Bioinformatics Data Skills

Working with **Python**

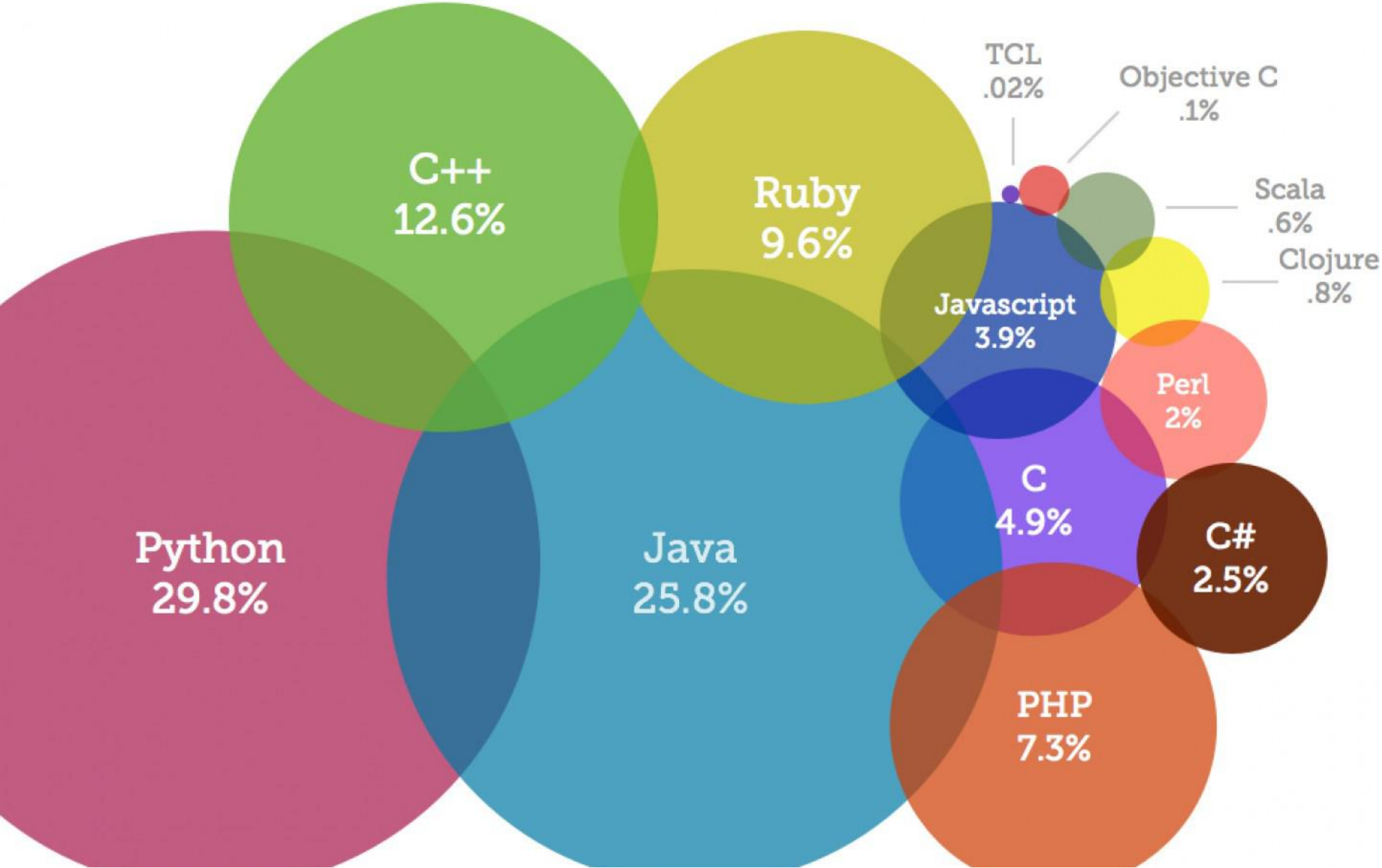
Tamposis Ioannis

Researcher & Software Development Engineer

B.Sc. in Informatics, M.Sc. in Biomedical, Ph.D. in Bioinformatics

Department of Computer Science and Biomedical Informatics, University of Thessaly

Most Popular Coding Languages of 2018



Υπάρχουν δύο τρόποι για να κατασκευάσετε το σχεδιασμό ενός λογισμικού: ο ένας είναι να το κάνετε τόσο απλό ώστε προφανώς να μην υπάρχουν ελαττώματα, και ο άλλος είναι να το κάνετε τόσο πολύπλοκο ώστε να μην είναι προφανή τα ελαττώματα.

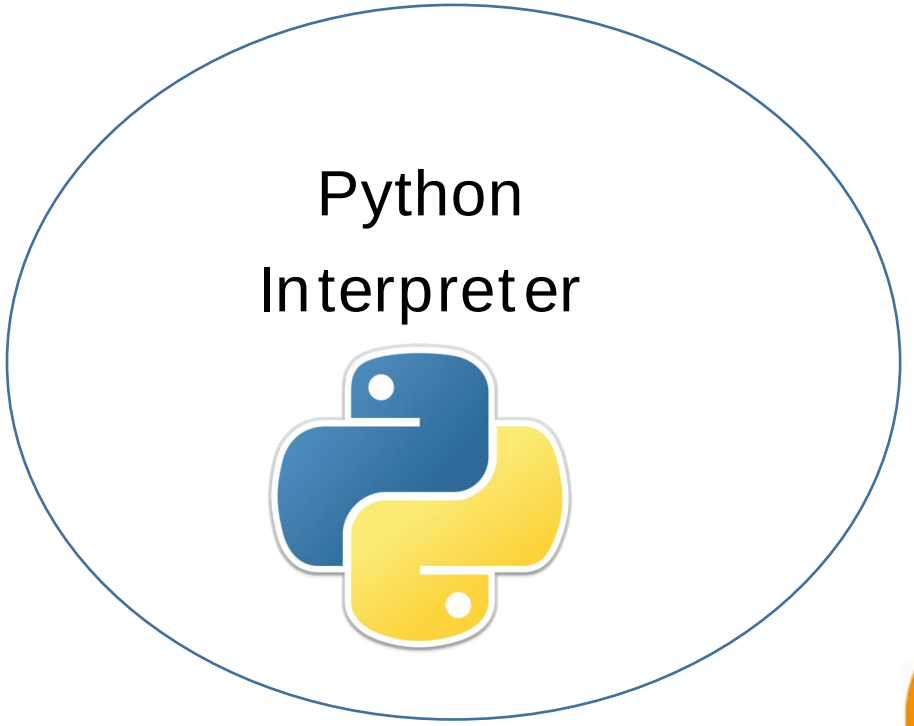
C. A. R. Hoare

Γιατί Python;

Βασική αρχή της Python είναι ότι ο χρόνος του προγραμματιστή είναι πολύ σημαντικότερος από τους κύκλους μηχανής και τον απαλλάσσει από τις χαμηλού επιπέδου λεπτομέρειες που πρέπει να φροντίσει στη C/C++ ή, ως ένα βαθμό, στη Java.

- Εύκολη στην εκμάθηση
- Συντακτική Απλότητα
- Υποχρεωτική Στοίχιση
- Δομές Δεδομένων
- Γλώσσα υψηλού επιπέδου
- Διερμηνευόμενη
- Επεκτάσιμη
- Αντικειμενοστρεφής
- Συνδυασμός Προγραμματιστικών Παραδειγμάτων
- Εκτεταμένες βιβλιοθήκες (modules)

