Προηγμένες αρχιτεκτονικές υπολογιστών και προγραμματισμός παραλλήλων συστημάτων

LAB-01. Εισαγωγή στην Python

Βασική περιγραφή

- Υψηλού επιπέδου γλώσσα προγραμματισμού γενικού σκοπού.
- Χωρίς compiler. Τα προγράμματα Python εκτελούνται από τον Python interpreter.
- Ευκολότερη από την Java.
- Πολύ ευκολότερη από τις C/C++.
- Ταχεία ανάπτυξη εφαρμογών.
- Προγράμματα μικρότερου μήκους – λιγότερος κώδικας.

Python vs. C++ (performance, 1)

- Φυσικά, ότι κερδίζεται σε χρόνο ανάπτυξης και σε κόπο υλοποίησης χάνεται σε ταχύτητα εκτέλεσης.
  - Ισχύει και το αντίστροφο.
- Σε σχέση με τα προγράμματα C/C++, αυτά της Python είναι 1-3 τάξεις μεγέθους αργότερα…
  - 10 έως 1000 φορές!
- Κατά κανόνα δεσμεύουν (πολύ) περισσότερη μνήμη
- Python: άριστη γλώσσα για εκπαίδευση και γρήγορους ελέγχους ορθότητας αλγορίθμων.
- Production περιβάλλον: Μάλλον C/C++/Java.

Python 3 vs. C++ (performance, 2)

mandelbrot

source	secs	mem	gz	busy	cpu load
C++ g++	**0.84**	34,608	3542	3.27	98% 98% 96% 99%
Python 3	172.58	12,216	688	689.62	100% 100% 100% 100%

binary-trees

source	secs	mem	gz	busy	cpu load
C++ g++	**1.12**	200,184	890	4.05	88% 88% 88% 99%
Python 3	49.35	278,628	589	174.26	95% 86% 86% 86%

spectral-norm

source	secs	mem	gz	busy	cpu load
C++ g++	**0.72**	2,360	1044	2.85	99% 100% 99% 100%
Python 3	118.40	22,912	407	470.86	99% 99% 99% 100%

fasta

source	secs	mem	gz	busy	cpu load
C++ g++	**1.04**	4,464	2344	3.87	92% 93% 93% 93%
Python 3	39.10	846,628	1947	70.52	43% 8% 61% 69%

n-body

source	secs	mem	gz	busy	cpu load
C++ g++	**4.09**	1,748	1808	4.12	0% 100% 1% 0%
Python 3	586.17	8,012	1196	589.84	0% 0% 0% 100%

k-nucleotide

source	secs	mem	gz	busy	cpu load
C++ g++	**1.93**	156,432	1631	5.93	69% 70% 70% 98%
Python 3	46.37	239,472	1967	176.69	94% 95% 98% 94%

fannkuch-redux

source	secs	mem	gz	busy	cpu load
C++ g++	**8.07**	1,848	980	31.38	93% 98% 100% 98%
Python 3	367.49	12,288	950	1,454.68	100% 98% 99% 98%

reverse-complement

source	secs	mem	gz	busy	cpu load
C++ g++	**0.63**	499,624	2093	0.64	0% 0% 100% 2%
Python 3	7.16	1,005,968	814	10.65	20% 53% 47% 29%

Χρήσεις Python

- Machine Learning, Artificial Intelligence.
  - Διάσημα πακέτα:
    - Google's tensorflow
    - Scikit learn
- Data Science/Data analysis.
- Web programming/Web application development.
- Εκπαιδευτική.
- Ανάπτυξη Desktop GUIs.
  - wxWidgets
  - Kivy, for writing multitouch applications.
  - Qt via pyqt or pyside

IDEs

- PyCharm - https://www.jetbrains.com/pycharm/
- PyDev - http://www.pydev.org/
- Jupyter - https://jupyter.org/
  - Web-based
- Anaconda - https://www.anaconda.com/
  - birthplace of Python data science
  - Not free

Μεταβλητές και εκτύπωση στην οθόνη

```
print("Hello world")
a = 15
print (type(a))
b = 3.14
print (type(b))
c = 'Parallel Programming'
print (c)
print (type(c))

#X[I:J] the selected characters are
from I until J but without J
print ("c[0]=" + c[0])
print ("c[1]=" + c[1])
print ("c[-1]=" + c[-1])
print(c[1], c[-1], c[1:3], c[1:10])
```

```
Hello world
<class 'int'>
<class 'float'>
Parallel Programming
<class 'str'>
c[0]=P
c[1]=a
c[-1]=g
a g ar arallel P
```

String concatenation

```
c = 'Parallel Programming'
print (c + c)
print (c + " LAB")
```

```
Parallel ProgrammingParallel Programming
Parallel Programming LAB
```

```
text = ('distributed ' 'systems ' 'and ' 'python '
'programming ')
```

```
distributed systems and python programming
```

