellow
-4	-3	-2	-1

```
colors = ['red', 'green', 'blue', 'yellow']
```

```
print(colors[0])    # 'red'  
print(colors[1])    # 'green'  
print(colors[-1])   # 'yellow'
```

```
>>> colors[1:3]
['green', 'blue']

>>> colors[1:]
['green', 'blue', 'yellow']

>>> colors[:2]
['red', 'blue']

>>> colors[::-1]
['yellow', 'blue', 'green', 'red']

>>> len(colors)
4
```

Παρουσιάζουν ομοιότητες με τις συμβολοσειρές.

## Τελεστές + \* in

```
>>> colors + ['orange', 'purple']  
['red', 'green', 'blue', 'yellow', 'orange', 'purple']  
  
>>> 2 * colors  
['red', 'green', 'blue', 'yellow', 'red', 'green', 'blue', 'ye  
  
>>> 'red' in colors  
True  
>>> 'brown' in colors  
False
```

# Προσθήκη / Αφαίρεση από τη λίστα

```
print(colors)
# ['red', 'green', 'blue', 'yellow']

colors.append('brown')

print(colors)
# ['red', 'green', 'blue', 'yellow', 'brown']

colors.remove('brown')

print(colors)
# ['red', 'green', 'blue', 'yellow']
```

## Διάτρεξη στοιχείων λίστας - for

```
for color in colors:  
    print(color, end=" ")  
  
# red green blue yellow  
  
for color in colors:  
    print(color.upper(), end=" ")  
  
# RED GREEN BLUE YELLOW
```

# Μέθοδοι

```
colors.sort()
# ['blue', 'green', 'red', 'yellow']

colors.pop()
# ['blue', 'green', 'red']

[2, 3, 4, 5, 4].count(4)      # 2

['blue', 'green', 'red'].insert(1, 'yellow')
# ['blue', 'yellow', 'green', 'red']
```





# list comprehension

```
# επιστρέφει καινούργια λίστα (χωρίς χρήση του for)
```

```
squares = [x**2 for x in range(0, 10)]  
print(squares)
```

```
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
squares = [x**2 for x in range(0, 10) if x > 5]  
print(squares)
```

```
# [36, 49, 64, 81]
```

# Πλειάδες (tuples)

```
a = (1 , 'asdf', 3.14)
print(a)

location = ('pos4work', 'patra', 'greece')
print(location[0])
location[0] = 'other' # error - δεν αλλάζει τιμή

for part in location:
    print(part, end=' ')
# pos4work patra greece
```

Έχει χαρακτηριστικά της λίστας αλλά δεν αλλάζει μέγεθος ούτε στοιχεία.  
Χρήσιμη για να επιστρέφουμε πολλές τιμές σε συναρτήσεις.

# Λεξικά (dictionaries)

```
prices = { 'milk': 3.67, 'bread': 1.67, 'cheese': 4.67 }  
print(prices)  
  
print(prices['milk'])  
  
# προσθήκη ζεύγους κλειδιού τιμής  
prices['butter'] = 1.95  
  
# διαγραφή στοιχείου  
del prices['butter']
```

Μπορούμε να αντιστοιχήσουμε σε λέξεις κλειδιά κάποιες τιμές. Αλλιώς και πίνακες κατακερματισμού (hash tables)



# Διάτρεξη τιμών

```
for food in prices:
    print('{} costs {}'.format(food, prices[food]))

# bread costs 1.67
# cheese costs 4.67
# milk costs 3.67

for k, v in prices.items():
    print(k, v)

# bread 1.67
# cheese 4.67
# milk 3.67
```

# Συναρτήσεις / Μέθοδοι

```
len(prices)      # 3

prices.keys()
list(prices.keys())    # ['bread', 'cheese', 'milk']

prices.values()
list(prices.values())  # [1.67, 4.67, 3.67]

'milk' in prices      # True
'beer' in prices      # False

if 'milk' in prices:
    print('milk costs', prices['milk'])
```

Exercise

## Σύνολα (sets)

```
basket = {'apple', 'orange', 'apple', 'orange', 'banana'}  
print(basket)  
# {'banana', 'orange', 'apple'}  
  
'orange' in basket      # True  
'pear' in basket       # False  
  
for fruit in basket:  
    print(fruit, end=' ')
```

Δομή δεδομένων που αναπαριστάται ως ένα μη διατεταγμένο σύνολο μοναδικών στοιχείων.



# Κλάσεις/Αντικείμενα

```
# δημιουργία κλάσης
class Vehicle:
    def __init__(self, number_of_wheels, fuel_type, maximum_ve
        self.number_of_wheels = number_of_wheels
        self.type_of_tank = fuel_type
        self.maximum_velocity = maximum_velocity

    def make_noise(self):
        print('VRUUUUUUUM')

tesla = Vehicle(4, 'electric', 250)
print(tesla.number_of_wheels)
```



```
class Vehicle:
    def __init__(self, number_of_wheels, fuel_type, maximum_ve
        self.number_of_wheels = number_of_wheels
        self.type_of_tank = fuel_type
        self.maximum_velocity = maximum_velocity

    def get_number_of_wheels(self):
        return self.number_of_wheels

    def set_number_of_wheels(self, number):
        self.number_of_wheels = number

bmw = Vehicle(4, 'petrol', 200)
bmw.set_number_of_wheels(2)
print(bmw.number_of_wheels)
```

# Modules

- Επαναχρησιμοποίηση κώδικα (συναρτήσεις, κλάσσεις)
- Κάθε αρχείο .py είναι και ένα module

```
import sys
print('Η μεταβλητή συστήματος PYTHONPATH έχει την τιμή', sys.p
```

## Packages / Πακέτα

- Οργάνωση modules
- Ιεραρχία  
(namespaces)

# The Python Standard Library

## Batteries Included

# Δημοφιλή πακέτα εκτός Standard Library

- requests, BeautifulSoup
- wxPython
- PIL, Pillow
- Numpy, SciPy, matplotlib
- nltk, scikit-learn
- SQLAlchemy
- django, flask

Awesome Python

## Resources

- [Python.org](#)
- [Python gr](#)
- [Byte of Python \(greek\)](#)
- [Learn Python the hard way](#)
- [Full Stack Python](#)
- Google is your friend



## Πηγές

- «Εισαγωγή στον Προγραμματισμό με Python», Βασίλης Βασιλάκης, Γιώργος Μπουκέας, 2015.
- «Οδηγός Python Μέσω Παραδειγμάτων», Δημήτρης Λεβεντέας, Python.org.gr
- The Python Tutorial, python.org
- Learning python from zero to hero