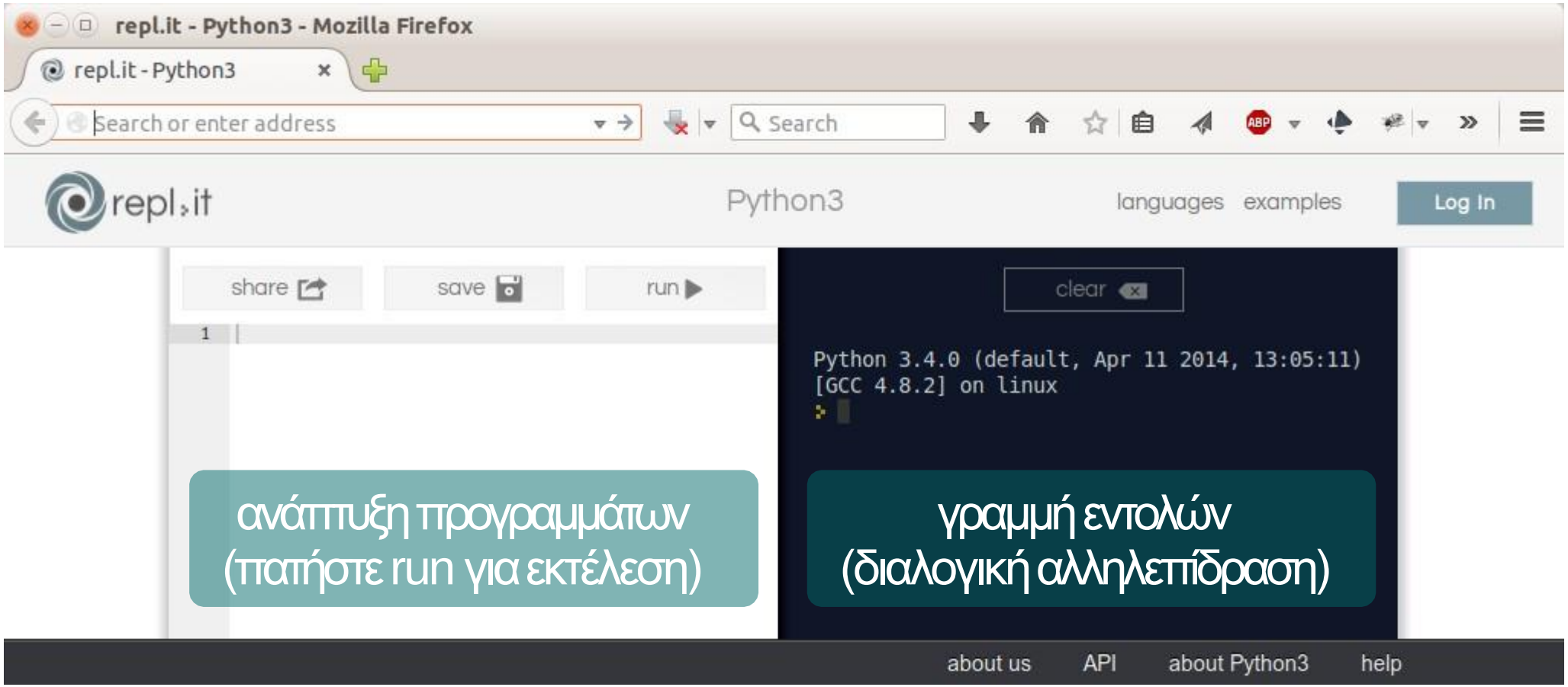
Python Interpreter + IDE



PyScripter

Online Python IDE

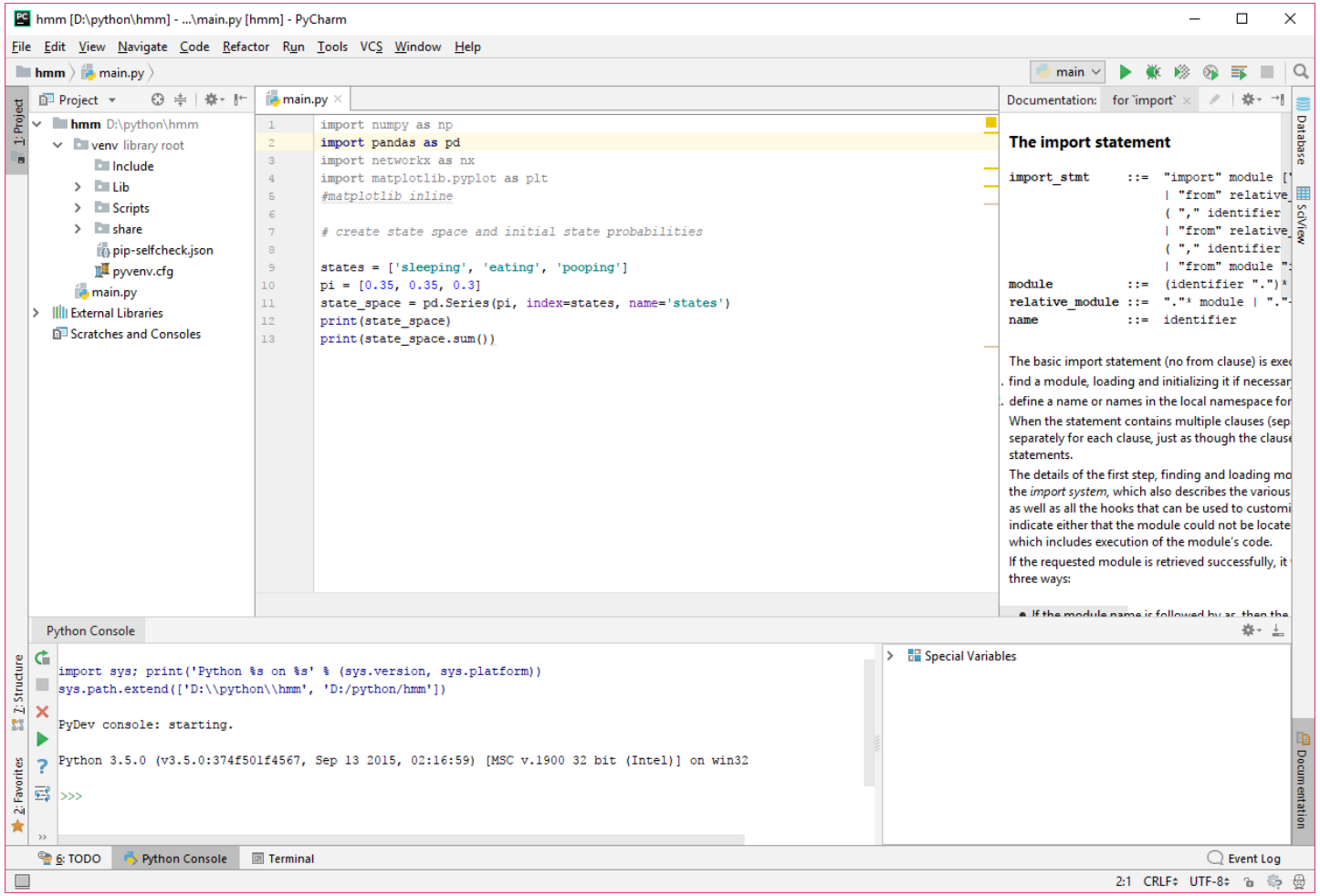
http://repl.it/languages/Python3

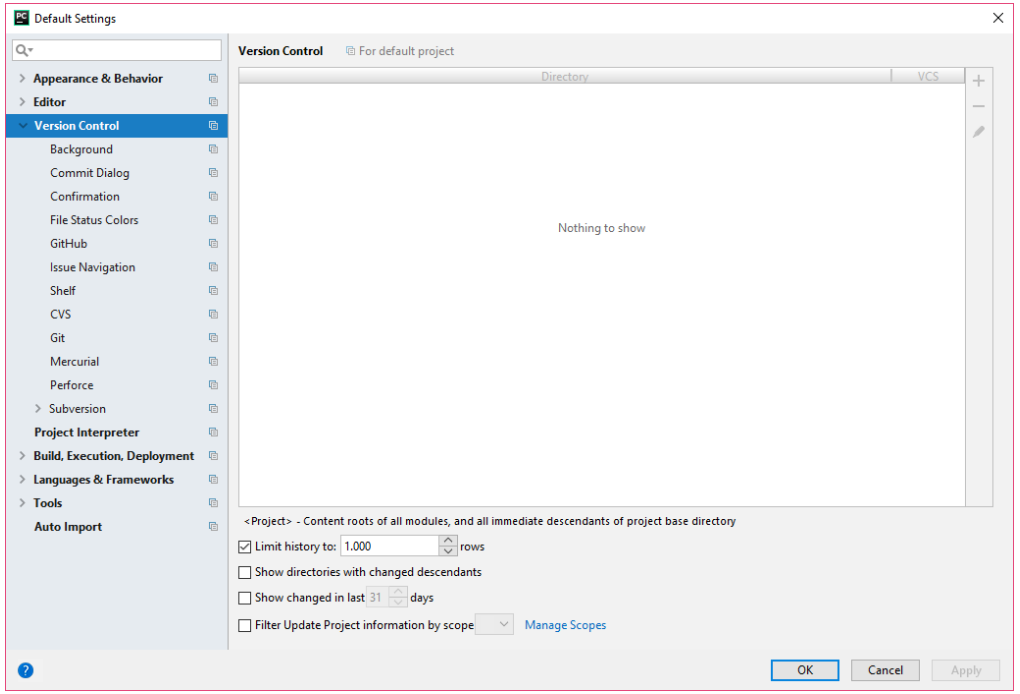


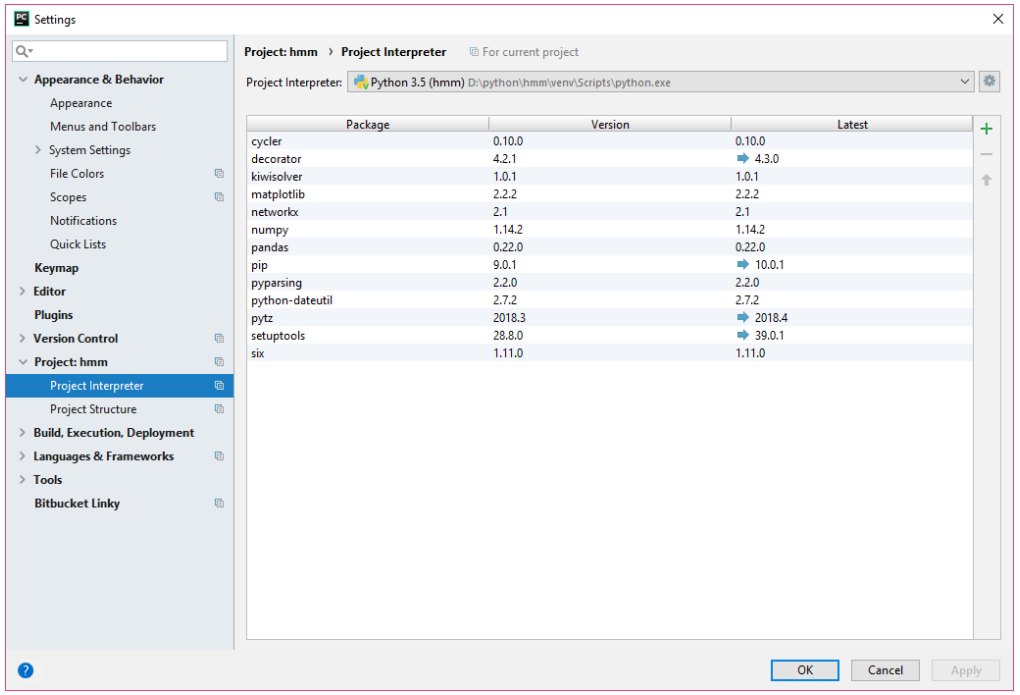
Γιατί IDE;

- Quickly navigating
- Compiling and managing lib dependencies
- Debugging
- Refactoring and performance hints
- Automatic code generation
- Organise imports (automatically adding appropriate imports)
- Warning-as-you-type (i.e. some errors don't even require a compile cycle)
- Keeping a view of files, errors/warnings/console/unit tests etc and source code all on the screen at the same time in a useful way
- Integrated source control

