Ελληνικό Μεσογειακό Πανεπιστήμιο
Σχολή Μηχανικών / Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

HMY01K06
Επιστημονικός Προγραμματισμός με Python

Διάλεξη Δεύτερη
Εισαγωγή στην Python- Βασικές έννοιες

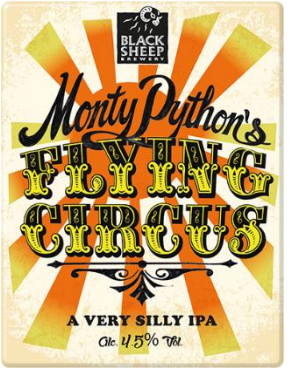
Φθινόπωρο 2025

Ιστορική Αναδρομή της Python

Ξεκίνησε να αναπτύσσεται το Δεκέμβριο του *1989* από τον *Guido van Rossum* στο ερευνητικό κέντρο *CWI* (Centrum voor Wiskunde en Informatica) της *Ολλανδίας*

- Έκδοση 1.0 ⇨ *1994* / Έκδοση 2.0 ⇨ *2000* / Έκδοση 3.0 ⇨ *2008*
- Πιο πρόσφατη έκδοση μέχρι σήμερα: ***Python 3.14.0*** *7/10/ 2025*

Το όνομα της γλώσσας προέρχεται από την τηλεοπτική σειρά του *BBC Monty Python's Flying Circus* (1969-74) της οποίας ο *Guido van Rossum* υπήρξε φανατικός οπαδός



Βασικά χαρακτηριστικά της Python

- Διερμηνευόμενη *(interpreted)*

  Το πρόγραμμα διερμηνεύεται και εκτελείται *γραμμή-προς-γραμμή*

  *Αντίθετα, ενας **compiler** μεταγλωττίζει πρώτα ολόκληρο το πρόγραμμα πριν το εκτελέσει*

- Αντικειμενοστρεφής *(object oriented)*

  Τα πάντα στην Python είναι αντικείμενα (με *ιδιότητες* και *μεθόδους χειρισμού*)

  *Εντούτοις, υποστηρίζει εξ-ίσου καλά και τον πιο 'κλασσικό' διαδικαστικό προγραμματισμό **(procedural)***

- Επεκτάσιμη *(extensible)* & *Ενσωματώσιμη (embeddable)*

  Μπορεί να *επεκταθεί* περιλαμβάνοντας κομμάτια κώδικα γραμμένα σε άλλη γλώσσα (πχ C++)

  Κομμάτια κώδικα γραμμένα σε Python μπορούν να *ενσωματωθούν* σε άλλη γλώσσα (πχ C++)

Βασικά πλεονεκτήματα της Python

- Απλή

  Απλή σύνταξη - *άμεσα κατανοητός* κώδικας

- Ισχυρή και επεκτάσιμη

  Τεράστιος αριθμός από *βιβλιοθήκες συναρτήσεων* για κάθε χρήση

  NumPy, sciPy, matplotlib, Tkinter, wxPython, requests, Pandas, PyGame, Pyglet …

  Επιστημονικός προγρ/σμός / Αριθμ. ανάλυση Ανάπτυξη διεπαφών GUI Χειρισμός HTTP Ανάλυση Δεδομένων Ανάπτυξη Παιχνιδιών

- Ανεξάρτητη πλατφόρμας *(platform independent)*

  Φορητά προγράμματα που μπορούν να τρέξουν σε οποιαδήποτε πλατφόρμα *Windows*, *Linux*, *MacOs*… *"από PlayStation μέχρι Supercomputer"*

- Προϊόν ελεύθερου και ανεξάρτητου λογισμικού *(open source)*

  Δωρεάν διάθεση, τεράστια *κοινότητα υποστήριξης*, συχνές *αναβαθμίσεις*

Μερικές από τις περιοχές εφαρμογών της Python

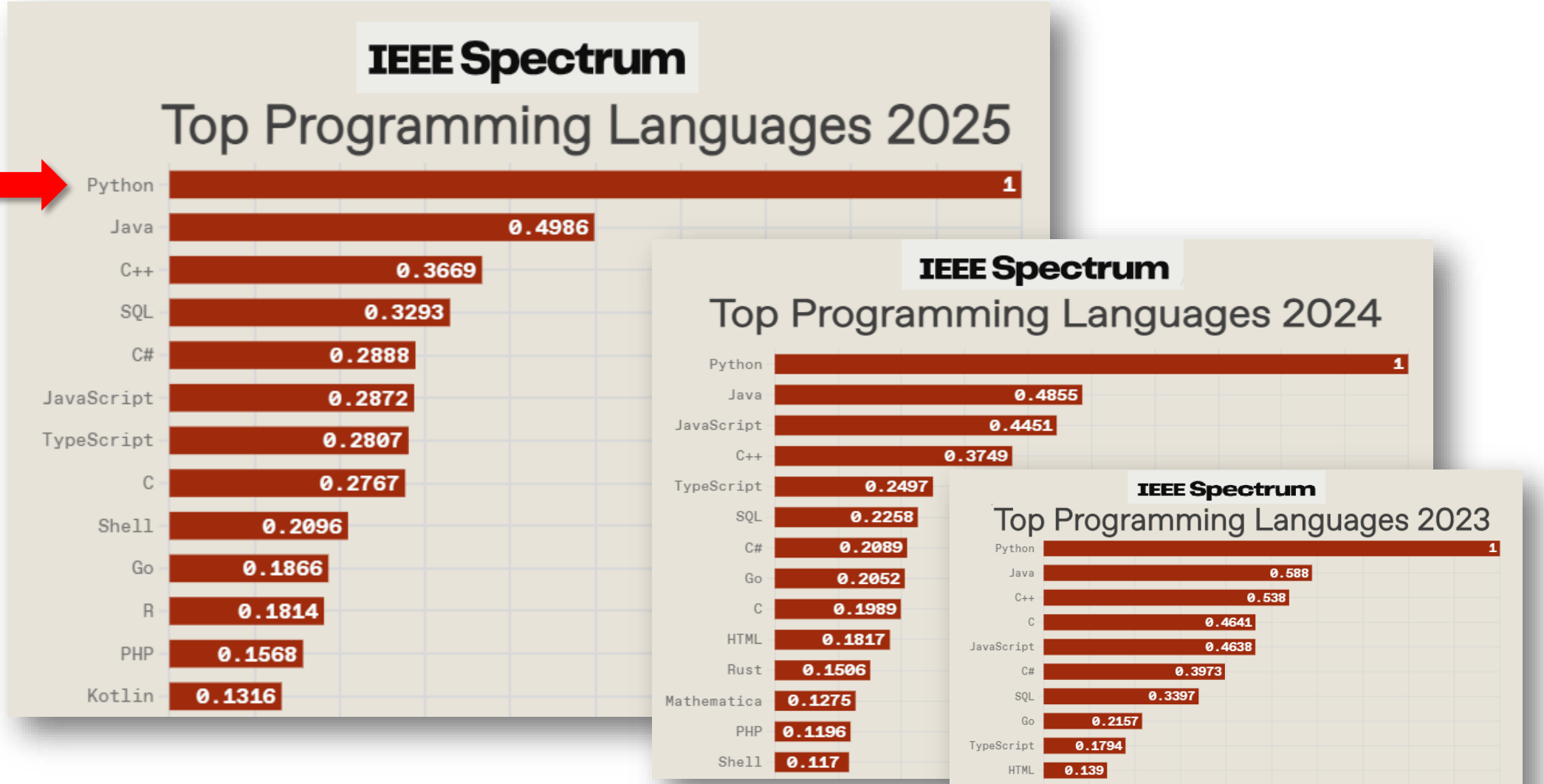
- Επιστημονικός προγραμματισμός / Αριθμητική ανάλυση
  (Scientific /numerical computing)
- Επιστήμη δεδομένων, μηχανική εκμάθηση και τεχνητή νοημοσύνη
  (Data science machine learning & Artificial Intelligence)
- Ανάπτυξη εφαρμογών Web
  (Web application development)
- Ανάπτυξη πρότυπου λογισμικού
  (Software prototyping)
- Ανάπτυξη διεπαφής χρήστη-υπολογιστή
  (GUI programming)
- Ανάπτυξη παιχνιδιών
  (Game programming)

Γιατί να μάθω Python;

- Εύκολη στην εκμάθηση
- Απλότητα και κομψότητα στη διατύπωση *(elegant syntax)*
- Ισχυρή και αποτελεσματική
- Μπορεί να χρησιμοποιηθεί παντού και να κάνει σχεδόν τα πάντα
- Τεράστια (και πολύ ενεργή) κοινότητα υποστήριξης
- Τεράστιος αριθμός βιβλιοθηκών (*200+* standard, πάνω από *70.000* συνολικά)

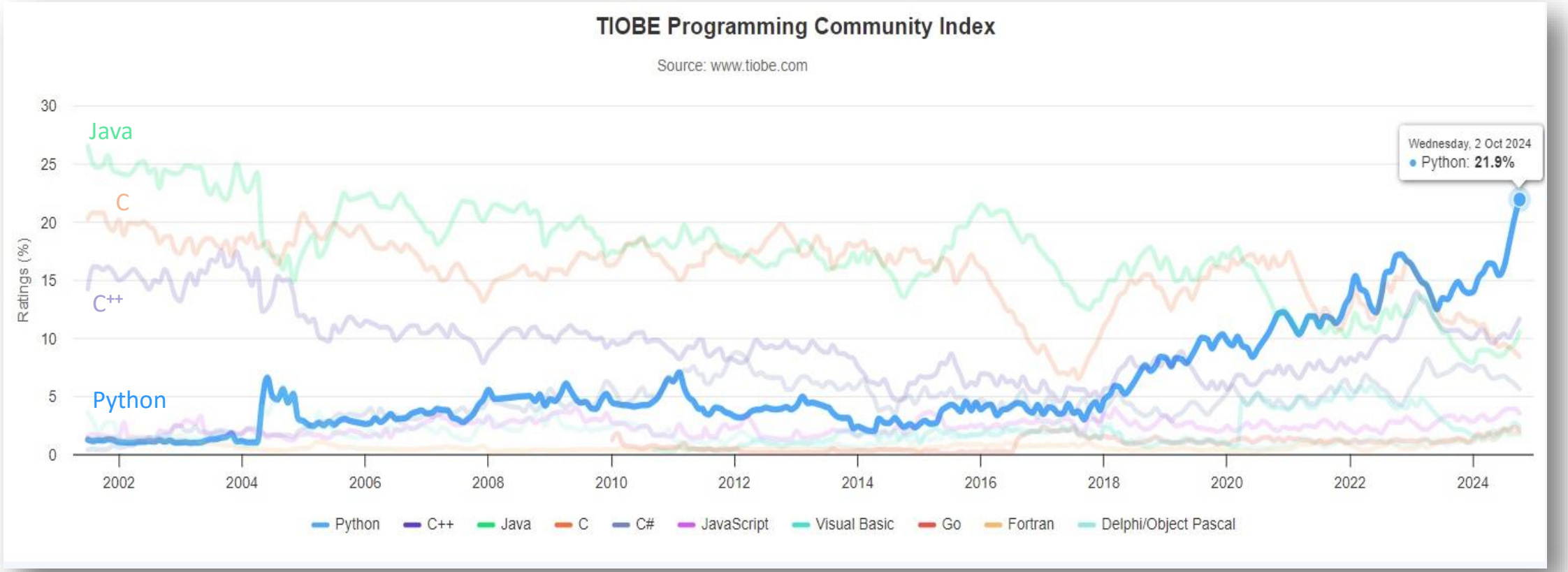
Για μια ακόμη χρονιά *(2025)* η Python βρίσκεται *στην κορυφή του Top-10* των γλωσσών προγραμματισμού