+ operator

- Μεταξύ αριθμών: πρόσθεση
- Μεταξύ string: concatenation
- Μεταξύ αριθμών και string: Type Error

```
a = 15
b = 3.14
c = 'Parallel Programming'

print (c+c)
print (c + " LAB")
print (a+b)
print (a+c)
```

```
Parallel ProgrammingParallel
Programming
Parallel Programming LAB
18.14
Traceback (most recent call last):
  File ……..
TypeError: unsupported operand
type(s) for +: 'int' and 'str'
```

Type Casting (int)

- Μετατροπή τύπου μεταβλητής.
- Από integer σε float.

```
a = 15
print(type(a), ':', a)
atofloat = (float)(a)
print(type(atofloat), ':', atofloat)
```

```
<class 'int'> : 15
<class 'float'> : 15.0
```

- Από integer σε string.

```
a = 15
print(type(a), ':', a)
atostring = (str)(a)
print(type(atostring), ':', atostring)
```

Type Casting (float/string)

- Από float σε integer και string.

```
# b is a float
b = 3.14

# b to int
btoint = (int)(b)
print(type(btoint), ':', btoint)

# b to string
btostring = (str)(b)
print(type(btostring), ':', btostring)
```

```
<class 'int'> : 3
<class 'str'> : 3.14
```

- Από string σε integer/float (υπό περιορισμούς).

```
c = 'a string'
ctoint = (int)(c)
ctofloat = (float)(c)

d = '12.55'
dtoint = (int)(d)
dtofloat = (float)(d)
```

```
e='5'
etoint = (int)(e)
etofloat = (float)(e)
```

Αριθμητικοί τελεστές

- Βασικές πράξεις:
  - + πρόσθεση
  - – αφαίρεση
  - * πολλαπλασιασμός
  - ** ύψωση σε δύναμη (5 ** 3 = 5^3)
  - / κλασική διαίρεση, επιστρέφει float (17 / 3 = 5,66…)
  - // ακέραια διαίρεση, επιστρέφει int (17 // 3 = 5)
  - % επιστρέφει το υπόλοιπο (mod) της διαίρεσης (17%3=2)

Strings

- Αποθηκεύουν αλφαριθμητικές τιμές.
- Indexing: Οι χαρακτήρες ενός string δεικτοδοτούνται με βάση το 0.

```
c = 'Parallel Programming'
print ("c[0]=" + c[0])
print ("c[1]=" + c[1])
print ("c[-1]=" + c[-1])
```

```
c[0]=P
c[1]=a
c[-1]=g
```

- Slicing: Επιστροφή υποσυνόλου του αρχικού string.
- `c[i:j]`: Επιστροφή του υποσυνόλου του c που ξεκινάει από τον χαρακτήρα `i` και λήγει στον `j`.

```
c = 'Parallel Programming'
print(c[1], c[-1], c[1:3], c[1:10])
```

```
a g ar arallel P
```

Μέθοδοι διαχείρισης strings (1)

- https://docs.python.org/3/library/stdtypes.html#string-methods
- `str.lower()`, `str.upper()`, `str.casefold()`
  - Μετατροπή όλων των χαρακτήρων σε πεζούς, κεφαλαίους, και πεζούς χωρίς διακρίσεις.
- Έλεγχοι: `str.isalnum()`, `str.isalpha()`, `str.isupper()`, `str.isnumeric()`, `str.isdecimal()`, `str.islower()`,…
- `str.replace(old, new, count)`: Αντικατάσταση του υποσυνόλου `old` με το υποσύνολο `new` μέσα στο `str`.
  - Αντικατάσταση μόνο των πρώτων count εμφανίσεων

Μέθοδοι διαχείρισης strings (2)

- https://docs.python.org/3/library/stdtypes.html#string-methods
- `str.split(sep)`: Επιστρέφει λίστα που περιέχει τα υποσύνολα του `str` που χωρίζονται από `sep`.

```
str = 'Parallel Programming with Python'
list1 = str.split(' ')
print(list1)
list2 = str.split('P')
print(list2)
```

```
['Parallel', 'Programming', 'with', 'Python']
['', 'arallel ', 'rogramming with ', 'ython']
```

- `str.strip(chars)`: Αφαιρεί από την αρχή και το τέλος του `str` όλους τους `chars`.

```
str = '   many spaces   '
print(str.strip(' '))
str = 'www.example.com'
print(str.strip('cmowz.'))
```

```
many spaces
example
```

Λίστες

- Αποθηκεύουν τιμές σε μία ενιαία δομή.
- Οι τιμές μπορεί να είναι ίδιου ή διαφορετικού τύπου.
- Μπορεί ακόμα να είναι και άλλες (nested) λίστες!

```
squares = [1, 4, 9, 16, 25]                    # List declaration
newlist = squares + [36, 49, 64, 81, 100]      # List concatenation
print(newlist)
cubes = [1, 8, 27, 65, 125]                    # Another List
cubes[3] = 64                                  # Direct access by index
cubes.append(216)                              # Append another element
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g'] # Another List
letters[2:5] = ['C', 'D', 'E']                 # Replace some values directly
listlength = len(letters)                      # Get the length of the list
```

```
mixed = (squares,letters)
print(mixed)
print(mixed[0])
print(mixed[0][2], mixed[1][3])
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
([1, 4, 9, 16, 25], ['a', 'b', 'C',
'D', 'E', 'f', 'g'])
[1, 4, 9, 16, 25]
9 D
```

Λίστες: Indexing και Slicing

- Ισχύει ότι και στα strings.
- Zero-based indexing.
- Τα strings μπορούν να θεωρηθούν σαν λίστες χαρακτήρων.