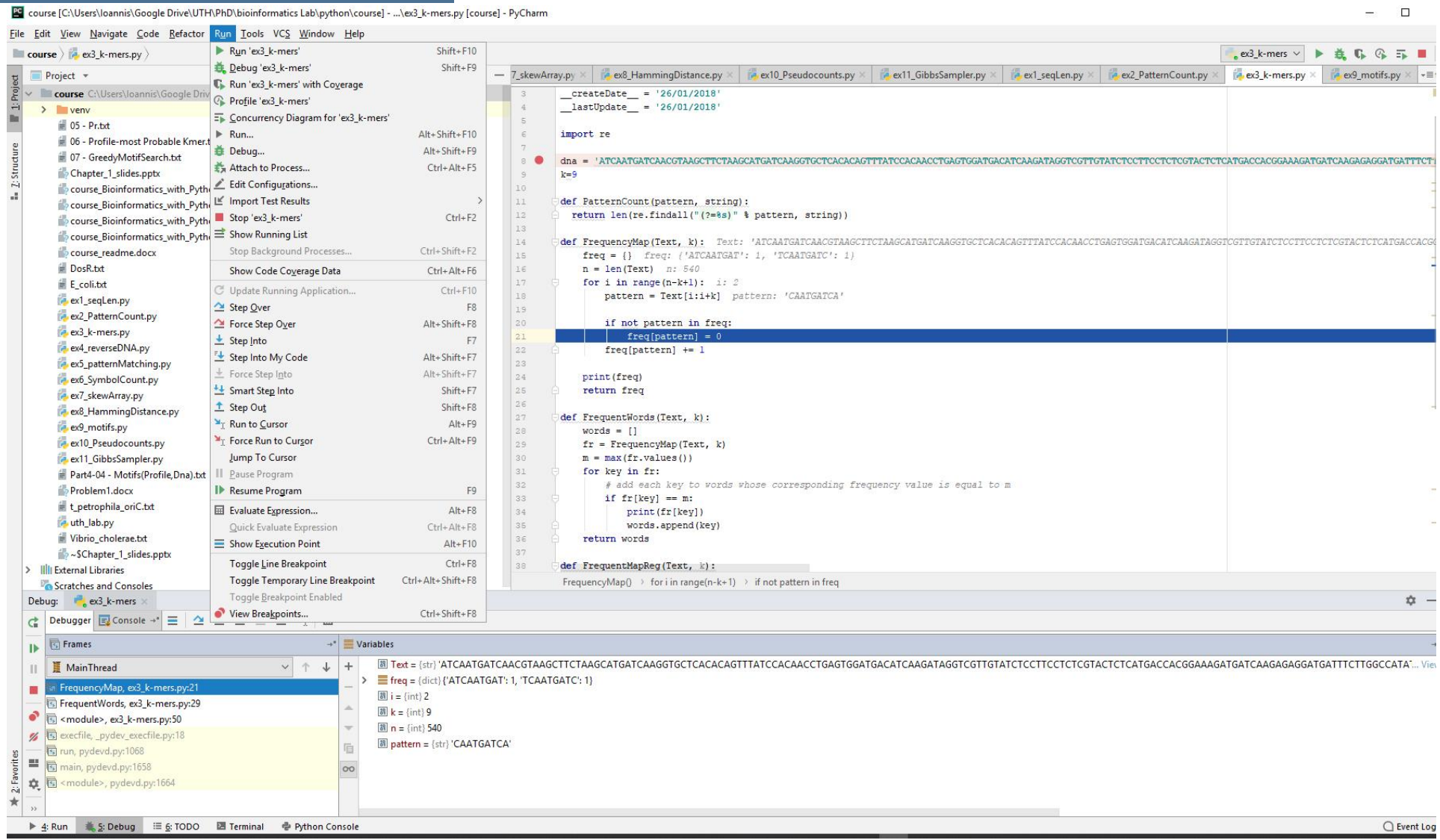
Γιατί IDE;


hmm [D:\python\hmm] - ...\main.py [hmm] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
hmm main.py
Project
hmm D:\python\hmm
venv library root
Include
Lib
Scripts
share
pip-selfcheck.json
pyvenv.cfg
main.py
External Libraries
Scratches and Consoles
main.py
1 import numpy as np
2 import pandas as pd
3 import networkx as nx
4 import matplotlib.pyplot as plt
5 #matplotlib inline
6
7 # create state space and initial state probabilities
8
9 states = ['sleeping', 'eating', 'pooping']
10 pi = [0.35, 0.35, 0.3]
11 state_space = pd.Series(pi, index=states, name='states')
12 print(state_space)
13 print(state_space.sum())
Documentation: for `import`
The import statement
import_stmt ::= "import" module
| "from" relative_
("," identifier
| "from" relative_
("," identifier
| "from" module
module ::= (identifier ".")*
relative_module ::= "."* module | "."
name ::= identifier
The basic import statement (no from clause) is exe
find a module, loading and initializing it if necessar
define a name or names in the local namespace for
When the statement contains multiple clauses (sep
separately for each clause, just as though the clause
statements.
The details of the first step, finding and loading mo
the import system, which also describes the various
as well as all the hooks that can be used to customi
indicate either that the module could not be locate
which includes execution of the module's code.
If the requested module is retrieved successfully, it
three ways:
Python Console
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\python\\hmm', 'D:/python/hmm'])
PyDev console: starting.
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
>>>
Special Variables
6: TODO
Python Console
Terminal
Event Log
2:1 CRLF UTF-8


Default Settings
Appearance & Behavior
Editor
Version Control
Background
Commit Dialog
Confirmation
File Status Colors
GitHub
Issue Navigation
Shelf
CVS
Git
Mercurial
Perforce
Subversion
Project Interpreter
Build, Execution, Deployment
Languages & Frameworks
Tools
Auto Import
Version Control
For default project
Directory
VCS
Nothing to show
<Project> - Content roots of all modules, and all immediate descendants of project base directory
Limit history to: 1.000 rows
Show directories with changed descendants
Show changed in last 31 days
Filter Update Project information by scope
Manage Scopes
OK
Cancel
Apply


Settings
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Keymap
Editor
Plugins
Version Control
Project: hmm
Project Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Bitbucket Linky
Project: hmm
Project Interpreter
For current project
Project Interpreter: Python 3.5 (hmm) D:\python\hmm\venv\Scripts\python.exe
Package Version Latest
cycler 0.10.0 0.10.0
decorator 4.2.1 4.3.0
kiwisolver 1.0.1 1.0.1
matplotlib 2.2.2 2.2.2
networkx 2.1 2.1
numpy 1.14.2 1.14.2
pandas 0.22.0 0.22.0
pip 9.0.1 10.0.1
pyparsing 2.2.0 2.2.0
python-dateutil 2.7.2 2.7.2
pytz 2018.3 2018.4
setuptools 28.8.0 39.0.1
six 1.11.0 1.11.0
OK
Cancel
Apply

Γιατί IDE;


course [C:\Users\Ioannis\Google Drive\UTH\PhD\bioinformatics Lab\python\course] - ...\ex3_k-mers.py [course] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Run 'ex3_k-mers' Shift+F10
Debug 'ex3_k-mers' Shift+F9
Run 'ex3_k-mers' with Coverage
Profile 'ex3_k-mers'
Concurrency Diagram for 'ex3_k-mers'
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Process... Ctrl+Alt+F5
Edit Configurations...
Import Test Results
Stop 'ex3_k-mers' Ctrl+F2
Show Running List
Stop Background Processes... Ctrl+Shift+F2
Show Code Coverage Data Ctrl+Alt+F6
Update Running Application... Ctrl+F10
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Step Into My Code Alt+Shift+F7
Force Step Into Alt+Shift+F7
Smart Step Into Shift+F7
Step Out Shift+F8
Run to Cursor Alt+F9
Force Run to Cursor Ctrl+Alt+F9
Jump To Cursor
Pause Program
Resume Program F9
Evaluate Expression... Alt+F8
Quick Evaluate Expression Ctrl+Alt+F8
Show Execution Point Alt+F10
Toggle Line Breakpoint Ctrl+F8
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
__createDate__ = '26/01/2018'
__lastUpdate__ = '26/01/2018'
import re
k=9
def PatternCount(pattern, string):
return len(re.findall("(?=%s)" % pattern, string))
def FrequencyMap(Text, k):
freq = {}
n = len(Text)
for i in range(n-k+1):
pattern = Text[i:i+k]
if not pattern in freq:
freq[pattern] = 0
freq[pattern] += 1
print(freq)
return freq
def FrequentWords(Text, k):
words = []
fr = FrequencyMap(Text, k)
m = max(fr.values())
for key in fr:
# add each key to words whose corresponding frequency value is equal to m
if fr[key] == m:
print(fr[key])
words.append(key)
return words
def FrequentMapReg(Text, k):
Debug: ex3_k-mers
Debugger Console
Frames
Variables
MainThread
FrequencyMap, ex3_k-mers.py:21
FrequentWords, ex3_k-mers.py:29
<module>, ex3_k-mers.py:50
execfile, _pydev_execfile.py:18
run, pydevd.py:1068
main, pydevd.py:1658
<module>, pydevd.py:1664
Text = {str} 'ATCAATGATCAACGTAAGCTTCTAAGCATGATCAAGGTGCTCACACAGTTTATCCACAACCTGAGTGGATGACATCAAGATAGGTCGTTGTATCTCCTTCCTCTCGTACTCTCATGACCACGGAAAGATGATCAAGAGAGGATGATTTCTTGGCCATA
freq = {dict} {'ATCAATGAT': 1, 'TCAATGATC': 1}
i = {int} 2
k = {int} 9
n = {int} 540
pattern = {str} 'CAATGATCA'
4: Run
5: Debug
6: TODO
Terminal
Python Console
Event Log

Έκδοση 2 → 3

- 7 / 4 = 1.75 (και ο ειδικός τελεστής // πλέον αφορά αριθμητική ακεραίων)
- Εκτεταμένη χρήση Γεννητόρων για καλύτερη χρήση μνήμης.
- Η print έγινε συνάρτηση
  - Έκδοση 2 print “Hello world”
  - Έκδοση 3 print(“Hello world”)
- Οι βιβλιοθήκες αναδιοργανώθηκαν
- Εύκολη υποστήριξη Unicode
- Και διάφορες άλλες αλλαγές

https://wiki.python.org/moin/Python2orPython3

Hello World

Υπάρχουν δύο διαφορετικές εκδόσεις Python. Τόσο η Python 2 όσο και η Python 3 χρησιμοποιούνται σε ολόκληρο τον πλανήτη. Η πιο σημαντική διαφορά μεταξύ των δύο είναι ο τρόπος με τον οποίο γράφεται μια δήλωση εκτύπωσης.

Έκδοση 2

```
print 'Hello world'
```

Έκδοση 3

```
print('Hello world')
```

Handling Errors

```
print "How do you make a hot dog stand?'
print You take away its chair!
```

Αν ο κώδικας σας έχεις ένα σφάλμα η Python θα το εντοπίσει, το πρόγραμμα θα διακοπεί απότομα και θα εμφανιστεί ένα μήνυμα σφάλματος που θα σας ενημερώνει κατάλληλα

```
File "script.py", line 1
   print("How do you make a hot dog stand?')

SyntaxError: EOL while scanning string literal
```

Μεταβλητές

int
float
bool
str

```
int1 = 1
int2 = 10
int3 = -5
pi = 3.14
float1 = 10.
float2 = -5.5
# this evaluates to 150:
float4 = 1.5e2
DNA= 'CCATTGA'
a = True
b = False
```

Πολλαπλή εκχώρηση

```
a = b = c = 0
name, age = 'Άλαν', 13
```

Βασικοί Τελεστές

- Πρόσθεση (+)
- Αφαίρεση (-)
- Πολλαπλασιασμός (*)
- Διαίρεση (/)
- Διαίρεση στρογγυλοποιημένη (//)
- Ύψωση σε δύναμη (**)
- Υπόλοιπο(modulo) (%)

Προσοχή στην προτεραιότητα

- Ίσο με (==)
- Διάφορο από (! =)
- Μεγαλύτερο από (>)
- Μικρότερο από (<)
- Μεγαλύτερο ή ίσο με (>=)
- Μικρότερο ή ίσο με (<=)

Αριθμητικοί

\+ - * / ** // %

```
>>> 120 // 13
9
>>> 13 ** 3
2197
>>> 15 % 2
1
>>> 100 / 6
16
>>> float(100) / 6
16.666666667
```

Λογικοί

and or not

```
>>> a = 5
>>> b = 4
>>> a > b
True
>>> c = 6
>>> a > b and a > c
False
```

Συγκριτικοί

\> >= < <= == !=

Ποιόν τελεστή θα χρησιμοποιήσουμε για να ελέγξουμε αν το 1398 διαιρείται ακριβώς από το 11?