Top Programming Languages in 2025

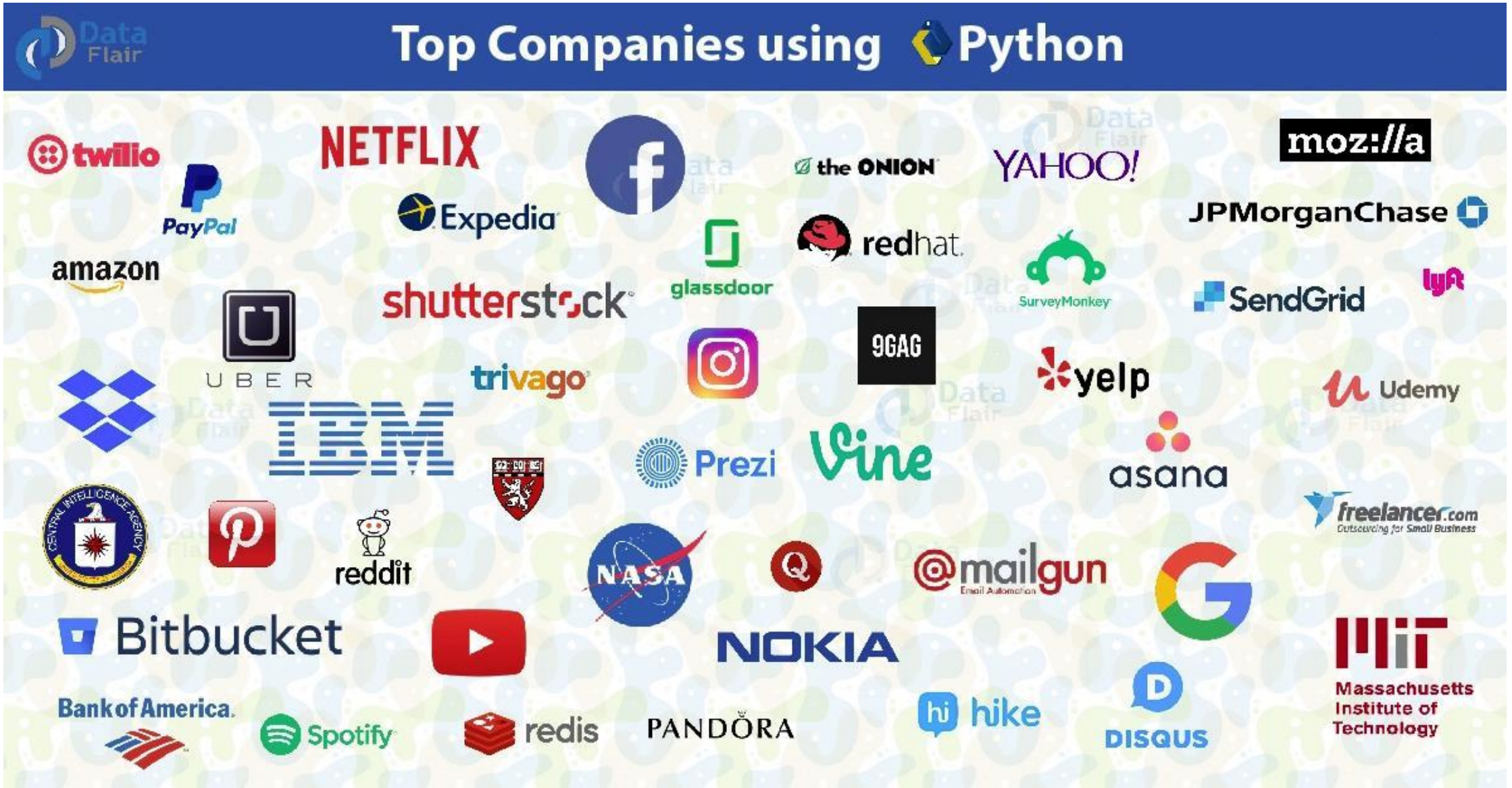


Πηγή: https://spectrum.ieee.org/top-programming-languages-2025

Top 10 Programming Languages in Oct 2024

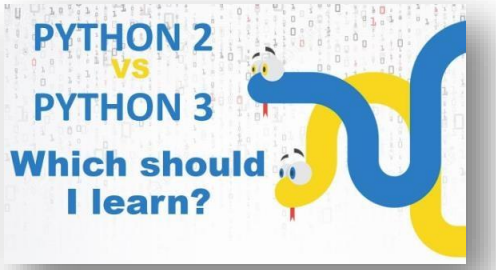


Python vs. Java vs. C / C++ την τελευταία 20ετία



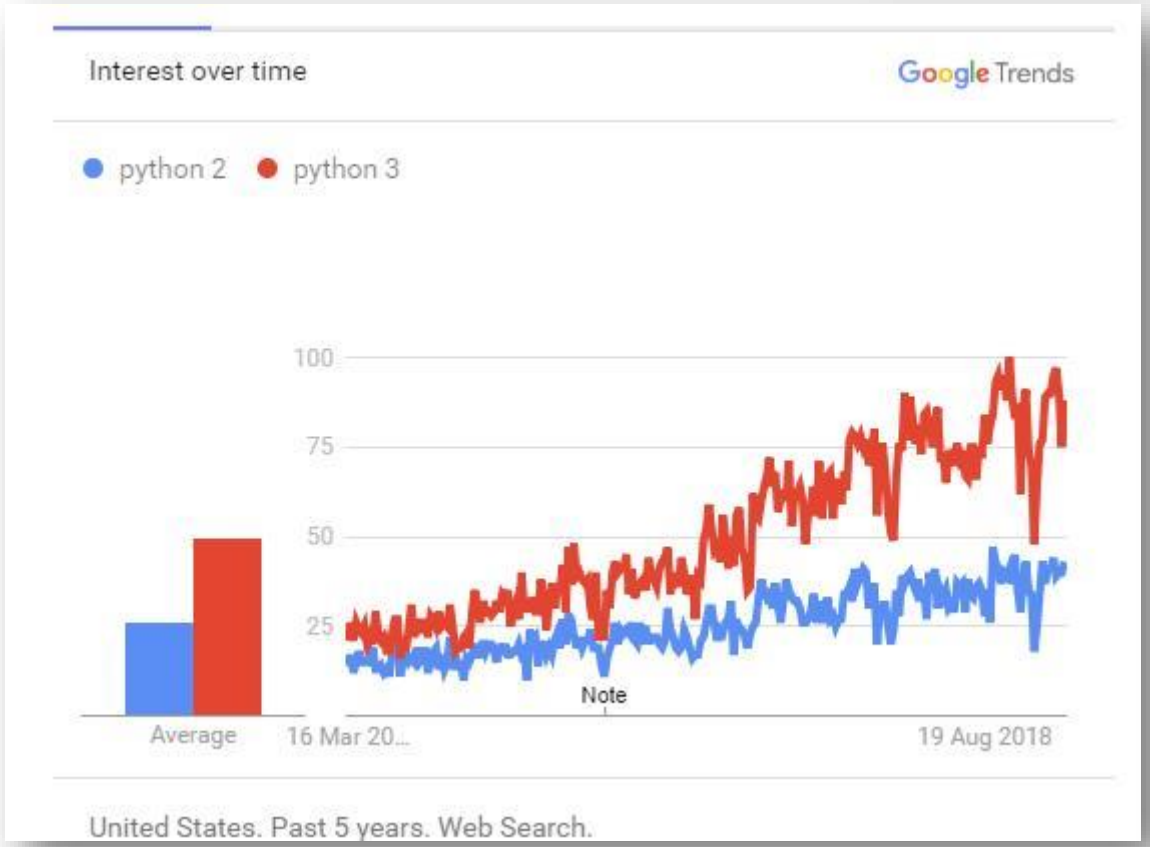
Πηγή: https://www.quora.com/Which-companies-use-Python-as-a-primary-language

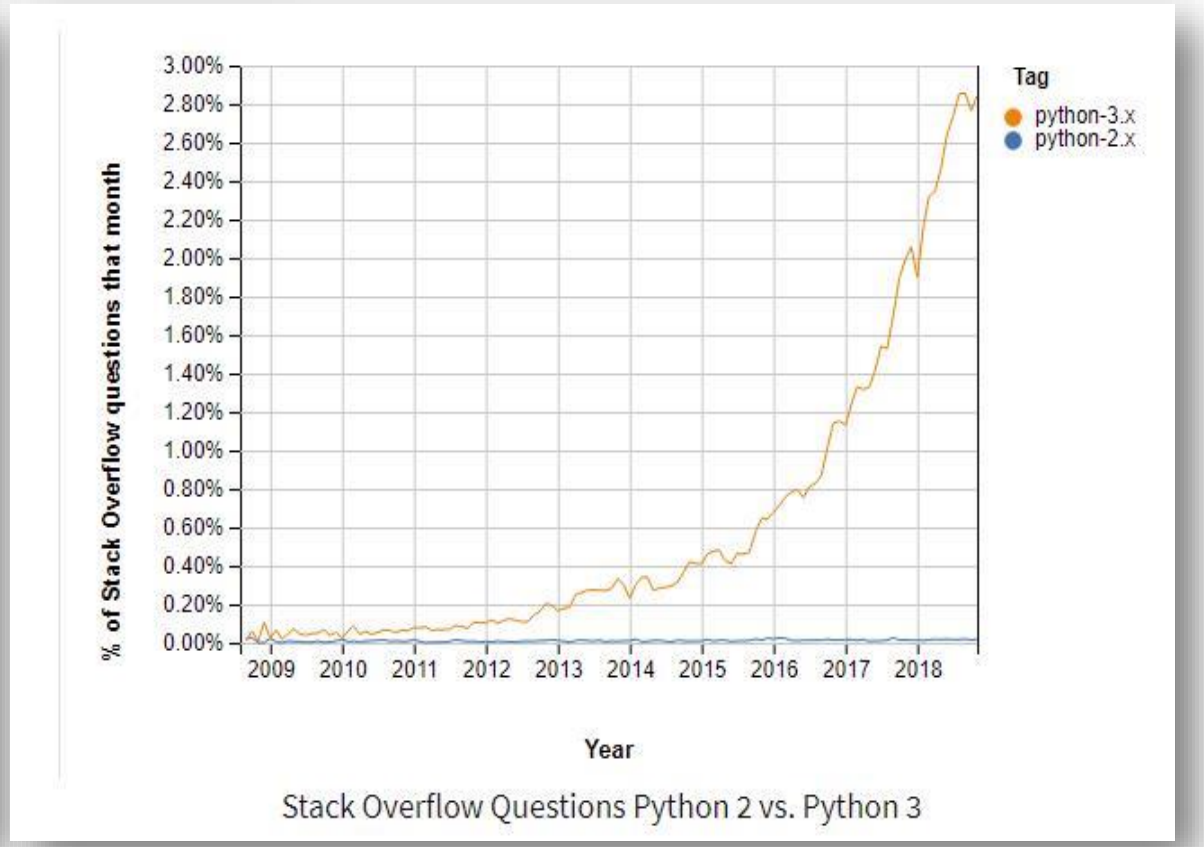
- *Η Python 2.0* εκδόθηκε για πρώτη φορά το *2000.* Η τελευταία της έκδοση ήταν η 2.7 που κυκλοφόρησε το *2010*
- *Η Python 3.0* εκδόθηκε για πρώτη φορά το *2008.* Η πιο πρόσφατη έκδοσή της μέχρι σήμερα (Οκτώβριος 2025) είναι η *3.14.0*
- Η Python 3 διαφέρει σε λίγα αλλά *κομβικά* σημεία από την 2, τα οποία δημιουργούν *μεγάλες ασυμβατότητες* ανάμεσά τους
- Για αρκετά χρόνια υπήρχε μια *έντονη δημόσια συζήτηση* στην κοινότητα των προγραμματιστών για το ποια Python ήταν καλύτερη 2 ή 3 *(πιο συγκεκριμένα Python 2.7 ή 3.5)*



- Η συζήτηση αυτή έχει λήξει εδώ και αρκετό καιρό, και η επιλογή είναι *ξεκάθαρα η Python 3*
- Ενας πιθανός λόγος να μάθει κανείς Python 2, είναι αν χρειάζεται να *διαχειριστεί ή να τροποποιήσει προγράμματα* ήδη γραμμένα στην 2 *(και υπάρχουν πάρα πολλά!)*

learntocodewith.me/programming/python/python-2-vs-python-3/





Αν ενδιαφέρεστε να μάθετε ποιες είναι οι διαφορές ανάμεσα στην Python 2 και 3 , μπορείτε να επισκεφτείτε διάφορες ιστοσελίδες, μια από τις οποίες είναι και η https://www.guru99.com/python-2-vs-python-3.html απ' όπου και τα δυο παραπάνω σχήματα