Έλεγχος Ροής (1)

- Κλασικό if – elseif – else statement:

```
x = int(input("Please enter an integer: "))
if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

- Προσοχή: δεν υπάρχουν brackets όπως στις C-like γλώσσες.
- Τα περιθώρια (indentation) οριοθετούν τα μπλοκ κώδικα.

Έλεγχος Ροής (2)

- Οι συνθήκες ελέγχου δεν μπαίνουν σε παρένθεση όπως στις C-like γλώσσες.
- Μπορούν να είναι και πιο πολύπλοκες, με λογικές εκφράσεις.

```
x = int(input("Please enter an integer: "))
y = int(input("Please enter another integer: "))
if x < 0 and y < 0:
    x = 0
    y = 0
    print('Negative changed to zero')
elif x == 0 or y == 0:
    print('At least one is Zero')
else:
    print('More')
```

Loops – Βρόχοι επανάληψης (1)

- Επαναληπτικές διαδικασίες ελεγχόμενες από λογικές εκφράσεις.

```
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
```

```
cat 3
window 6
defenestrate 12
```

- Μέθοδος `range()`: Δημιουργία αριθμητικών προόδων.

```
for i in range(5):
    print(i)

x = sum(range(10))
print(x)

y = list(range(10))
print(y)
```

```
0
1
2
3
4
45
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Loops – Βρόχοι επανάληψης (2)

- Παραδείγματα…

```
words = ['you', 'never', 'stare', 'the', 'stars']
for w in words[ 2:4 ]:
    print(w, len(w))

a = ['Advanced', 'Computer', 'Architecture', 'And', 'Parallel', 'Programming']
for i in range(len(a)):
    print(i, a[i])

for x in 'python':
    print(x)
```

Διαχείριση αρχείων

- Δύο παράμετροι απαιτούνται:
  - Filename και τρόπος πρόσβασης (access mode).
- Άνοιγμα για εγγραφή και εγγραφή:

```
f = open('test.txt', 'w')
f.write('Just write to a test file!')
f.close()
```

- Άνοιγμα για ανάγνωση και ανάγνωση:

```
f = open('test.txt', 'r')
t = f.read()
print(t)
f.close()
```

ή

```
with open('test.txt') as file:
    data = file.read()
    print(data)
```

- Ανάγνωση γραμμή – γραμμή:

```
linecnt = 0
with open('test.txt') as file:
    for line in file:
        linecnt += 1
        print (linecnt, line)
```

Συναρτήσεις (1)

- Η Python διαθέτει μία μεγάλη συλλογή συναρτήσεων.
- Παρέχει όμως στο χρήστη και τη δυνατότητα δημιουργίας άλλων (user-defined functions).
- Οι συναρτήσεις δέχονται ως όρισμα δεδομένα, πραγματοποιούν πράξεις επί αυτών των δεδομένων και (ίσως να) επιστρέφουν αποτέλεσμα.
- Ορίζονται με τη δεσμευμένη λέξη `def`.
- Και ακολουθεί το όνομα της συνάρτησης με τα ορίσματα (arguments) της.
- Η κλήση της συνάρτησης γίνεται με το ίδιο όνομα.

Ορισμός και κλήση συναρτήσεων

```
def foo():
    print("This is a function!")
    return

foo()
```

- Στην Python όλα τα ορίσματα περνούν με αναφορά (by reference). Δηλαδή, αν η συνάρτηση αλλάξει την τιμή του ορίσματος, αυτή η αλλαγή είναι ορατή και έξω από τη συνάρτηση.

```
def foo2(value1):
    value2 = 5
    value1 += value2
    return value1

x = 10
y = foo2(x)
print(y, x)
```

```
def changelist(thelist):
    thelist.append(100)
    return

thelist = [10,50,80]
changelist(thelist)
print ("My list now has: ", thelist)
```

Ορίσματα μεταβλητού μήκους

- Κάποιες φορές προκύπτει η ανάγκη κλήσης μίας συνάρτησης με περισσότερα ορίσματα από ότι έχει οριστεί.
- Αυτά τα ορίσματα λέγονται variable-length arguments και δεν κατονομάζονται στον ορισμό της συνάρτησης.

```
def foo3(*value):
    for var in value:
        print (var)
    return

foo3(5)
foo3(10,50,100)
```

```
5
10
50
100
```