Ενημέρωση μεταβλητών

```
money_in_wallet = 40
sandwich_price = 7.50
sales_tax = .08 * sandwich_price

sandwich_price += sales_tax
money_in_wallet -= sandwich_price
```

Αθροιστής

```
sum += x    ⇔    sum = sum + x
```

Μετρητής

```
count += 1 ⇔ count = count + 1
```

Σχόλια

```
# σχόλιο μιας γραμμής
```

```
'''
σχόλιο
πολλαπλών
γραμμών
'''
```

Μετατροπή Τύπου (casting)

```
number1 = "100"
number2 = "10"

string_addition = number1 + number2
#string_addition now has a value of "10010"

int_addition = int(number1) + int(number2)
#int_addition has a value of 110
```

```
string_num = "7.5"
print int(string_num)
>>> 7
print float(string_num)
>>> 7.5
```

```
product = 0.25 * 40.0
big_string = "The product was "+str(product)
```

Δομή Επιλογής

η **στοίχιση** ΔΕΝ είναι προαιρετική

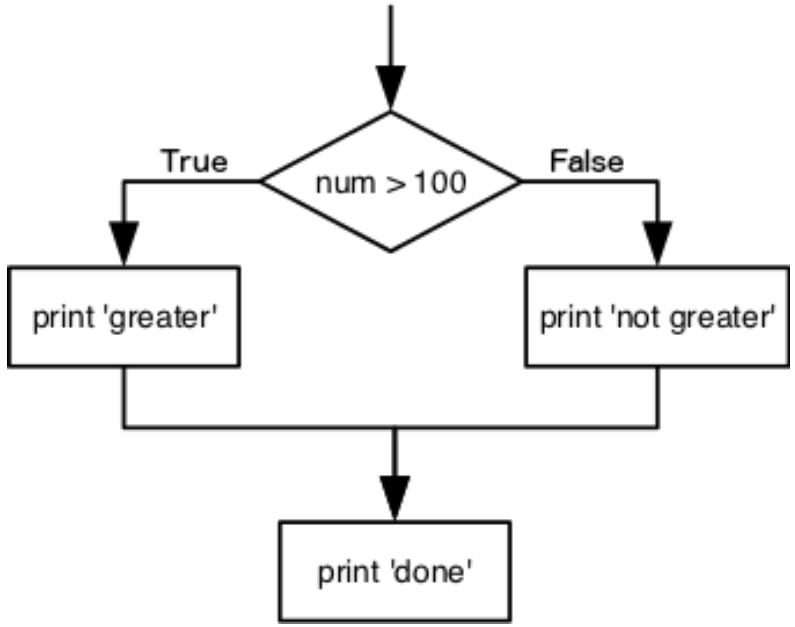
εσοχή = εμφώλευση

μην παραλείπετε το σύμβολο :

```
if συνθήκη:
    εντολές
```

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

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



Δομή Επιλογής

παραδείγματα

```
if x > max:
    max = x
```

```
if x > y:
    max = x
else:
    max = y
print max
```

```
 if x == "A":
    dnaA += 1
 elif x == "C":
    dnaC += 1
 elif x == "G":
    dnaG += 1
 elif x == "T" :
    dnaT += 1
 else:
    print "Το σύμβολο δεν υπάρχει στο DNA"
```

Δομή Επανάληψης

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

for

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

συχνά η for χρησιμοποιείται σε συνδυασμό με την ακολουθία range

range(a,b,step)

το b ΔΕΝ συμπεριλαμβάνεται στην ακολουθία

```
for i in range(10):
    print i, end=" "
```

0 1 2 3 4 5 6 7 8 9

```
for i in range(5,50,10):
    print i, end=" "
```

5 15 25 35 45

```
for c in "ACCTTCGT"
    print c, end="-"
```

A-C-C-T-T-C-G-T-

Δομή Επανάληψης

while

Οι βρόγχοι while θα εκτελούνται συνεχώς όσο πληρείται μια συγκεκριμένη συνθήκη.

```
while συνθήκη:
    εντολή(ές)
```

```
num = 10
while num != 0:
    sum += num
print sum
```

```
count = 0
while count < 5:
   print count, " is  less than 5"
   count = count + 1
else:
   print count, " is not less than 5"
```

```
flag = 1
while (flag): print 'Given flag is really true!'
print "Good bye!"
```

Δομή Επανάληψης

Loop Control Statements

Οι εντολές ελέγχου βρόχου αλλάζουν την εκτέλεση από την κανονική ακολουθία.

Η **break** χρησιμοποιείται για τη διακοπή μιας εντολής βρόχου.

Η **continue** χρησιμοποιείται για να παραλείψει τις επόμενες εντολές της τρέχουσας επιλογής.

```
for letter in 'Python':
   if letter == 'h':
      break
   print 'Letter :', letter
var = 10
while var > 0:
   print 'Number :', var
   var = var -1
   if var == 5:
      break
print "Good bye!"
```

```
Letter : P
Letter : y
Letter : t
Number : 10
Number  : 9
Number  : 8
Number  : 7
Number  : 6
Good bye!
```

```
for letter in 'Python':
   if letter == 'h':
      continue
   print 'Letter :', letter
var = 10
while var > 0:
   print 'Number :', var
   var = var -1
   if var == 5:
      continue
print "Good bye!"
```

```
Letter : P
Letter : y
Letter : t
Letter : o
Letter : n
Number : 9
Number : 8
Number : 7
Number : 6
Number : 4
Number : 3
Number : 2
Number : 1
Number : 0
Good bye!
```

Συναρτήσεις (Functions)

Μια συνάρτηση πρέπει να έχει οριστεί πριν χρησιμοποιηθεί.

Μία συνάρτηση μπορεί να επιδέχεται πολλά ορίσματα.

Τα αποτελέσματα επιστρέφονται με την **return**

```
def functionname( parameters ):
   function_body
   return [expression]
```

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

```
# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str
   return

# Now you can call printme function
printme()
```

Δομές Δεδομένων

Συμβολοσειρές (Strings)
Λίστες (Lists)
Πλειάδες (Tuples)
Λεξικά (Dicts)

Με τον όρο δομή δεδομένων εννοούμε τον τρόπο αποθήκευσης δεδομένων, οργανωμένα και βασιζόμενα σε πιο απλούς τύπους δεδομένων

Συμβολοσειρές (strings)

```
>>> word = "PROTEIN"
```

0	1	2	3	4	5	6
P	R	O	T	E	I	N
-7	-6	-5	-4	-3	-2	-1

Immutable

[a:b:step]

```
>>> var1 = 'Hello World!'
>>> var2 = "Python Programming"
>>> print var1[0]
H
>>> print var2[1:5]
ytho
>>> print var2[:5]
Pytho
>>> print var1[::4]
Hor
```

Συμβολοσειρές (strings)

```
>>> str = "Molecular" + "Biology"
>>> print(str)
Molecular Biology
```

Τελεστές + * in
Συνάρτηση len()

```
>>> "A" * 10
AAAAAAAAAA
```

```
>>> "Biology" in str
True
>>> "DNA" in str
False
```

```
>>> len(str)
14
```

Συμβολοσειρές (strings)

Διάσχιση

```
temp = 'Python'
for letter in temp:     # First Example
   print 'Current Letter :', letter
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
```

```
for i in range(len(temp )):     # Second Example
   print 'Current Letter :', letter[i]
```

Αντιστροφή (reverse)

```
>>> seq = 'GUUAGCUAUG'
>>> print seq
>>> print seq[::-1]
GUUAGCUAUGA
AGUAUCGAUUG
```

Συμβολοσειρές (strings)

Μέθοδοι

```
str = 'ACCTTCTGGGAAA'
count(str, beg= 0,end=len(string))
str.count('A')

replace(old, new [, max])
str.replace('T', 'U')

split(str="", num=string.count(str))
str.split()

find(str, beg=0 end=len(string))
str.find('ACC')

str.isupper()

len(string)
len(str)
```

Λίστες (Lists)

```
>>> k = ['ATA','ATC','CAT', 'CCA']
```

0	1	2	3
ATA	ATC	CAT	CCA
-4	-3	-2	-1

mutable

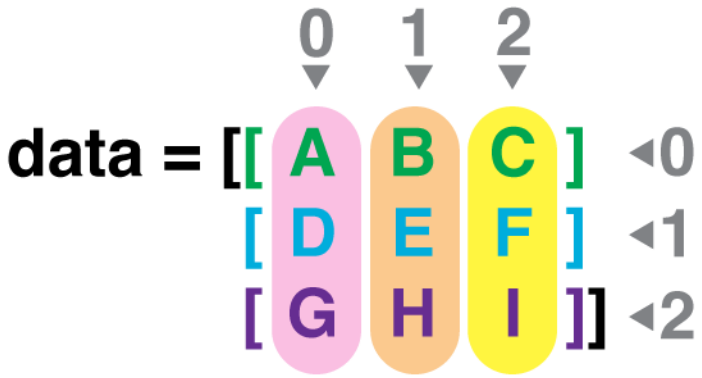
Οι λίστες είναι η σημαντικότερη από τις σύνθετες δομές δεδομένων της Python

```
>>> print k[0]                    (Πρώτο στοιχείο)
ATA
>>> print var2[1:2]
['ATC','CAT']
>>> print k[-1]                   (Τελευταίο στοιχείο)
CCA
```

```
list1 = ['physics', 'chemistry', 1997, 2000]
len([1, 2, 3]) = 3                                  (Μέγεθος)
['Hi!'] * 4  = ['Hi!', 'Hi!', 'Hi!', 'Hi!']         (Επανάληψη)
[1, 2, 3] + [4, 5, 6] = [1, 2, 3, 4, 5, 6]          (Συνδυασμός)
```

Λίστες (Lists)

```
data = [ [ A B C ],
         [ D E F ],
         [ G H I ] ]
```



data[0][0] = A
data[0][1] = B
data[0][2] = C

data[1][0] = D
data[1][1] = E
data[1][2] = F

data[2][0] = G
data[2][1] = H
data[2][2] = I

Λίστες (Lists)

```
>>> k = ['ATA','ATC','CAT', 'CCA']
>>> k[2] = 'ACC'
>>> print k
['ATA','ATC', 'ACC', 'CCA']
```