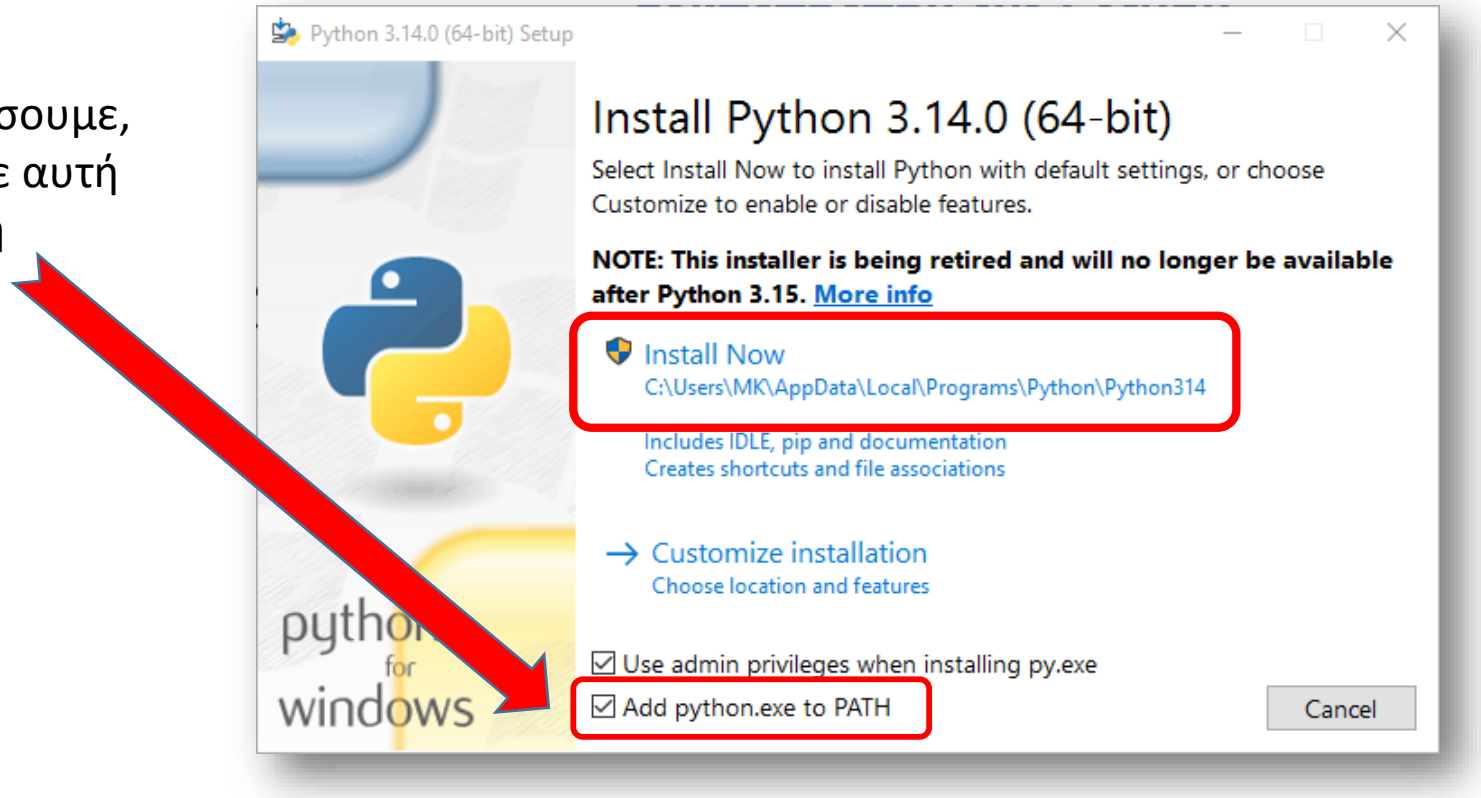
Εγκατάσταση της Python

1. Κατεβάζουμε από εδώ την πιο πρόσφατη έκδοση της Python
https://www.python.org/downloads/

Εγκατάσταση της Python

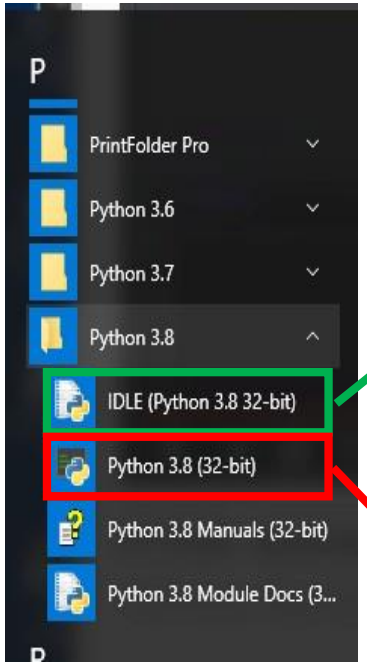
2. Τρέχουμε το πρόγραμμα εγκατάστασης

Πριν ξεκινήσουμε, τσεκάρουμε αυτή την επιλογή



Πως ξεκινάμε την Python μετά την εγκατάστασή της στα Windows

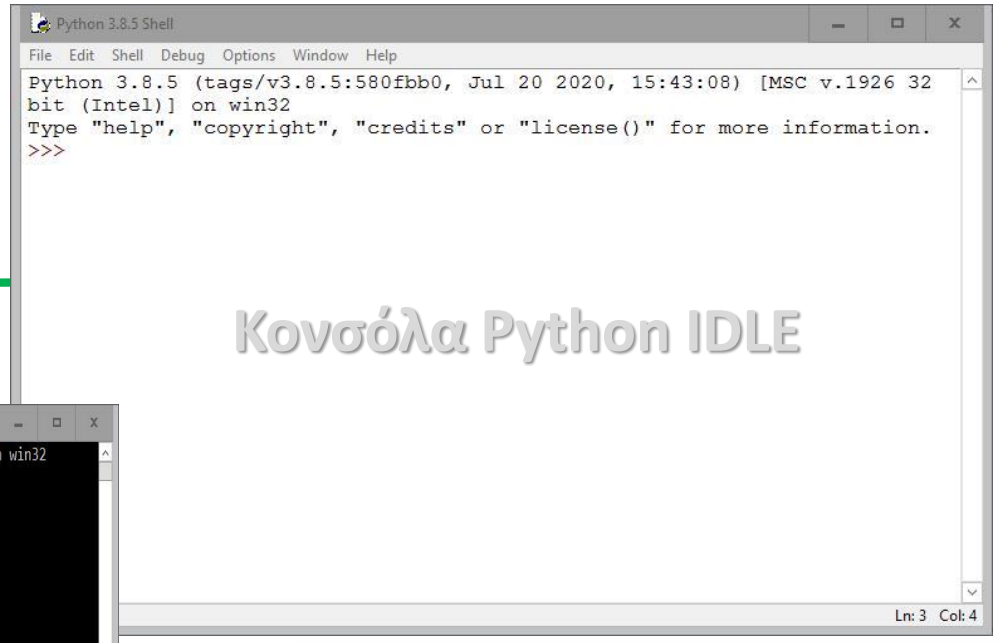
Από το Μενού Έναρξης - δυο εναλλακτικές επιλογές:

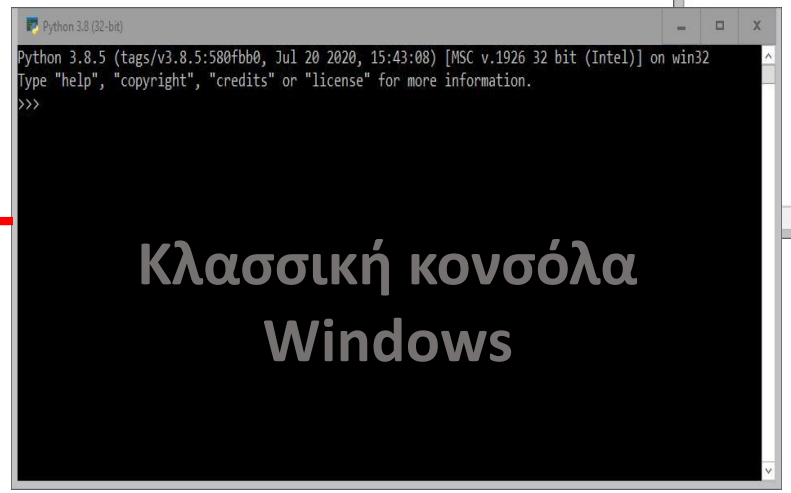


Μενού Έναρξης των Windows

1η επιλογή (προτεινόμενη)

2η επιλογή





*Το **IDLE** (συντομογραφία του **I**ntegrated **D**evelopment and **L**earning **E**nvironment) είναι ένα (πολύ απλό και βασικό) ολοκληρωμένο περιβάλλον ανάπτυξης για την γλώσσα προγραμματισμού Python που παρέχεται δωρεάν μαζί με την Python*

Ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDEs)

Integrated Development Environments

Περιβάλλοντα ανάπτυξης κώδικα που παρέχουν στον προγραμματιστή *πολλές δυνατότητες συγκεντρωμένες σε ένα μέρος* όπως *γράψιμο κώδικα*, *μεταγλώττιση*, *εντοπισμό σφαλμάτων*, *εκτέλεση*, *αυτόματη συμπλήρωση*, *βιβλιοθήκες προγραμμάτων* κλπ.

Στη βάση τους αποτελούνται από έναν *"έξυπνο" επεξεργαστή κώδικα* (*code* editor) ο οποίος *αναγνωρίζει το συντακτικό και τη δομή των εντολών της γλώσσας καθώς πληκτρολογούνται και παρέχει στον προγραμματιστή μια σειρά από εργαλεία για την διευκόλυνσή του όπως:*

- *χρωματική επισήμανση διαφορετικών στοιχείων της γλώσσας (εντολές, σταθερές…)*
- *δυνατότητα αυτόματης συμπλήρωσης λέξεων-κλειδιών (autocomplete)*
- *εντοπισμό και προτάσεις διόρθωσης συντακτικών λαθών*
- *εξελιγμένο σύστημα εύρεσης/αντικατάστασης*
- *διαχείριση μεγάλων και σύνθετων πακέτων προγραμμάτων (modules, packages)*
- ….

Για να γράψουμε -και να τρέξουμε- κώδικα σε Python *δεν είναι απαραίτητο* να διαθέτουμε ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης, όμως *βοηθάει πάρα-πάρα πολύ*

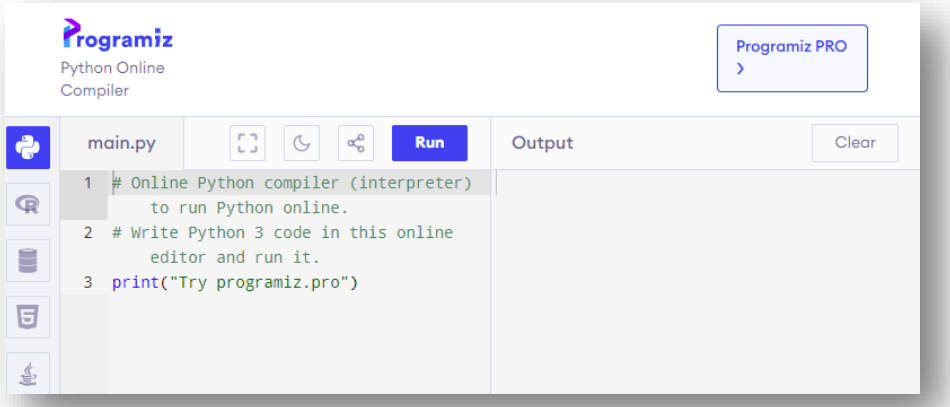
Ολοκληρωμένα περιβάλλοντα ανάπτυξης για την Python

Κυκλοφορούν πάρα πολλά – τόσο δωρεάν όσο και επί πληρωμή

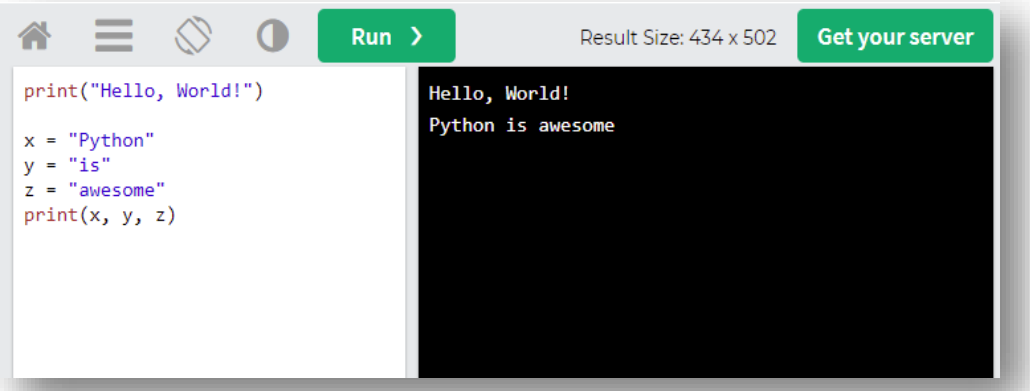
IDLE - Αυτό θα χρησιμοποιούμε στο μάθημα