Ενημέρωση

list[]

Προσθήκη

list.append()

list.insert(index, obj)

```
>>> k.append('GTC')
>>> k.insert(1,'GGG')
>>> print k
['ATA', 'GGG', 'ATC', 'ACC', 'CCA', 'GTC']
```

Διαγραφή

del list[]

list.remove(obj)

list.pop(index)

```
>>> del k[2]
>>> k.remove('GGG')
>>> print k
['ATA','ATC', 'CCA', 'GTC']
```

Λίστες (Lists)

```
>>> k = ['CAT','ATC','CCA','ATA']
>>> print k.sort()
['ATA', 'ATC', 'CAT', 'CCA']
>>> int = [4,8,3,1]
>>> print k.sort()
[1, 3, 4, 8]
```

Μέθοδοι

list.sort()
list.count()
list.reverse()

```
>>> k = ['CAT','ATC','CCA','ATA','ATC']
>>> print k.count('ATC')
2
```

```
>>> k = ['CAT','ATC','CCA','ATA']
>>> print k.reverse()
['ATA', 'CCA', 'ATC', 'CAT']
```

Λίστες (Lists)

```
>>> dna = 'ATTTCCG'
>>> list(dna)
['A', 'T', 'T', 'T', 'C', 'C', 'G']
```

μετατροπή συμβολοσειράς σε λίστα και αντιστρόφως

```
>>> k = ['CAT','ATC','CCA','ATA']
>>> " ".join(k)
CATATCCCAATA
```

Εμφωλευμένες λίστες

```
>>> list1 = ["Maria", "Lena"]
>>> list2 = [list1, "Dimitra", "Tonia", "Python"]
>>> print mylist
["Maria", "Lena", "Dimitra", "Tonia", "Python"]
```

Καθαρισμός της λίστας

```
>>> mylist[:] = []
```

Πλειάδες (Tuples)

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

immutable

Μια πλειάδα είναι μια ακολουθία αμετάβλητων αντικειμένων της Python. Τα Tuples είναι ακολουθίες, ακριβώς όπως λίστες. Οι διαφορές μεταξύ πλειάδων και λιστών είναι, οι πλειάδες δεν μπορούν να αλλάξουν σε αντίθεση με τους καταλόγους και οι πλειάδες χρησιμοποιούν παρενθέσεις, ενώ οι λίστες χρησιμοποιούν αγκύλες.

```
>>> print tup1[0]
physics
>>> print tup2[1:3]
(2, 3, 4)
```

```
>>> del tup;
>>> print tup;
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
NameError: name 'tup' is not defined
```

Λεξικά (Dictionary)

πίνακες κατατερματισμού (hashtables)

πίνακες συσχετισμού (associative arrays)

mutable

ζεύγη κλειδιών: τιμών

```
dict = { key1:val1, key2:val2, ...}
```

```
>>> dna = {'A': 7, 'C': 4, 'G': 2 , 'T': 9}
>>> print dna['A']
7
>>> dna['A'] = 10          (Ενημέρωση)
>>> dna['U'] = 0           (Εισαγωγή)
>>> del dna['T']           (Διαγραφή)
>>> dna.clear();           (Διαγραφή όλων)
>>> del dna                (Διαγραφή Λεξικού)
```

```
>>> print dna.has_key('A')
True
>>> print dna.values()
[7,4,2,9]
>>> print dna.keys()
['A', 'C', 'G','T']
```

Λεξικά (Dictionary)

```
dna = "AATGATGAACGAC"
dinucleotides = ['AA','AT','AG','AC',
                 'TA','TT','TG','TC',
                 'GA','GT','GG','GC',
                 'CA','CT','CG','CT']
all_counts = {}
for dinucleotide in dinucleotides:
    count = dna.count(dinucleotide)
    if count > 0:
        all_counts[dinucleotide] = count
print(all_counts)
```

```
{'AA': 2, 'AC': 2, 'CG': 1, 'AT': 2, 'GA': 3, 'TG': 2}
```

```
print(all_counts['TA'])
```

Αρχεία (κειμένου)

Άνοιγμα

```
with open("test.txt", "r", \encoding="utf-8") as myfile
```

Ανάγνωση

```
for line in myfile:
  lines.append(line)

lines = myfile.readlines()
```

Εγγραφή

```
myfile.write(“DNA")
```

Κλείσιμο

```
myfile.close()
```

Αρχεία (csv)

Άνοιγμα

```
import csv
with open('test.csv', 'r') as csvfile:
  reader = csv.reader(csvfile, delimiter=' ')
```

Ανάγνωση

```
for line in reader :
  lines.append(line)

lines = myfile.readlines()
```

Εγγραφή

```
fieldnames = ['f_name', 'l_name']
writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

writer.writeheader()
writer.writerow({'f_name': 'John', 'l_name': ‘Tamposis'})
```

Κλείσιμο

```
myfile.close()
```

Αρχεία (Fasta)

Άνοιγμα

```
import re          (Regular expression module)

with open('test.fa', 'r') as file:
```

Ανάγνωση

```
for line in file:
    if line.startswith('>'):
        header = line
        #print header
    else:
        #print sequence
        seq = line
        motif="ML[A-Z][A-Z][IV]R"
        if re.findall(motif,seq):
            print seq
```

Κλείσιμο

```
myfile.close()
```

Βιβλιοθήκες (modules)

Δημιουργία

```
def greeting(name):
  print("Hello, " + name)

import mymodule

mymodule.greeting("Jonathan")
```

Built-in Modules

Μπορείτε να ενσωματώνεται δικά σας modules , άλλων κατασκευαστών ή ενσωματωμένα

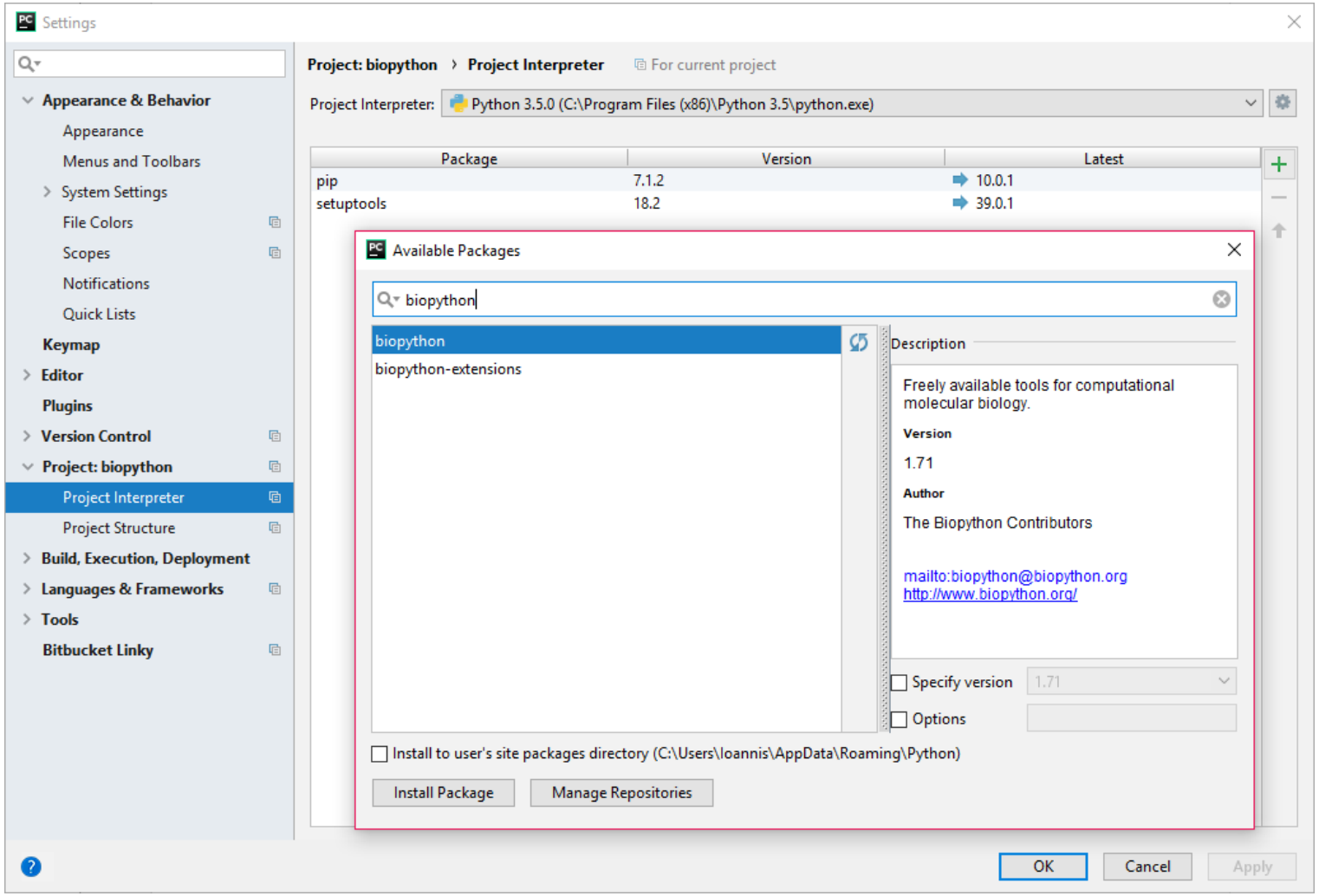
```
import random
log = random.randint(1,6)

import time
time.sleep(60)

import math
content = math.sqrt(4)
```

Third-part modules

Biopython
Networkx (Biology Network)
Numpy (Lists)
matplotlib (Plots)

Βιβλιοθήκες (modules)

Εγκατάσταση



Αρχεία (Διαχείριση)

Μετονομασία

```
import os          (module)
os.rename("old.txt", "new.txt")
os.rename("/home/biology/old.txt",
"/home/biology/new.txt")
```

Δημιουργία φακέλου/ων

```
os.mkdir("/home/martin/python")
os.mkdirs("/a/long/path/with/lots/of/folders")
```

Διαγραφή αρχείου/φακέλου

```
os.remove("/home/martin/unwanted_file.txt")
os.rmdir("/home/martin/empty")
```

Λίστα περιεχομένων φακέλου

```
for file_name in os.listdir("."):
    print("one file name is " + file_name)

for file_name in os.listdir("/home/bio"):
    print("one file name is " + file_name)
```

Εκτέλεση εξωτερικών προγραμμάτων

```
import subprocess          (module)
subprocess.call("/bin/date")
```

Εκτέλεση shell εντολής

```
out = subprocess.Popen(['wc', '-l', seq_file.txt'],
        stdout=subprocess.PIPE,
        stderr=subprocess.STDOUT)
```

Εδώ η τυποποιημένη έξοδος περιέχει το αποτέλεσμα από την εντολή wc -l και το stderr περιέχει **None**, καθώς δεν υπάρχουν σφάλματα.

```
>stdout,stderr = out.communicate()
>print(stdout)
   3503 my_text_file.txt
>print(stderr)
None
```

Μπορούμε στη συνέχεια να αναλύσουμε το stdout για να πάρουμε το αποτέλεσμα από την εντολή shell στο Python

```
>stdout.split()[0]
'3503'
```

Math (module)

Η βιβλιοθήκη math παρέχει τις συνηθισμένες συναρτήσεις άλγεβρας και τριγωνομετρίας μαζί με διάφορες σταθερές μαθηματικών.

Σημείωση: οι τριγωνομετρικές λειτουργίες λειτουργούν σε ακτίνια και όχι σε μοίρες.

Μπορείτε να μετατρέψετε από ακτίνια σε βαθμούς με math.degrees και αντίστροφα με math.radians.

```
import math          (module)
>> math.e
2.718281828459045
>> math.pi
3.141592653589793
>> math.log10(100)
2.0
>> math.log(math.e)
1.0
>> math.cos(math.pi)
-1.0
>> math.exp(1)
2.7182818284590455
>> math.pow(5,2)
25.0
>> math.sin(math.pi)
1.2246467991473532e-16
```

numpy (module)

Το NumPy είναι το θεμελιώδες πακέτο για την επιστημονική πληροφορική με την Python. Περιέχει, μεταξύ άλλων

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

```
import numpy as np

>>x = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>x.shape
(3,3)
>>x.ndim
2
>>x.size
9
```

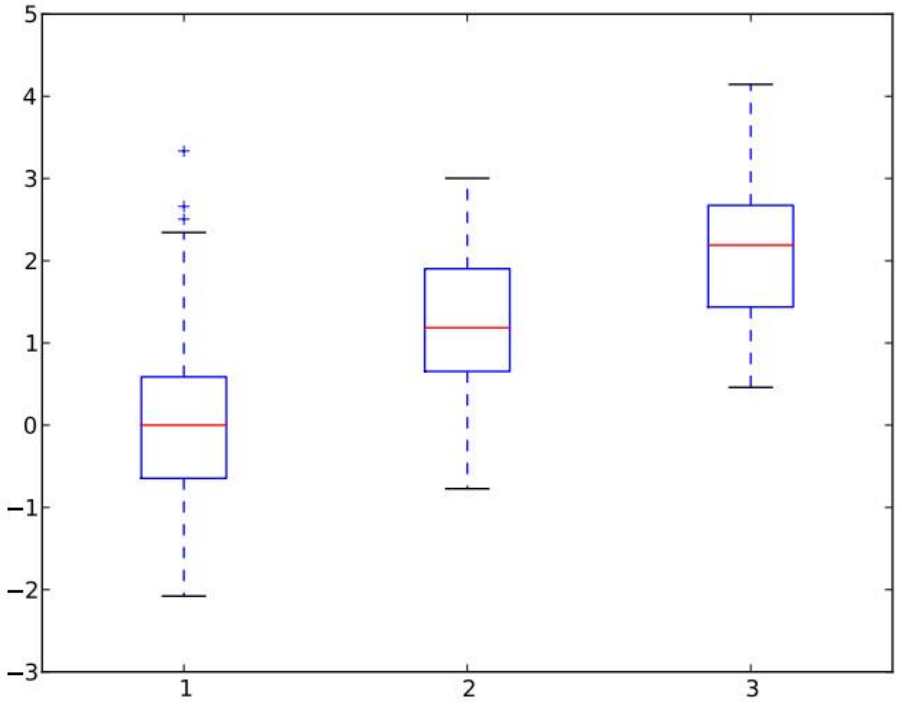
Βασικές εντολές

NumPy command	Note
a.ndim	returns the num. of dimensions
a.shape	returns the num. of rows and colums
arange(start,stop,step)	returns a sequence vector
linspace(start,stop,steps)	returns a evenly spaced sequence in the specificed interval
dot(a,b)	matrix multiplication
vstack([a,b])	stack arrays a and b vertically
hstack([a,b])	stack arrays a and b horizontally
where(a>x)	returns elements from an array depending on condition
argsort(a)	returns the sorted indices of an input array

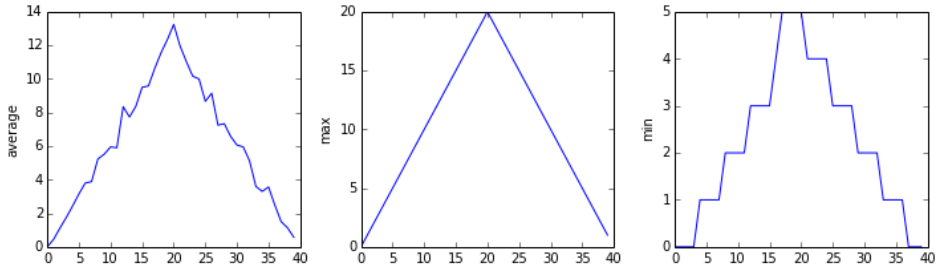
matplotlib (module)

Το matplotlib είναι το πιο συχνά χρησιμοποιούμενο πακέτο σχεδίασης σε Python

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111)
x1 = np.random.normal(0,1,50)
x2 = np.random.normal(1,1,50)
x3 = np.random.normal(2,1,50)
ax.boxplot([x1,x2,x3])
plt.show()
```



Ανάλυση δεδομένων από πολλά αρχεία




```
import numpy
import matplotlib.pyplot

filenames = sorted(glob.glob('inflammation*.csv'))
filenames = filenames[0:3]
for f in filenames:
    print(f)
    data = numpy.loadtxt(fname=f, delimiter=',')

    fig = matplotlib.pyplot.figure(figsize=(10.0, 3.0))

    axes1 = fig.add_subplot(1, 3, 1)
    axes2 = fig.add_subplot(1, 3, 2)
    axes3 = fig.add_subplot(1, 3, 3)

    axes1.set_ylabel('average')
    axes1.plot(numpy.mean(data, axis=0))

    axes2.set_ylabel('max')
    axes2.plot(numpy.max(data, axis=0))

    axes3.set_ylabel('min')
    axes3.plot(numpy.min(data, axis=0))

    fig.tight_layout()
    matplotlib.pyplot.show()
```

Biopython (module)

Python Tools for Computational Molecular Biology

Seq and SeqRecord objects

Bio.SeqIO – sequence input/output

Bio.AlignIO - alignment input/output

Bio.PopGen - population genetics

Bio.PDB - structural bioinformatics

Biopython's BioSQL interface

http://biopython.org/

Το Biopython είναι ένα σύνολο ελεύθερα διαθέσιμων εργαλείων για βιολογικούς υπολογισμούς γραμμένα στην Python από μια διεθνή ομάδα προγραμματιστών. Πρόκειται για μια κατανεμημένη συνεργατική προσπάθεια για την ανάπτυξη βιβλιοθηκών και εφαρμογών της Python που καλύπτουν τις ανάγκες της τρέχουσας και μελλοντικής εργασίας στη βιοπληροφορική

Uses	**Note**
Blast	finds regions of local similarity between sequences
ClustalW	multiple sequence alignment program
GenBank	NCBI sequence database
PubMed and Medline	Document database
ExPASy	SIB resource portal (Enzyme and Prosite)
SCOP	Structural Classification of Proteins (e.g. 'dom','lin')
UniGene	computationally identifies transcripts from the same locus
SwissProt	annotated and non-redundant protein sequence database

Biopython (Fasta files)

```
import Bio          (Biopython module)
```

Ανάγνωση

```
from Bio import SeqIO
from Bio.SeqRecord import SeqRecord
with open("example.fasta", "rU") as file:
for seq_record in SeqIO.parse( file, "fasta"):
    print seq_record.id
    print seq_record.seq
    my_seq = seq_record.seq
    #reverse the sequence
    print my_seq[::-1]
```

Εγγραφή

```
SeqIO.write(sequences, "example.fasta", "fasta")
```

Κλείσιμο

```
file.close()
```

Biopython (Swiss-Prot files)

```
FT   CHAIN         1    261       RPII140-upstream gene protein.
FT                                /FTId=PRO_0000064352.
FT   TRANSMEM     67     87       Potential.
FT   TRANSMEM    131    151       Potential.
FT   TRANSMEM    183    203       Potential.
FT   CONFLICT     64     64       S -> F (in Ref. 1).
SQ   SEQUENCE   261 AA;  29182 MW;  5DB78CF6CFC4435A CRC64;
     MNFLWKGRRF LIAGILPTFE GAADEIVDKE NKTYKAFLAS KPPEETGLER LKQMFTIDEF
     GSISSELNSV YQAGFLGFLI GAIYGGVTQS RVAYMNFMEN NQATAFKSHF DAKKKLQDQF
     TVNFAKGGFK WGWRVGLFTT SYFGIITCMS VYRGKSSIYE YLAAGSITGS LYKVSLGLRG
     MAAGGIIGGF LGGVAGVTSL LLMKASGTSM EEVRYWQYKW RLDRDENIQQ AFKKLTEDEN
     PELFKAHDEK TSEHVSLDTI K
```

Ανάγνωση

```
from Bio import SwissProt
with open("transmem.swiss", "rU") as file:
for record in SwissProt.parse( file):
   print record.sequence
   print record.features
   print record.accessions
```

record

```
>>> dir(record)
['__doc__', '__init__', '__module__', 'accessions', 'annotation_update',
'comments', 'created', 'cross_references', 'data_class', 'description',
'entry_name', 'features', 'gene_name', 'host_organism', 'keywords',
'molecule_type', 'organelle', 'organism', 'organism_classification',
'references', 'seqinfo', 'sequence', 'sequence_length',
'sequence_update', 'taxonomy_id']
```

Biopython
File Format Conversion

GenBank to Fasta

Bio.SeqIO.parse()

```
from Bio import SeqIO
with open("cor6_6.gb", "rU") as input_handle:
    for record in SeqIO.parse(input_handle, "genbank"):
        print record
```

Bio.SeqIO.write()

```
from Bio import SeqIO

with open("cor6_6.gb", "rU") as input_handle,
open("cor6_6.fasta", "w") as output_handle:
    sequences = SeqIO.parse(input_handle, "genbank")
    SeqIO.write(sequences, output_handle, "fasta")
```

Bio.SeqIO.convert()