Online Περιβάλλοντα ανάπτυξης για την Python

Παρέχονται από συγκεκριμένους δικτυακούς τόπους και ανοίγουν *μέσα σε ένα παράθυρο του browser*. Συνήθως υποστηρίζουν *πολλές διαφορετικές γλώσσες, προγραμματισμού*, ανάμεσά τους και την Python



www.programiz.com/python-programming/online-compiler/

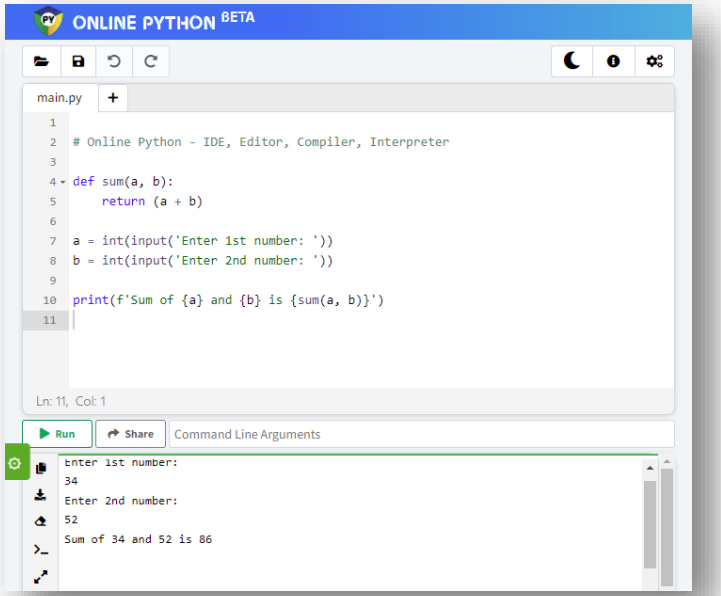


www.w3schools.com/python/trypython.asp?filename=demo_compiler

Μας προσφέρουν την δυνατότητα να *δημιουργήσουμε* ή να *μεταφορτώσουμε* και να *τρέξουμε* τον κώδικά μας και να τον *κατεβάσουμε πίσω* στον υπολογιστή μας

Κάποιοι από αυτούς μας προσφέρουν την δυνατότητα να *ανοίξουμε λογαριασμό* ώστε να διατηρούμε εκεί τα προγράμματά μας

Online Περιβάλλοντα ανάπτυξης για την Python



www.online-python.com

Αποτελούν εναλλακτική επιλογή αν (μεταξύ άλλων)

- *αδυνατούμε* - ή *δεν επιθυμούμε-* να εγκαταστήσουμε ένα ολοκληρωμένο περιβάλλον IDE στον υπολογιστή μας
- θέλουμε να δούμε κάτι *"στα γρήγορα"* στο *κινητό* ή το *tablet* μας ή σε *κάποιον υπολογιστή που δεν έχει Python*

Μερικά ακόμα sites με δωρεάν βασικές υπηρεσίες:

repl.it trinket.io ideone.com jdoodle.com pythonanywhere.com

Ποιό περιβάλλον/editor της Python να διαλέξω;

Πόσο καλά ξέρεις Python;

- Καθόλου ως λίγο – IDLE ή *online editors*
- Μέτρια - PyCharm, Sublime, Atom, Vs Code.
- Πολύ καλά - PyCharm, Vim, Emacs, Sublime, Atom, Vs Code.

Τι είδους εφαρμογές σε ενδιαφέρουν κυρίως;

- Web development — PyCharm Professional, VS Code
- Data Science — Spyder, Jupyter Notebook, PyCharm Professional
- Scripting — Sublime, Atom, PyCharm Community, Eclipse + PyDev
- Quality Assurance — Sublime, Atom, PyCharm Community, Jupyter Notebook

Τι λειτουργικό έχεις;

- Linux ή macOS– PyCharm, Sublime, Atom, Vim, Jupyter
- Windows- PyCharm, Sublime, Atom, Vs Code.
- Πολλαπλά Λ.Σ.- PyCharm, Vim, Emacs, Sublime, Atom, Vs Code.

https://www.geeksforgeeks.org/top-10-python-ide-and-code-editors-in-2020/

Περιβάλλον προγραμματισμού Python IDLE

Παρέχεται δωρεάν - περιέχεται ήδη στην εγκατάσταση της Python
Αποτελείται από δυο στοιχεία που *ανοίγουν σε ξεχωριστά παράθυρα*

1. την *κονσόλα* της Python για διαδραστική επικοινωνία
2. εναν *επεξεργαστή κώδικα* (*IDLE editor*) για την ανάπτυξη των προγραμμάτων

   Ο επεξεργαστής κώδικα αναγνωρίζει το συντακτικό της Python και ξεχωρίζει με διαφορετικά χρώματα τα διάφορα στοιχεία της (εντολές, συναρτήσεις, συμβολοσειρές κλπ), ενώ έχει και κάποιες άλλες απλές δυνατότητες όπως autocomplete και απλό συντακτικό έλεγχο

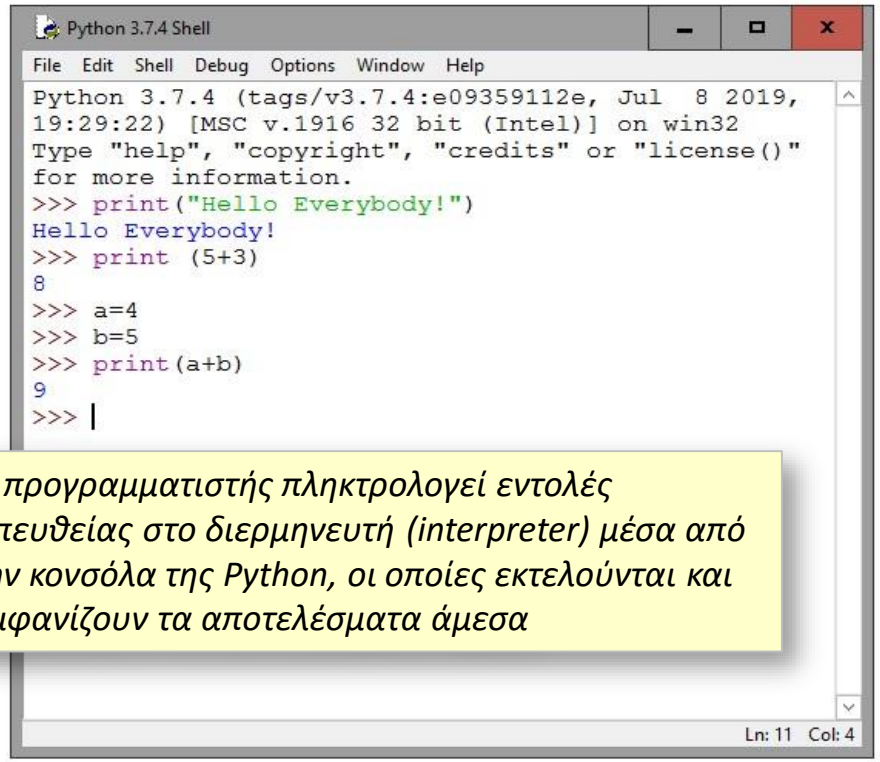
Το περιβάλλον IDLE αποτελεί μια καλή επιλογή για *αρχάριους προγραμματιστές* ώστε ή έμφαση να δίνεται κυρίως στην *εκμάθηση της γλώσσας* και όχι τόσο στο περιβάλλον προγραμματισμού

Σε επόμενη φάση, οι εξοικειωμένοι πλέον προγραμματιστές μπορούν να επιλέξουν από την μεγάλη -όπως είδαμε- *ποικιλία άλλων IDEs* (δωρεάν ή επί πληρωμή) ανάλογα με τις *προσωπικές τους ανάγκες* και το *γούστο τους*

Δυο διαφορετικοί τρόποι (modes) προγραμματισμού για την Python

Interactive mode programming

Διαδραστικός ή άμεσος προγραμματισμός

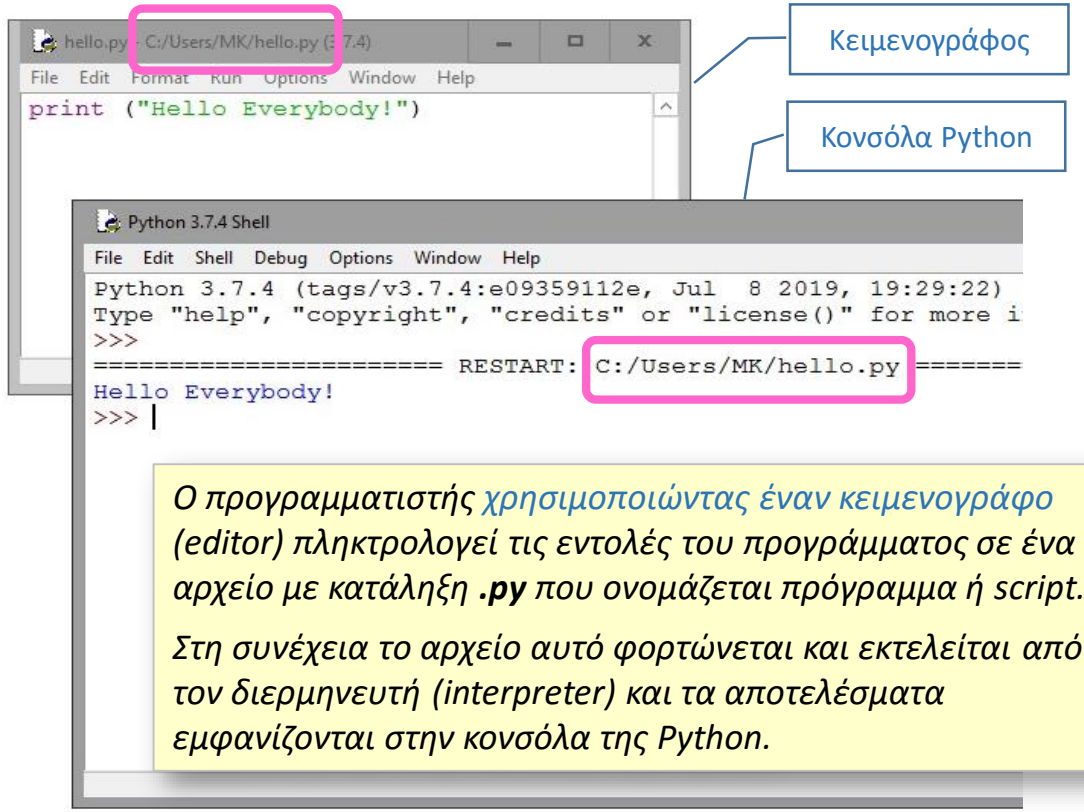


Ο προγραμματιστής πληκτρολογεί εντολές απευθείας στο διερμηνευτή (interpreter) μέσα από την κονσόλα της Python, οι οποίες εκτελούνται και εμφανίζουν τα αποτελέσματα άμεσα

Ενδείκνυται μόνο για απλές πράξεις ή μικρά προγράμματα με λίγες και απλές εντολές

Script mode Programming

Μέσω αρχείων εντολών (προγράμματα/scripts)

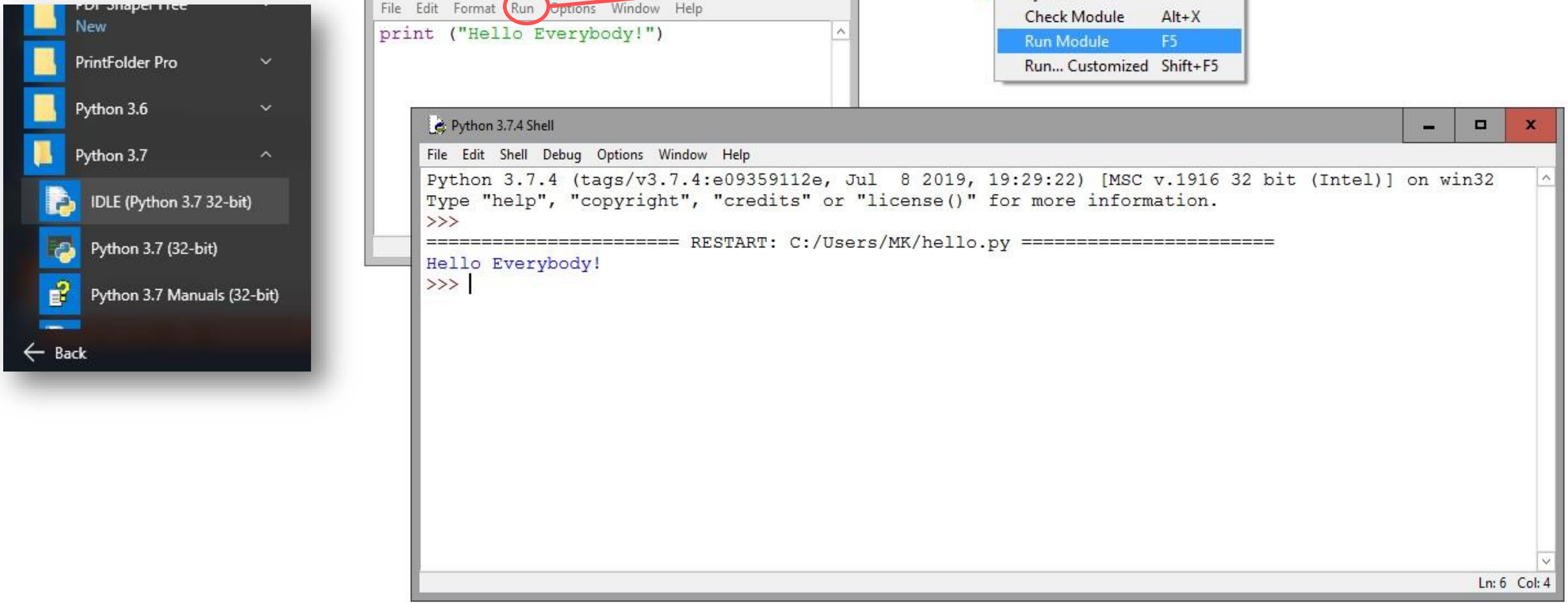


*Ο προγραμματιστής χρησιμοποιώντας έναν κειμενογράφο (editor) πληκτρολογεί τις εντολές του προγράμματος σε ένα αρχείο με κατάληξη **.py** που ονομάζεται πρόγραμμα ή script.*

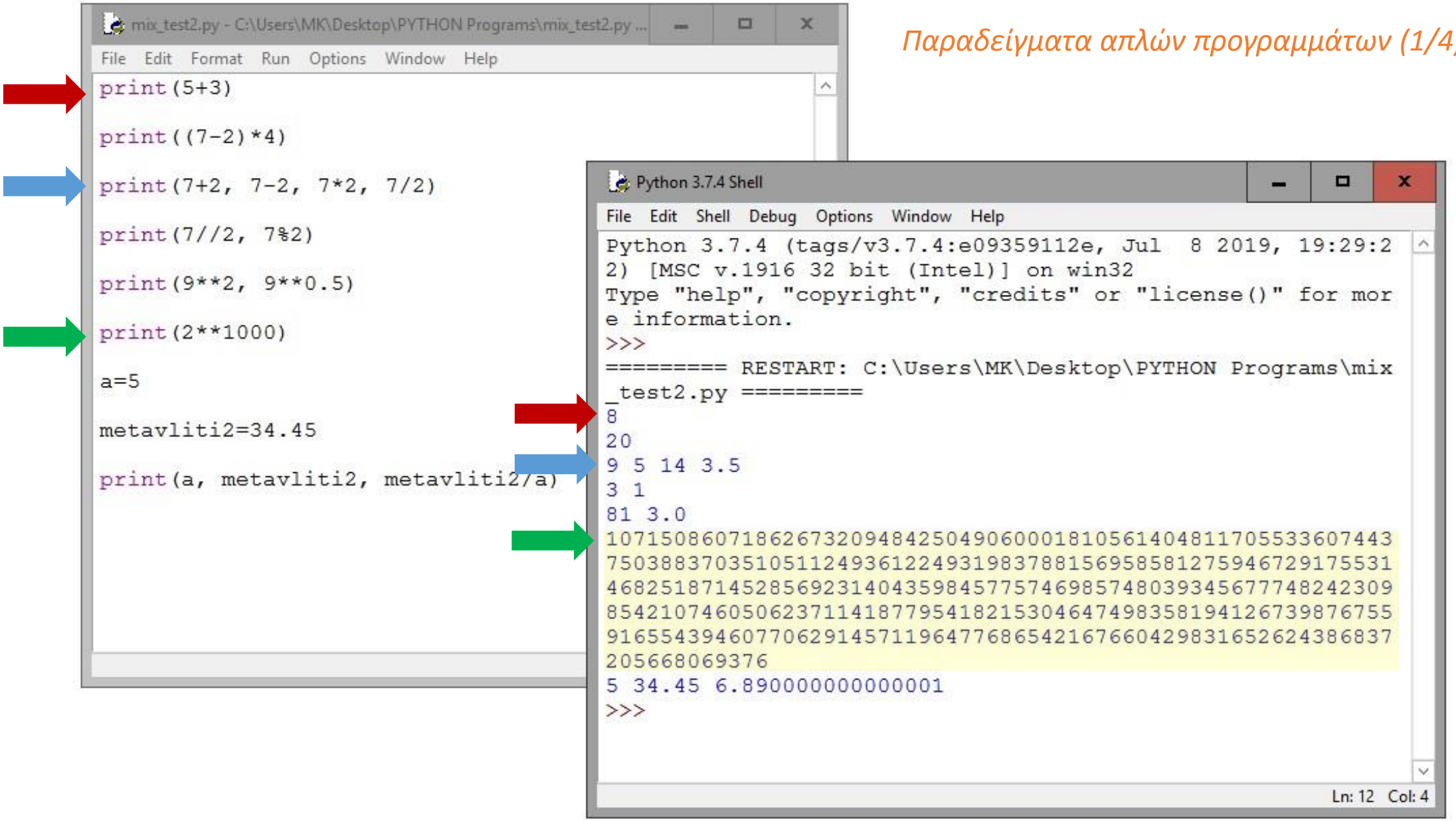
Στη συνέχεια το αρχείο αυτό φορτώνεται και εκτελείται από τον διερμηνευτή (interpreter) και τα αποτελέσματα εμφανίζονται στην κονσόλα της Python.

Αποτελεί τον βασικό τρόπο ανάπτυξης προγραμμάτων της Python

Το πρώτο μου πρόγραμμα σε Python



Παραδείγματα απλών προγραμμάτων (1/4)



Παραδείγματα απλών προγραμμάτων (2/4)

Calendar Example.py - C:\Users\MK\Desktop\PYTHON Programs\Calendar Example.py (3.7.4)

File Edit Format Run Options Window Help

```
# Create a plain text calendar

import calendar
c = calendar.TextCalendar(calendar.MONDAY)

str= c.formatyear(2019,2,1,6,3 )
''' w=2 (week col width)
    l=1 (line per cal row)
    c=6 (spaces between months)
    m=3 (months/row) '''

print (str)
```

*Calendar Month Example.py - C:/Users/MK/Desktop...

File Edit Format Run Options Window Help

```
import calendar
c = calendar.TextCalendar(calendar.MONDAY)
print (c.formatmonth(2019,10))
```

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
>>>
=== RESTART: C:/Users/MK/Desktop/PYTHON
Programs/Calendar Month Example.py ===
    October 2019
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

Ln: 280 Col: 4

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\MK\Desktop\PYTHON Programs\Calendar Example.py ======
                                2019

      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6                  1  2  3                  1  2  3
 7  8  9 10 11 12 13       4  5  6  7  8  9 10       4  5  6  7  8  9 10
14 15 16 17 18 19 20      11 12 13 14 15 16 17      11 12 13 14 15 16 17
21 22 23 24 25 26 27      18 19 20 21 22 23 24      18 19 20 21 22 23 24
28 29 30 31               25 26 27 28               25 26 27 28 29 30 31

      April                     May                       June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7                  1  2  3  4  5                  1  2
 8  9 10 11 12 13 14       6  7  8  9 10 11 12       3  4  5  6  7  8  9
15 16 17 18 19 20 21      13 14 15 16 17 18 19      10 11 12 13 14 15 16
22 23 24 25 26 27 28      20 21 22 23 24 25 26      17 18 19 20 21 22 23
29 30                     27 28 29 30 31            24 25 26 27 28 29 30

       July                    August                  September
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7                  1  2  3  4                        1
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       2  3  4  5  6  7  8
15 16 17 18 19 20 21      12 13 14 15 16 17 18       9 10 11 12 13 14 15
22 23 24 25 26 27 28      19 20 21 22 23 24 25      16 17 18 19 20 21 22
29 30 31                  26 27 28 29 30 31         23 24 25 26 27 28 29
                                                    30

      October                   November                  December
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6                  1  2  3                        1
 7  8  9 10 11 12 13       4  5  6  7  8  9 10       2  3  4  5  6  7  8
14 15 16 17 18 19 20      11 12 13 14 15 16 17       9 10 11 12 13 14 15
21 22 23 24 25 26 27      18 19 20 21 22 23 24      16 17 18 19 20 21 22
28 29 30 31               25 26 27 28 29 30         23 24 25 26 27 28 29
                                                    30 31

>>>
```

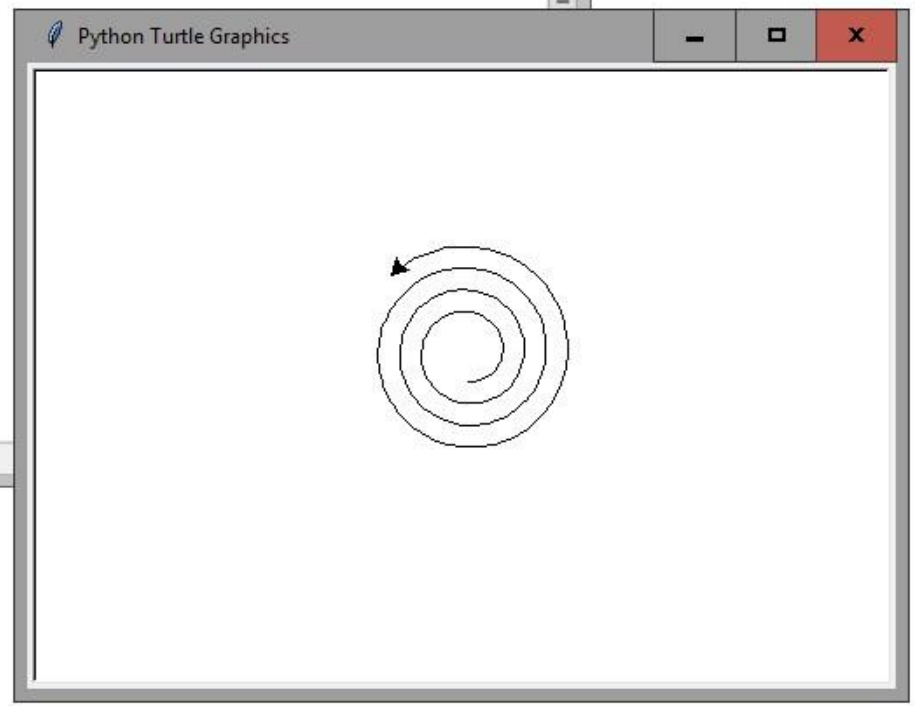
Ln: 41 Col: 4

Παραδείγματα απλών προγραμμάτων (3 /4)

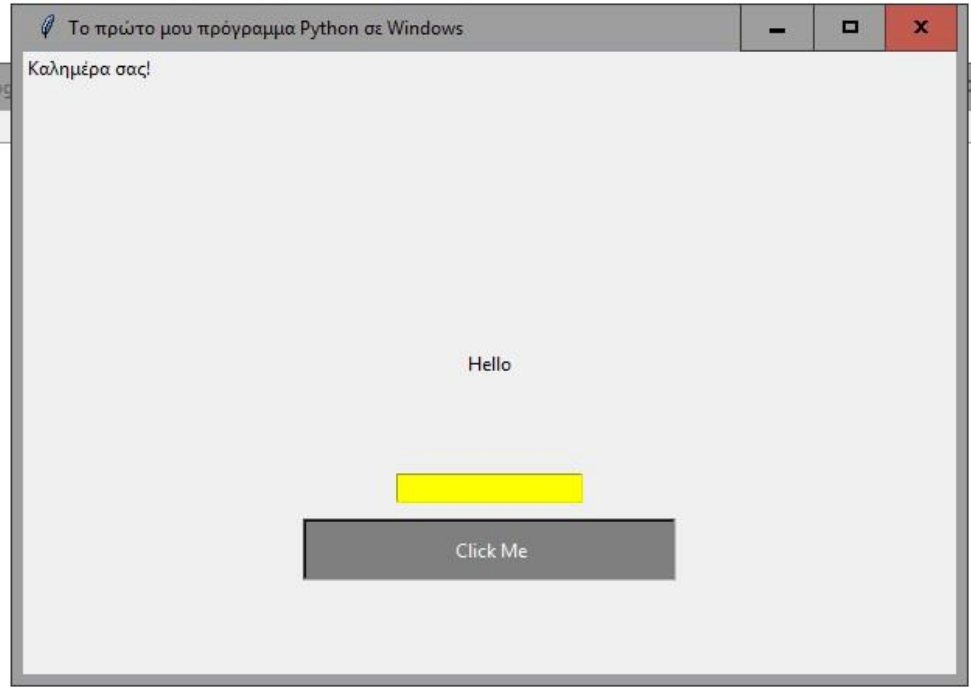
Turtle.py - C:\Users\MK\Desktop\PYTHON Programs\Turtle.py (3.7.4)

File Edit Format Run Options Window Help

```
import turtle

def draw_spiral(t, n, length=3, a=0.1, b=0.0002):
    """Draws an Archimedian spiral starting at the origin.