```
from Bio import SeqIO
SeqIO.convert("cor6_6.gb", "genbank", "cor6_6.fasta", "fasta")
```

Κανονικές Εκφράσεις (Regular Expressions)

Οι κανονικές Εκφράσεις χρησιμοποιούνται για Αναζήτηση προτύπων σε ένα κείμενο

- protein domains
- DNA transcription factor binding motifs
- restriction enzyme cut sites
- degenerate PCR primer sites
- runs of mononucleotides

```
import re

dna = "ATCGCGAATTCAC"
if re.search(r"GG(A|T)CC", dna):
    print("restriction site found!")
```

```
dna = "CGATNCGGAACGATC"
m = re.search(r"[^ATGC]", dna)

if m:
    print("ambiguous base found!")
    print("at position " + str(m.start()))
```

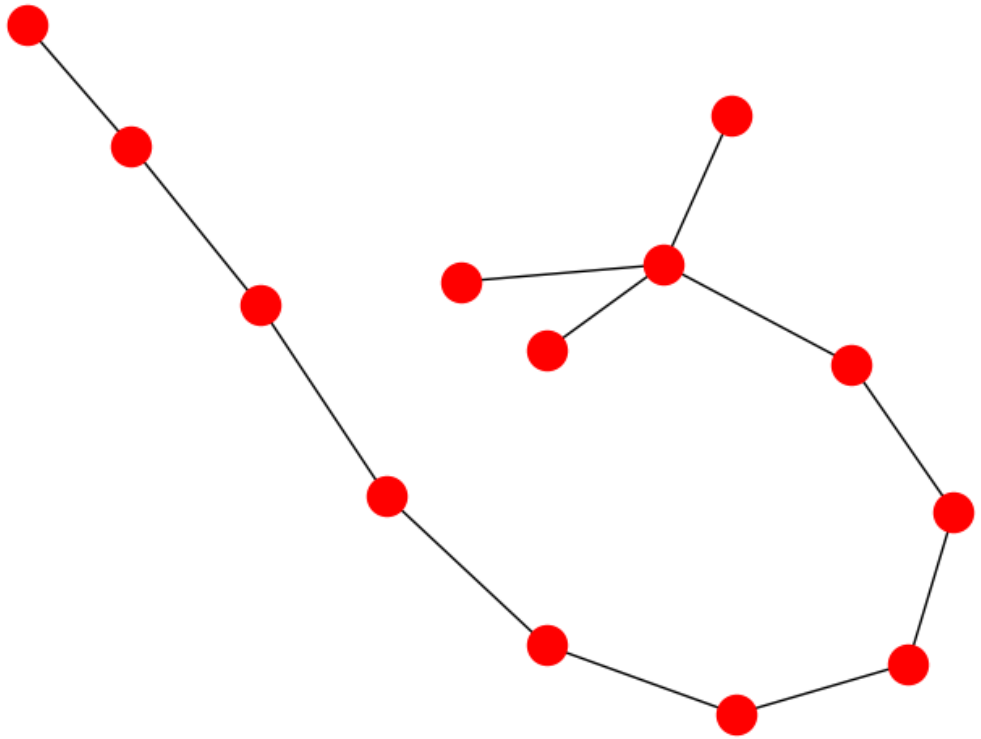
```
dna = "CTGCATTATATCGTACGAAATTATACGCGCG"

matches = re.finditer(r"[AT]{6,}", dna)

result = []
for m in matches:
    result.append(m.group())

print(result)
```

ACTGC**ATTATAT**CGTACG**AAATTATA**CGCGCG

NetworkX (module)

https://networkx.github.io/

NetworkX είναι ένα πακέτο Python για τη δημιουργία, το χειρισμό και τη μελέτη της δομής, δυναμικής, και τις λειτουργίες των πολύπλοκων δικτύων



Features

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source 3-clause BSD license
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

NetworkX (module)

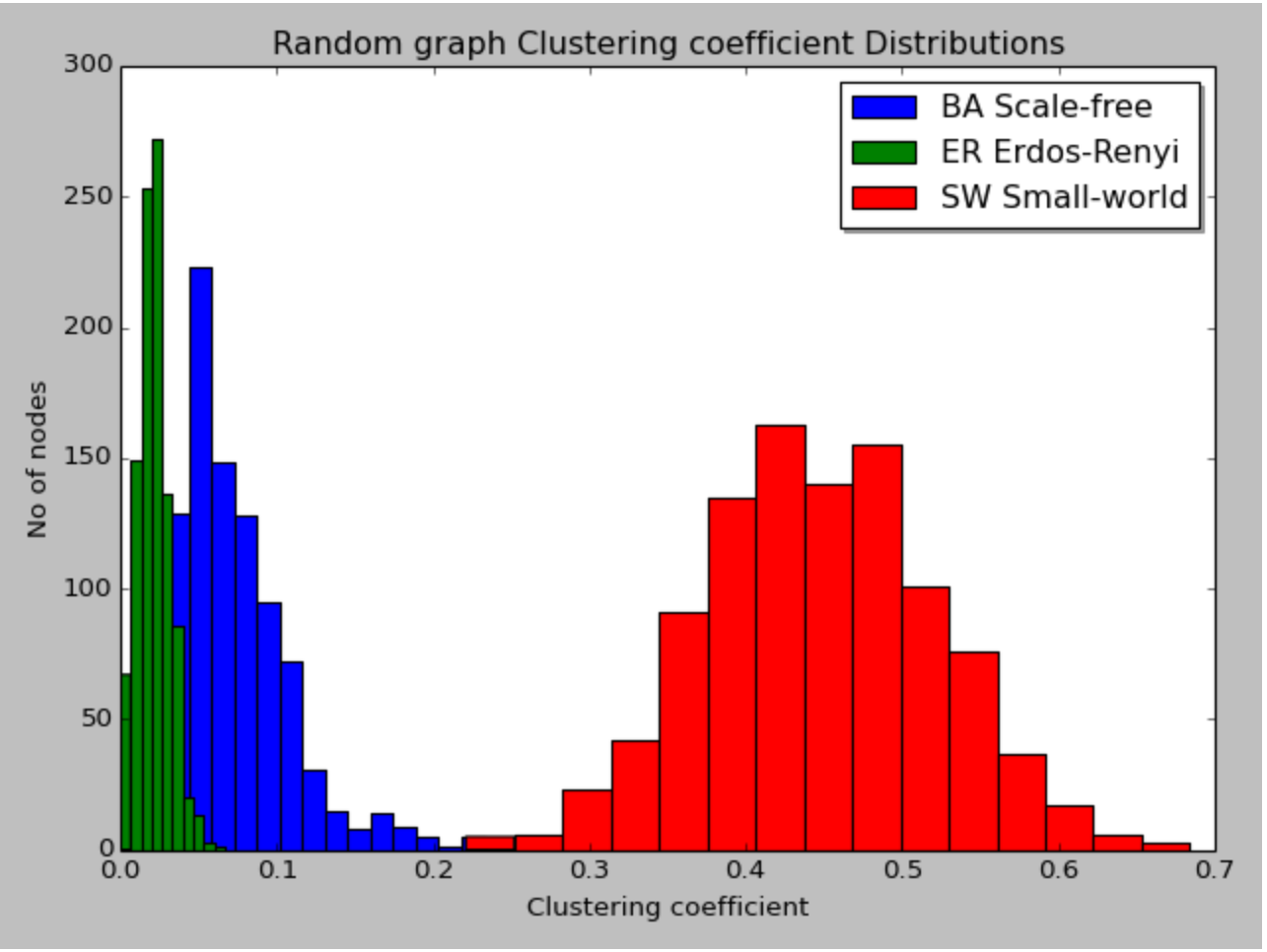
δημιουργία 3 διαφορετικών γράφων

υπολογισμός μέτρων συνολικής δομής

```
import networkx as nx
import matplotlib.pyplot as plt


gER = nx.erdos_renyi_graph(1000, 0.1)
gWS = nx.watts_strogatz_graph((1000, 3, 0.1)
gBA = nx.barabasi_albert_graph(1000, 3)

# Degree of each node, dictionary
erDegrees = nx.degrees(gER)
wsDegrees = nx.degrees(gWS)
baDegrees = nx.degrees(gBA)

# Clustering coefficient of each node.
erClustC = nx.clustering(gER)
wsClustC = nx.clustering(gWS)
baClustC = nx.clustering(gBA)

ddER = list(dER.values())
ddSW = list(dSW.values())
ddBA = list(dBA.values())
```

NetworkX (module)

ιστογράμματα για να παρατηρήσουμε τις κατανομές των 2 μέτρων στα 3 διαφορετικά δίκτυα