    Args:
      n: how many line segments to draw
      length: how long each segment is
      a: how loose the initial spiral starts out (larger is looser)
      b: how loosly coiled the spiral is (larger is looser)

    http://en.wikipedia.org/wiki/Spiral
    """
    theta = 0.0

    for i in range(n):
        t.fd(length)
        dtheta = 1 / (a + b * theta)

        t.lt(dtheta)
        theta += dtheta

# create the world and bob
bob = turtle.Turtle()
draw_spiral(bob, 300)

turtle.mainloop()
```



Παραδείγματα απλών προγραμμάτων (4/4)



Yet Another Simple Window Program.py - C:\Users\MK\Desktop\PYTHON Programs\Yet Another Simple Window Prog

File Edit Format Run Options Window Help

```
from tkinter import *
from tkinter import messagebox
from tkinter import simpledialog

window = Tk()

window.title("Το πρώτο μου πρόγραμμα Python σε Windows")

window.geometry('600x400')

lbl = Label(window, text="Hello")
lbl.place(relx=0.5, rely=0.5, relwidth=0.2, anchor="center")


txt = Entry(window,width=30, fg="blue",  bg="yellow" )
txt.place(relx=0.5, rely=0.7, relwidth=0.2, anchor="center")

l1=Label(window,text="Καλημέρα σας!")
l1.grid(column=4)

def clicked():

    lbl.configure(text="Button was clicked !!")
    info = messagebox.showinfo('Click Button', 'Button was clicked', parent=window)
    #okcancel = messagebox.askokcancel('Question Title', 'Do you want to perform this action?', parent=window) # OK / Cancel
    #retrycancel = messagebox.askretrycancel('Question Title', 'Do you want to retry?', parent=window) # Retry / Cancel
    #yesno = messagebox.askyesno('Question Title', 'Are you sure you want to undo?', parent=window) # Yes / No
    #yesnocancel = messagebox.askyesnocancel('Question Title', 'Are you sure you want to undo?', parent=window) # Yes / No / Cancel

btn = Button(window, text="Click Me", command=clicked, bg="gray50", fg="white", relief=SUNKEN)
btn.place(relx=0.5, rely=0.8, anchor="center", relwidth=0.4, relheight=0.1)

window.mainloop()
```

Ln: 16 Col: 43

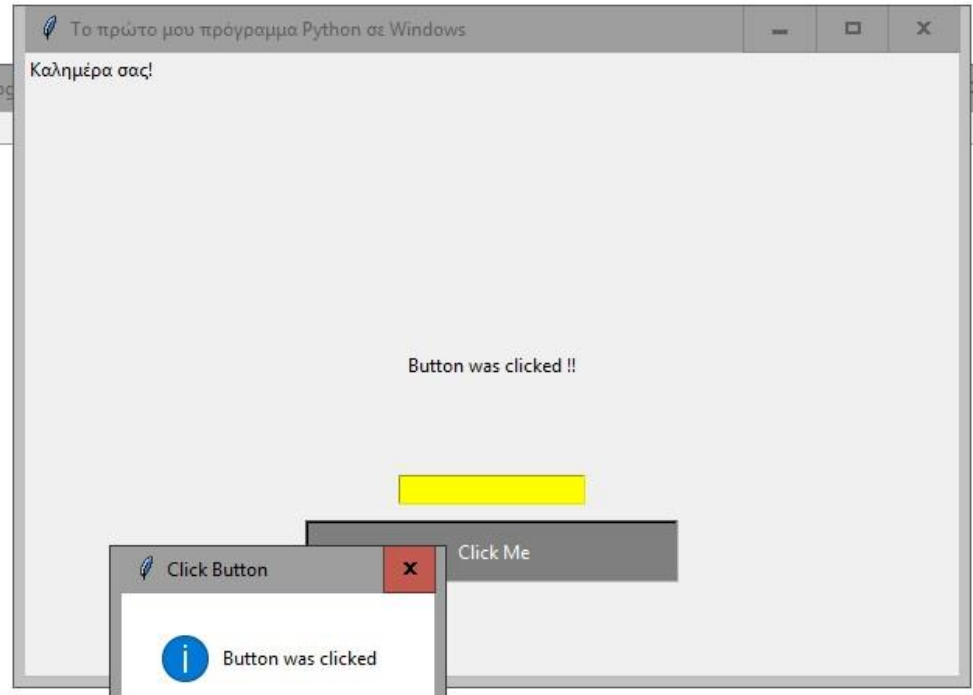
Παραδείγματα απλών προγραμμάτων (4/4)

Yet Another Simple Window Program.py - C:\Users\MK\Desktop\PYTHON Programs\Yet Another Simple Window Prog

File Edit Format Run Options Window Help

```
from tkinter import *
from tkinter import messagebox
from tkinter import simpledialog

window = Tk()

window.title("Το πρώτο μου πρόγραμμα Python σε Windows")

window.geometry('600x400')

lbl = Label(window, text="Hello")
lbl.place(relx=0.5, rely=0.5, relwidth=0.2, anchor="center")


txt = Entry(window,width=30, fg="blue",  bg="yellow" )
txt.place(relx=0.5, rely=0.7, relwidth=0.2, anchor="center")

l1=Label(window,text="Καλημέρα σας!")
l1.grid(column=4)

def clicked():

    lbl.configure(text="Button was clicked !!")
    info = messagebox.showinfo('Click Button', 'Button was clicked', pa
    #okcancel = messagebox.askokcancel('Question Title', 'Do you want to perform this action?', parent=window) # OK / Cancel
    #retrycancel = messagebox.askretrycancel('Question Title', 'Do you want to retry?', parent=window) # Retry / Cancel
    #yesno = messagebox.askyesno('Question Title', 'Are you sure you want to undo?', parent=window) # Yes / No
    #yesnocancel = messagebox.askyesnocancel('Question Title', 'Are you sure you want to undo?', parent=window) # Yes / No / Cancel

btn = Button(window, text="Click Me", command=clicked, bg="gray50", fg="white", relief=SUNKEN)
btn.place(relx=0.5, rely=0.8, anchor="center", relwidth=0.4, relheight=0.1)

window.mainloop()
```

Ln: 16 Col: 43



Βασικές Έννοιες της Python

Βασικές έννοιες της Python

- Αντικείμενα *(objects)*

  Στιγμιότυπα προκαθορισμένων κλάσεων / τύπων της Python

  - Οι βασικότερες κλάσεις έχουν *οριστεί εκ των προτέρων* και είναι *διαθέσιμες εξαρχής*
  - Η Python μας δίνει την δυνατότητα να *δημιουργήσουμε δικές μας κλάσεις* εφόσον χρειαστεί

- Λέξεις-κλειδιά *(keywords)*

  Σύνολο π*ροκαθορισμένων* λέξεων με *ειδική σημασία* και λειτουργικότητα

  - Στην πιο πρόσφατη έκδοσή της (3.12) η Python περιέχει 35 λέξεις-κλειδιά

- Προσδιοριστές *(identifiers)*

  Μοναδικά ονόματα που δίδονται σε *αντικείμενα* από τον προγραμματιστή

  - Ο σχηματισμός τους υπακούει σε *συγκεκριμένους κανόνες*
  - Δεν μπορεί να ταυτίζονται με λέξεις-κλειδιά

Η έννοια του αντικειμένου στην Python

- Η Python είναι μια *αμιγώς αντικειμενοστρεφής* γλώσσα προγραμματισμού
- Όλα *ανεξαιρέτως* τα στοιχεία της Python αποτελούν αντικείμενα

πχ. μια συνάρτηση, μια συμβολοσειρά, ένα αρχείο, μια λίστα, ενας μιγαδικός αριθμός, ενας ακέραιος, όλα αυτά είναι αντικείμενα στην Python.

- Σκεφτείτε ενα αντικείμενο σαν ένα *μίνι αυτόνομο πρόγραμμα* μέσα στην Python, με το δικό του σύνολο *ιδιοτήτων* και *μεθόδων χειρισμού*

*πχ. ένα **αρχείο** έχει ιδιότητες όπως μέγεθος, ημερομηνία, είδος περιεχομένου (κείμενο, δυαδικό) και μεθόδους όπως άνοιγμα, κλείσιμο, διάβασμα, γράψιμο κλπ.*

*πχ. ενας **μιγαδικός αριθμός** έχει ιδιότητες όπως τιμή πραγματικού μέρους, τιμή φανταστικού μέρους και μεθόδους όπως υπολογισμός μέτρου, υπολογισμός συζυγούς κλπ.*

Κλάσεις αντικειμένων

- Όλα τα αντικείμενα που χαρακτηρίζονται από το *ίδιο σύνολο ιδιοτήτων* και *μεθόδων χειρισμού* αποτελούν *(ομαδοποιούνται, συνιστούν)* μια *κλάση* αντικειμένων που ονομάζεται ***τύπος αντικειμένου*** *(object type)*
- Κάθε αντικείμενο αποτελεί *μέλος / εκπρόσωπο / στιγμιότυπο (instance)* της κλάσης *(ή αλλιώς του τύπου)* στον οποίο ανήκει
- Χρησιμοποιώντας την συνάρτηση type() μπορούμε άμεσα να διαπιστώσουμε σε ποια κλάση ανήκει ένα αντικείμενο

```
>>> type("hello")
<class 'str'>
>>> type([1,2,8])
<class 'list'>
>>> type({1:"a", 2:"b"})
<class 'dict'>
>>> type (2+3j)
<class 'complex'>
>>> type(2.7)
<class 'float'>
>>> type(2023)
<class 'int'>
>>>
```

Μας λέει ότι το αντικείμενο "hello" ανήκει στην κλάση αντικειμένων str (string – συμβολοσειρά)

Μας λέει ότι το αντικείμενο {1:"a", 2:"b"} ανήκει στην κλάση αντικειμένων dict (dictionary – συμβολοσειρά)

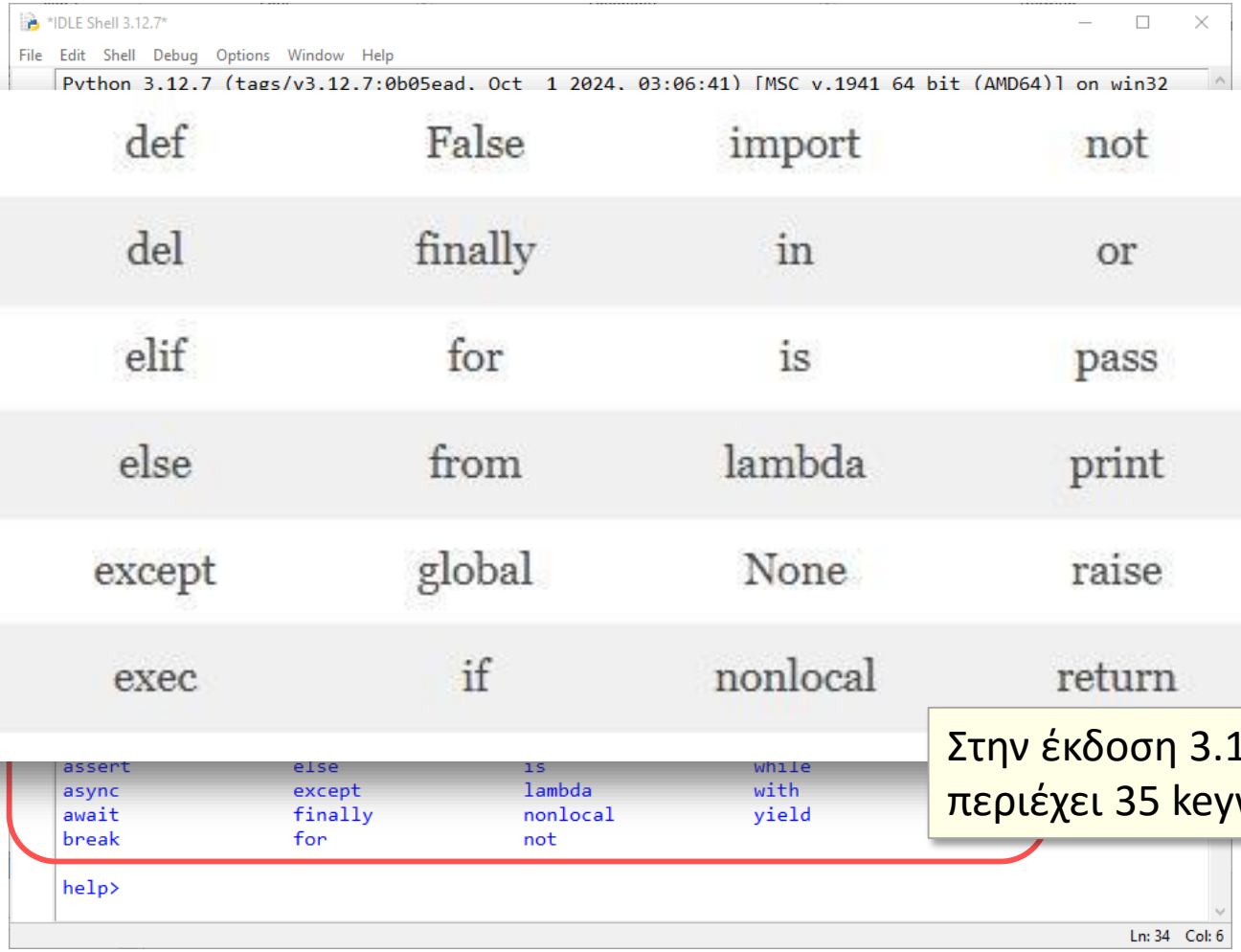
Ακόμα και η συνάρτηση type() έχει τον δικό της τύπο που είναι 'type'

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

Λέξεις κλειδιά (keywords) της Python

Προκαθορισμένες λέξεις με ειδική σημασία και λειτουργικότητα
Δεν μπορούν να χρησιμοποιηθούν σαν προσδιοριστές

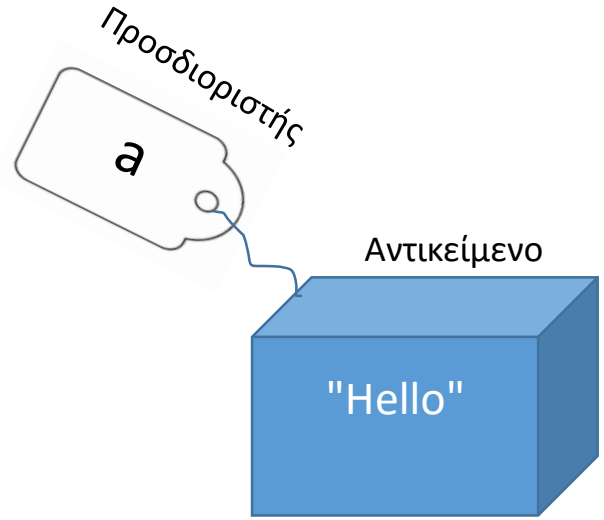
and	def	False	import	not	True
as	del	finally	in	or	try
assert	elif	for	is	pass	while
break	else	from	lambda	print	with
class	except	global	None	raise	yield
continue	exec	if	nonlocal	return	



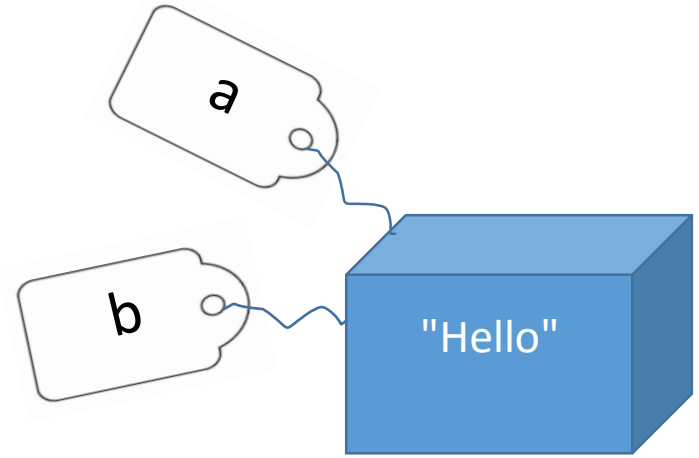
Στην έκδοση 3.14 η Python περιέχει 35 keywords

Η έννοια του προσδιοριστή (identifier)

- Κάθε αντικείμενο μπορεί να επισημαίνεται *(be labeled)* από έναν ή περισσότερους *προσδιοριστές (identifiers)* δηλαδή *ονόματα* που δημιουργούνται από τον προγραμματιστή σύμφωνα με *ειδικούς κανόνες (θα τους δούμε αμέσως μετά)*




```
a = "Hello"
```




```
a = "Hello"
b = "Hello"
```

ή εναλλακτικά:

```
a = "Hello"
b = a
```

Κανόνες δημιουργίας προσδιοριστών στην Python

- Οι μόνοι επιτρεπτοί χαρακτήρες για την δημιουργία ενός προσδιοριστή είναι
  - τα *γράμματα της αλφαβήτου κεφαλαία και πεζά* (*),
  - τα *αριθμητικά ψηφία 0-9*
  - ο χαρακτήρας *underscore* (_)
- Υπάρχει διάκριση κεφαλαίων πεζών (πχ ***SUM***, ***Sum*** , ***sum*** είναι *όλα διαφορετικά*)
- Ο *πρώτος χαρακτήρας* ενός προσδιοριστή δεν μπορεί να είναι *αριθμητικό ψηφίο*
- *Δεν υπάρχει (θεωρητικός) περιορισμός στο μήκος* ενός προσδιοριστή στην Python
- Οι *λέξεις κλειδιά* δεν μπορούν *-προφανώς-* να χρησιμοποιηθούν ως προσδιοριστές

(*) *Python 2: μόνο Αγγλικό αλφάβητο / Python 3: (υποστηρίζει. Unicode) οποιαδήποτε γλώσσα (άρα και Ελληνικά)*

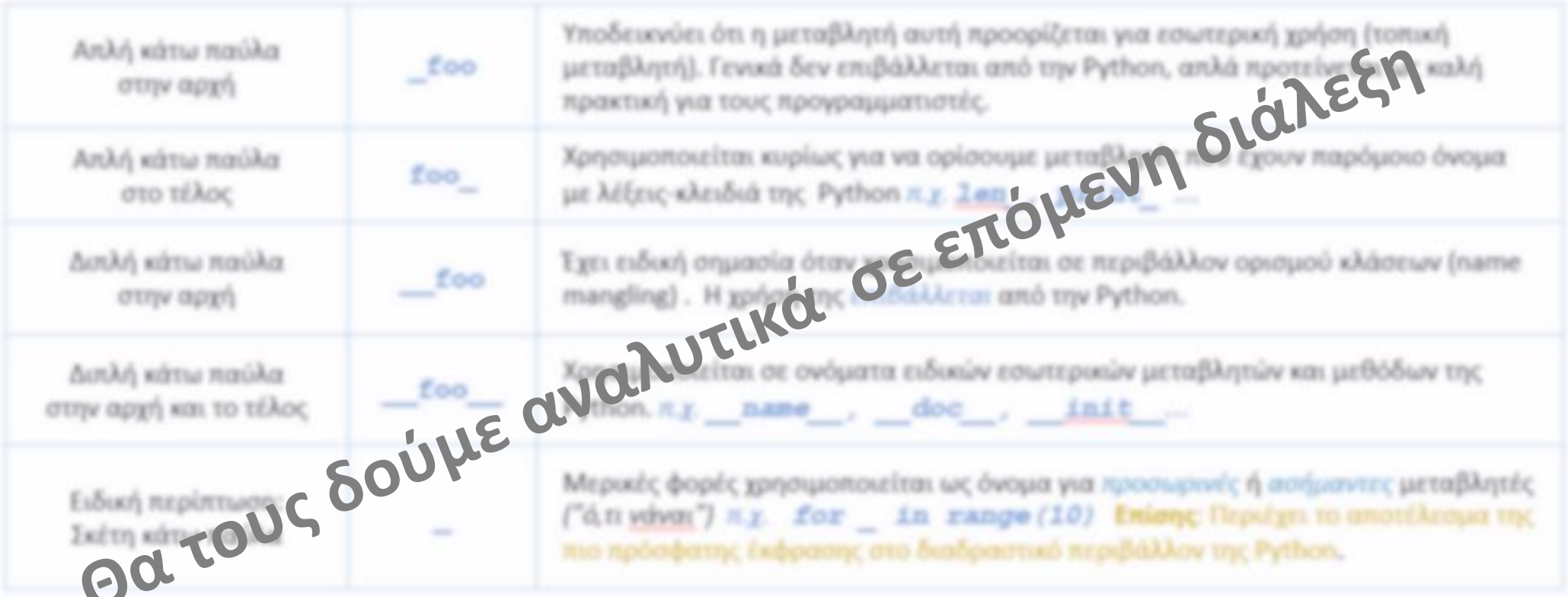
Παραδείγματα προσδιοριστών της Python

Επιτρεπτοί

```
balance
currentBalance
current_Balance
Global
_spam
account14
```

Ειδικές κατηγορίες προσδιοριστών

Οι προσδιοριστές που αρχίζουν ή/και τελειώνουν με κάτω παύλα ή διπλή κάτω παύλα (*double underscore - **dunder***) προορίζονται συνήθως για ειδική χρήση – αποφεύγουμε να τους χρησιμοποιούμε σε απλά προγράμματα



Θα τους δούμε αναλυτικά σε επόμενη διάλεξη

Η έννοια της "μεταβλητής" στην Python

Στις περισσότερες γλώσσες προγραμματισμού κάθε μεταβλητή που ορίζεται *έχει συγκεκριμένο τύπο που δεν μπορεί να αλλάξει*

πχ C++: `int A;` Το **A** *δηλώνεται στην αρχή* και παραμένει *ακέραια* μεταβλητή σε όλη τη διάρκεια εκτέλεσης του προγράμματος

`A = 5` Είναι ok

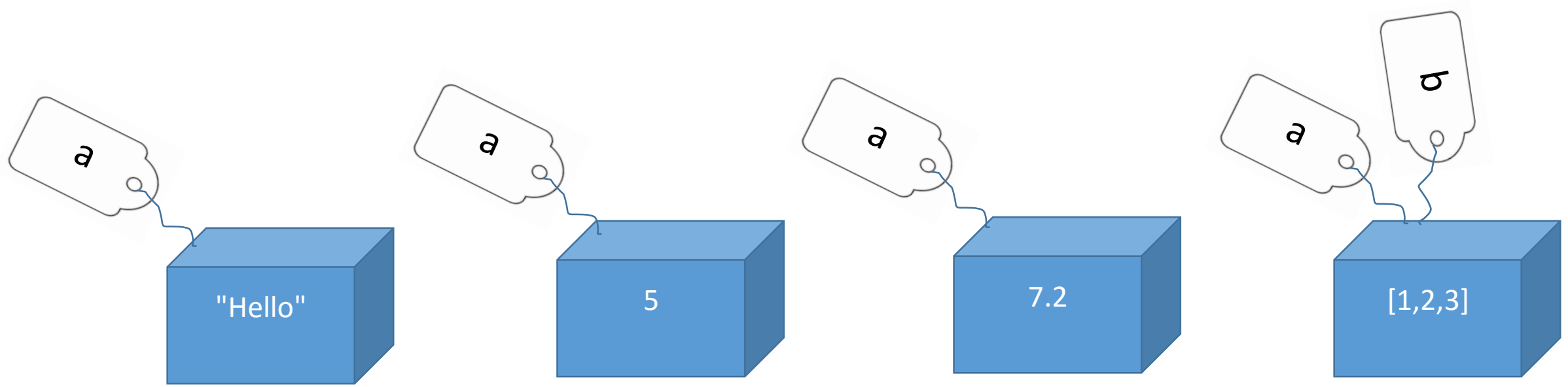
`A = 7.2` Συντακτικό λάθος

Αυτό δεν ισχύει στην Python. Τα στοιχεία που σε άλλες γλώσσες έχουμε συνηθίσει να ονομάζουμε "μεταβλητές", εδώ *είναι απλά προσδιοριστές (δηλαδή ετικέτες) που δίνονται από τον προγραμματιστή σε αντικείμενα* όπως είδαμε νωρίτερα

Στην πορεία του προγράμματος μπορούμε *χωρίς κανένα πρόβλημα* να πάρουμε μια ετικέτα από ένα αντικείμενο και να την μεταφέρουμε σε κάποιο άλλο

Έτσι πχ. `A = 5` `A = 7.2` `A = "Hello"` είναι όλα σωστά στην Python

Η έννοια της "μεταβλητής" στην Python