```
#Plot Clustering distribution
plt.hist(ccBA, bins=20, label="BA Scale-free")
plt.hist(ccER, bins=10, label="ER Erdos-Renyi")
plt.hist(ccSW, bins=15, label="SW Small-world")
plt.title("Random graph Clustering coefficient
Distributions")
plt.xlabel("Clustering coefficient")
plt.ylabel("No of nodes")
plt.legend(loc ='upper right',shadow = True )
plt.show()
```

Εξαιρέσεις (Exceptions)

Ορισμός

Οι εξαιρέσεις μας επιτρέπουν να διαχειριστούμε μια περίπτωση ενός συμβάντος για να αποφύγουμε ορισμένα λάθη που αλλάζουν την φυσιολογική ροή του προγράμματος.

Παράδειγμα

- Άνοιγμα ενός αρχείου που δεν υπάρχει
- Διαίρεση με το 0

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is any exception, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

IOError
RuntimeError
ZeroDivisionError
FloatingPointError
ArithmeticError
SystemExit
StopIteration
EOFError
ImportError

Παράδειγμα

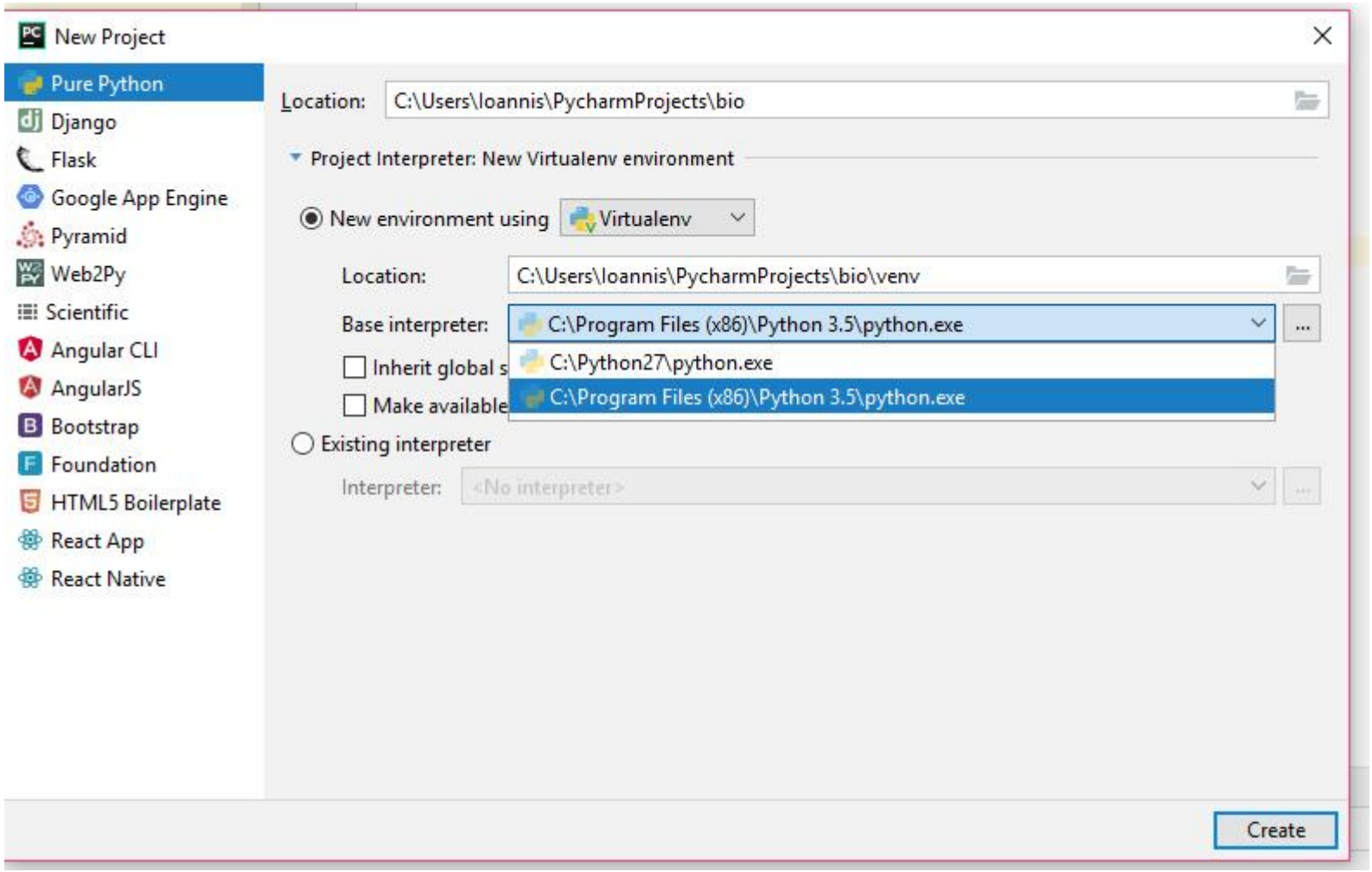
```
try :
   f = open( 'myfile.txt ‘ )
   s = f .readline ( )
   i = int ( s.strip ( ) )
except IOError as err :
   print ( " I/O error : { } " . format ( err ) )
except ValueError :
   print ( "Could not convert data to an integer . " )
except :
   print ( "Unexpected error : " , sys.exc_info ( ) [ 0 ] )
   raise
```

Virtual Environment (VirtualEnv)

Ένα εικονικό περιβάλλον, απλά, είναι ένα απομονωμένο αντίγραφο εργασίας της Python το οποίο σας επιτρέπει να εργάζεστε σε ένα συγκεκριμένο έργο χωρίς να ανησυχείτε να επηρεάζετε άλλα έργα

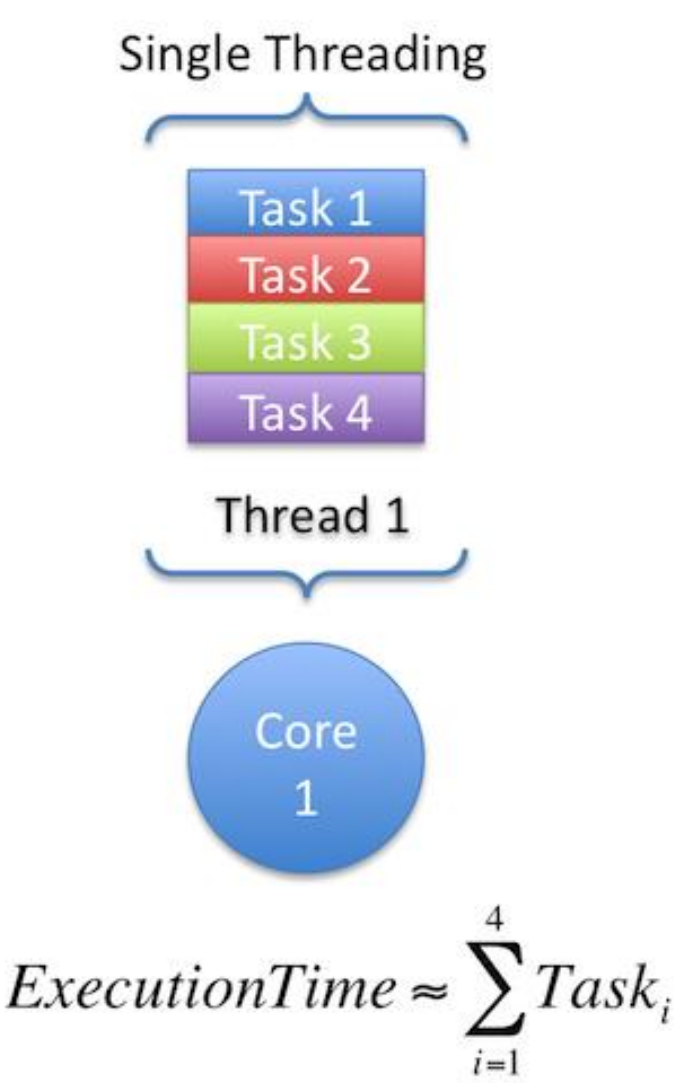
Επιτρέπει πολλαπλές εγκαταστάσεις δίπλα-δίπλα της Python, μία για κάθε έργο.

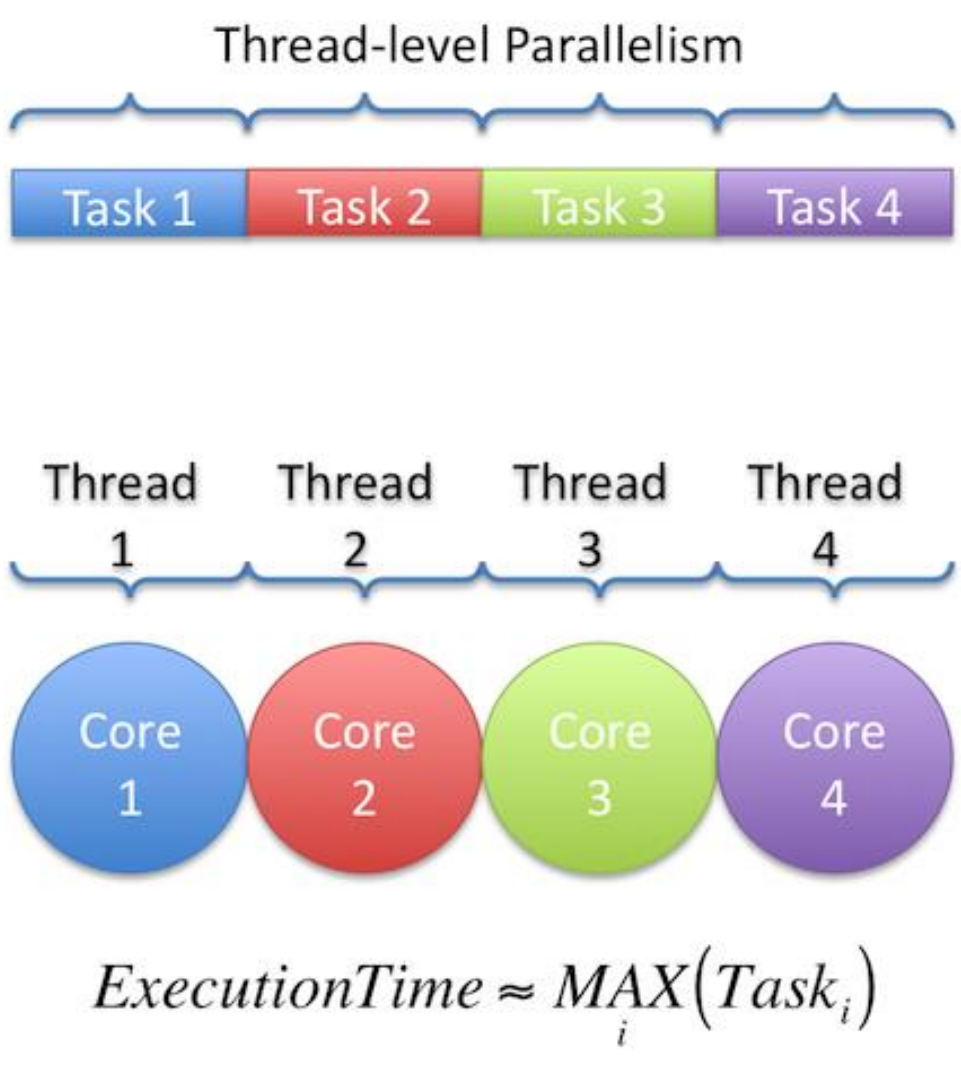
Στην πραγματικότητα δεν εγκαθιστά ξεχωριστά αντίγραφα της Python, αλλά παρέχει ένα έξυπνο τρόπο για να διατηρούνται απομονωμένα διαφορετικά περιβάλλοντα έργου.



Παραλληλοποίηση (Multiprocessing)

Οι CPU με πολλούς πυρήνες έχουν γίνει το πρότυπο στην πρόσφατη ανάπτυξη σύγχρονων αρχιτεκτονικών υπολογιστών





Παραλληλοποίηση (Multiprocessing)

```
# Prepare data

with open("sequences.fasta", "r", \encoding="utf-8") as
myfile

for line in myfile:
  data.append(line)

"""Returns sequence Length"""
def sequenceLen(seq):
      return len(seq)

for seq in data:
    results.append(sequenceLen(seq))

print(results)
```

```
import multiprocessing as mp
# Prepare data
with open("sequences.fasta", "r", \encoding="utf-8") as f

for line in f:
  data.append(line)

"""Returns sequence Length"""
def sequenceLen(seq):
      return len(seq)

# Step 1: Init multiprocessing.Pool()
pool = mp.Pool(mp.cpu_count())

# Step 2:
results = [pool.apply(sequenceLen, args=(seq)) for seq in data]

# Step 3: Don't forget to close
pool.close()

print(results)
```

Τέλος Ερωτήσεις

Ευχαριστώ