```
a = "Hello"
a = 5
a = 7.2
a = [1,2,3]
b = [1,2,3]
```

(ή εναλλακτικά `b = a`)

Κλάσεις (τύποι δεδομένων) *στην Python*

• Ακέραιοι	`<class 'int'>`	Απλές κλάσεις
• *Κινητής* υποδιαστολής	`<class 'float'>`	
• Μιγαδικοί	`<class 'complex'>`	
• Λογικοί	`<class 'boolean'>`	
• Συμβολοσειρές	`<class 'string'>`	
• Λίστες	`<class 'list'>`	Δομημένες κλάσεις
• Πλειάδες	`<class 'tuple'>`	
• Λεξικά	`<class 'dict'>`	
• Σύνολα	`<class 'set'>`	

Ειδική περίπτωση

• Η κλάση **None**	`<class 'NoneType'>`

Παραδείγματα κλάσεων

foo.py - C:/Users/MK/foo.py (3.7.4)

File Edit Format Run Options Window Help

```
a = 3
print(a, type(a), "\n")

b = 123.4567
print(b, type(b), "\n")

c = 3+5j
print(c, type(c), "\n")

d= True
print(d, type(d), "\n")

e = [1,8, "test", 7.23]
print(e, type(e), "\n")

f = "Hello There!"
print(f, type(f), "\n")

g = (1,8, "test", 7.23)
print(g, type(g), "\n")

h = {'city':'Heraklion','age':7}
print(h, type(h), "\n")

k = {1,2,"aaa"}
print(k, type(k), "\n")

z = None
print(z, type(z), "\n")
```

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
>>>
========== RESTART: C:/Users/MK/foo.py ==========
3 <class 'int'>

123.4567 <class 'float'>

(3+5j) <class 'complex'>

True <class 'bool'>

[1, 8, 'test', 7.23] <class 'list'>

Hello There! <class 'str'>

(1, 8, 'test', 7.23) <class 'tuple'>

{'city': 'Heraklion', 'age': 7} <class 'dict'>

{1, 2, 'aaa'} <class 'set'>

None <class 'NoneType'>

>>>
```

Ln: 639 Col: 4

Ακέραιοι

- Βασικές αριθμητικές πράξεις (+ - * /)

```
>>> 35+4
39
>>> 35/4    (στην Python 2,   το 35/4 επιστρέφει 8)
8.75
```

- Ακέραια διαίρεση (//) και ακέραιο Υπόλοιπο (%)

```
>>> 35//4
8
>>> 35%4
3
```

- Ύψωση σε δύναμη (**)

```
>>> 35**4
1500625
```

Το μέγεθος ενός ακεραίου στην Python είναι (πρακτικά) απεριόριστο

```
>>> a=99999999999999999999999999999999999999999999999   (47 ψηφία)
>>> b=1
>>> a+b
100000000000000000000000000000000000000000000000
```

```
>>> a=999999999999999999999999999999999999999999999999
>>> b=999999999999999999999999999999999999999999999998
>>> a-b
1
```

```
>>> 2**10000  (ο αριθμός 2^10000)
```

```
19950631168807583848837421626835850838234968318861924548520089498529438830221946631919961684036194597899331129423209124271556491349413781117593785932096323
95785573004679379452676524655126605989552055008691819331154250860846061810468550907486608962488809048989483800925394163325785062156830947390255691238806522
50966438744410467598716269854532228685381616943157756296407628368807607322285350916414761839563814589694638994108409605362678210646214273333940365255656495
30603142680234969400335934316651459297773279665775606172582031407994198179607378245683762280037302885487251900834464581454650557929601414833921615734588139
25709537976911927780082695773567444412306201875783632550272832378927071037380286639303142813324140162419567169057406141965434232463880124885614730520743199
22596117962501309928602417083408076059323201612684922884962558413128440615367389514871142563151110897455142033138202029316409575964647560104058458415660720
44962867016515061920631004186422275908670900574606417856951911456055068251250406007519842261898059237118054444788072906395242548339221982707404473162376760
84661303377870603980341319713349365462270056316993745550824178097281098329131440357187752476850985727693792643322159939987688660808368837838027643282775517
22736575727447841122943897338108616074232532919748131201976041782819656974758981645312584341359598627841301281854062834766490886905210475808826158239619857
70122407044330583075869039319604603404973156583208672105913300903752823415539745394397715257455290510212310947321610753474825740775273986348298498340756937
95564663862187456949927901657210370136443313581721431179139822298384584733440270964182851005072927748364550578634501100852987812389473928699540834346158800
70439591189858151457791771436196987281314594837832020814749821718580113890712282509058268174362205774759241765371568772561490458290499246102863008153558330
08130101987675856234343538955409175623400844887526162643568648833519463720377293240094456246923254350400678027273837755376406726898636241037491410966718557
05075909810024678988017827192595338128242195402830275940844895501467666838969799688624163631337639390337345580140763674187771105538422573949911018646821969
65816514851304942223699477147630691554682176828762003627772577237813653316111968112807926694818872012986436607685516398605346022978715575179473852463694469
23087894265948217008051120322365496288169035739121368338393591756418733850510970271613915439590991598154654417336311656936031122249937969999226781732358023
11186264457529913575817500819983923628461524988108896023224436217377161808635701546848405862232979285387562348655644053696262201896357102881236156751254333
83032700290976686505685571575055167275188991941297113376901499161813151715440077286505731895574509203301853048471138183154073240533190384620840364217637039
11550639789000742853672196280903477974533320468368795868580237952218629120080742819551317948157624448298518461509704888027274721574688131594750409732115080
49819045580341682694978714131606321068639151168177430479259670937 6
```

Στις πιο πρόσφατες εκδόσεις της Python έχει τεθεί ο περιορισμός ότι το μήκος ενός ακεραίου δεν πρέπει να υπερβαίνει τα 4.300 ψηφία!!

Μιγαδικοί

```
>>> a=complex(3, 5); print(a)
(3+5j)
>>> a.real
3.0
>>> a.imag
5.0
>>> b= -2+3j; print(b)
(-2+3j)
>>> a+b
(1+8j)
>>> a*b
(-21-1j)
>>> a**b
(-0.0013317022476498123-0.00011641225772990587j)
>>> a*b
(-21-1j)
>>> a**2
(-16+30j)
>>> (0+1j)**2
(-1+0j)
```

Πράξεις ανάμεσα σε μιγαδικούς και ακέραιους ή δεκαδικούς

```
>>> a=(3+5j)
>>> b=5
>>> a+b
(8+5j)

>>> a=(3+5j)
>>> b=4.123
>>> a+b
(7.123+5j)
```

Λογικοί τελεστές

Μπορούν να πάρουν μόνο δυο τιμές *True* ή *False*
Προκύπτουν ως αποτέλεσμα *συγκρίσεων*, λογικών *πράξεων*, ή λογικών *συναρτήσεων*

```
>>> print(10 > 9)
True

>>> print(10 == 9)
False

>>> print(10 < 9)
False
>>> (3 > 5) and (10 < 100)
False

>>> True and True
True
```

```
>>> a= "Hello World"

>>> a.isupper()
False

>>> a.islower()
False

>>> a.startswith("H")
True

>>> a.endswith("d")
True
```

Μεταβλητά και Αμετάβλητα Αντικείμενα στην Python

Αμετάβλητα αντικείμενα *(Immutable objects)*

Από τη στιγμή που δημιουργηθούν, *δεν μπορούν πια να αλλάξουν*

- Τα μέλη όλων των αριθμητικών και λογικών τύπων *(int, float, complex, bool)*
- Συμβολοσειρές *(strings)*
- Πλειάδες *(tuples)*

Μεταβλητά αντικείμενα *(Mutable objects)*

Κατά τη χρήση τους μπορούν να *αλλάζουν περιεχόμενο ή/και μέγεθος*

- Λίστες *(lists)*
- Λεξικά *(dictionaries)*
- Σύνολα *(sets) εκτός από τα frozen sets που είναι μεταβλητά*
- *bytearrays*

- Αριθμητικοί και λογικοί τύποι

- Ακέραιοι	`<class 'int'>`	immutable
- Κινητής υποδιαστολής	`<class 'float'>`	immutable
- Μιγαδικοί	`<class 'complex'>`	immutable
- Λογικοί	`<class 'boolean'>`	immutable
• Συμβολοσειρές	`<class 'string'>`	immutable
• Λίστες	`<class 'list'>`	mutable
• Πλειάδες	`<class 'tuple'>`	immutable
• Λεξικά	`<class 'dict'>`	mutable
• Σύνολα	`<class 'set'>`	mutable
• Bytearrays	`<class 'bytearray'>`	mutable

Ειδική περίπτωση

• Ο τύπος **`None`**	`<class 'NoneType'>`	immutable

Η Python χειρίζεται διαφορετικά τα μεταβλητά απ' ότι τα αμετάβλητα αντικείμενα

- Επιλέγουμε να δημιουργήσουμε ένα *αμετάβλητο* αντικείμενο, όταν είμαστε σίγουροι ότι το περιεχόμενό του θα *παραμένει πάντα ίδιο*

```
decimal_digits = (0,1,2,3,4,5,6,7,8,9)
```
με παρενθέσεις () ορίζεται μια πλειάδα

- Τα *μεταβλητά* αντικείμενα είναι εξαιρετικά *χρήσιμα* όταν είναι *απαραίτητο* να *αλλάζει το μέγεθος* τους, πχ λίστες, λεξικά κλπ.

```
prices = [12, 7.2, 35, 4]
```
με [ ] ορίζεται μια λίστα

- Τα *αμετάβλητα* αντικείμενα είναι *ταχύτερα προσβάσιμα* από τα μεταβλητά όμως αν χρειαστεί να τροποποιηθούν είναι *"ακριβά για να αλλάξουν"*, διότι κάτι τέτοιο συνεπάγεται τη *δημιουργία ενός αντιγράφου*

- Αντίθετα, η πρόσβαση στα *μεταβλητά* αντικείμενα είναι πιο αργή, κάτι που αντισταθμίζεται όμως από το ότι μπορούν να τροποποιούνται ελεύθερα

Τέλος Διάλεξης

Ερωτήσεις;

Τμήματα αυτής της διάλεξης περιέχουν πληροφορίες από πηγές που είναι ελεύθερες στο διαδίκτυο όπως η Βικιπαίδεια και ανοιχτές σημειώσεις παρεμφερών διαλέξεων.

Συνεχίζουμε με τις Ασκήσεις Πράξης

Εκφωνήσεις (φυλλάδια) στο *eclass > Έγγραφα > Ασκήσεις Πράξεις 2025*

Για κάθε άσκηση στο φυλλάδιο, *εκτός από την τελευταία*

- Σκεφτόμαστε όλοι μαζί πως θα τη λύσουμε
- Πληκτρολογώ και τρέχω τον κώδικα στην οθόνη

  Όσοι κρατάτε laptop μπορείτε να το κάνετε ταυτόχρονα μαζί μου
- Στο σπίτι δοκιμάζετε να την τρέξετε και μόνοι σας
- Λίγο μετά το μάθημα, οι λύσεις ανεβαίνουν στο eclass

Η τελευταία άσκηση *(συνήθως πιο απαιτητική από τις προηγούμενες)*

- Αποτελεί την *εργασία της εβδομάδας* *(κάθε εβδομάδα)*
- Την λύνετε μόνοι σας σπίτι και την ανεβάζετε στο eclass