Τα Βασικά για την Python

Μωυσής Α. Μπουντουρίδης

Τμήμα Μαθηματικών Πανεπιστημίου Πάτρας

mboudour@upatras.gr

Φεβρουάριος–Μάρτιος 2013

Πίνακας Περιεχομένων

Python

- Η Python είναι μια ερμηνευτική γλώσσα, δηλαδή, εκτελείται χωρίς να χρειάζεται να περάσει από την διαδικασία της σύνταξης.
- Η επίσημη Python διατίθεται για κατέβασμα (μαζί με οδηγίες εγκατάστασης) για όλα τα λειτουργικά συστήματα (**Mac**, **Windows** και **Linux**) από το python.org, αλλά μπορεί το σάϊτ αυτό να **μην** είναι η καλύτερη επιλογή. Ένας λόγος είναι ότι, μαζί με την Python, θέλουμε να εγκαταστήσουμε και κάποιες άλλες βιβλιοθήκες ρουτινών επιστημονικού προγραμματισμού (όπως SciPy, Matplotlib, ipython κλπ.), οι οποίες δεν διατίθενται εκεί.
- Μια επίσης καλή διανομή είναι της Enthought, που δίνεται δωρεάν για ακαδημαϊκή χρήση (και διατίθεται για όλα τα λειτουργικά συστήματα).
- Σημειωτέον ότι στα **Linux**, η Python είναι προ-εγκατεστημένη.

- Ιδιαίτερα για τα **Windows** συνιστάται η διανομή ActivePython, ενώ δυο άλλες καλές επιλογές είναι η Python(x,y) και η Anaconda CE (η οποία υπάρχει επίσης και για **Mac**).
- Εδώ υπάρχουν κάποιες οδηγίες για την διαδικασία εγκατάστασης της Python σε **Windows**.

- **ΣΗΜΑΝΤΙΚΟ**: Όλοι οι κώδικες που θα χρησιμοποιήσουμε στο μάθημα έχουν δοκιμασθεί πάνω στην έκδοση 2.7.3 της Python κι αυτή την έκδοση πρέπει να εγκαταστήσετε (αντί της πιο τελευταίας έκδοσης 3.x, για την οποία χρειάζονται κάποιες ελάχιστες τροποποιήσεις στους κώδικες που θα χρησιμοποιούμε εδώ).

Εκτέλεση Προγραμμάτων της Python

- Τα **προγράμματα** ή **σκριπτ (scripts)** της Python είναι συνήθη **αρχεία κειμένου**.

- Κάθε πρόγραμμα της Python πρέπει:
  - να έχει την **κατάληξη** *.py,
  - να είναι αρχείο **εκτελέσιμο** και
  - να περιέχει την εξής **πρώτη γραμμή**:

```
#!/usr/bin/python
```

- Τα προγράμματα της Python μπορούν να γραφούν με οποιοδήποτε **κειμενογράφο (text editor)**.

- Η εκτέλεση ενός προγράμματος της Python μπορεί να γίνει είτε μέσα σε ένα **τερματικό (shell)** γράφοντας:

```
python file.py
```

ή

```
./file.py
```

εφόσον βέβαια βρισκόμαστε στον κατάλογο όπου φυλάσσεται το αρχείο.

- Είναι όμως προτιμητέα η εκτέλεση των προγραμμάτων Python μέσα από ένα **Ολοκληρωμένο Περιβάλλον Ανάπτυξης (Integrated Development Environment** ή, σε συντομογραφία, **IDE**).

- Μεταξύ πολλών, δυο εύχρηστα και συχνά χρησιμοποιούμενα IDE για την Python είναι τα εξής:
  - IDLE
  - Spyder

- Ένα επιπλέον προσόν των IDE είναι ότι αποτελούν ενιαία περιβάλλοντα για την γραφή και την εκτέλεση προγραμμάτων Python.

Αριθμοί

```
>>> 2 + 3
5
>>> 5 - 8
-3
>>> 2*3
6
>>> 2**3
8
>>> 2/3
0.6666666666666666
>>> 2//3
0
>>> 2.0/3
0.6666666666666666
>>> 3 % 2  # 3 modulo 2
1
>>> x = 3
>>> y = 5
>>> y/x
1.6666666666666667
>>> type(x)
<type 'int'>
>>> type(y/x)
<type 'float'>
```

Λέξεις (Strings)

```
>>> s = "hobbit"
>>> s.count("i")
1
>>> s.count("b")
2
>>> s.startswith("home")
False
>>> s.startswith("hob")
True
>>> s.endswith("t")
True
>>> s.upper()
'HOBBIT'
>>> r = s.replace("bb","b")
>>> r.replace("o","a")
'habit'
>>> s[0]
'h'
>>> s[3]
'b'
>>> s[10]
IndexError: string index out of range
>>> s[1], s[3], s[5]
('o', 'b', 't')
>>> s[0:4]
'hobb'
>>> s[:4]
'hobb'
>>> s[2:]
'bbit'
>>> s[3:]
'bit'
```

```
>>> s = "hobbit"
>>> s[-1]
't'
>>> s[-2]
'i'
>>> s[-3:]
'bit'
>>> s * 3
'hobbithobbithobbit'
>>> t = s.replace("t","t ")
>>> t * 3
'hobbit hobbit hobbit '
>>> t = " is good"
>>> r = (s+t).replace("h","H")
>>> print r
Hobbit is good
>>> r[6]
' '
>>> r[:]
'Hobbit is good'
>>> s1 = """
Triple quotes can be used to create
multi-line strings, which is useful
when you want to record a lot of test.
"""
>>> print s1

Triple quotes can be used to create
multi-line strings, which is useful
when you want to record a lot of test.
```

Λίστες

```
>>> X = [3, 5, 7]
>>> sum(X)
15
>>> max(X), min(X)
(7, 3)
>>> X[0] = "book"
>>> X
['book', 5, 7]
>>> del X[2]
>>> X
['book', 5]
>>> del X[1]
>>> Y = X + ["pen", "table"]
>>> Y
['book', 'pen', 'table']
>>> Z = Y * 2
>>> Z
['book', 'pen', 'table', 'book', 'pen',
    'table']
>>> Z[1:4]
['pen', 'table', 'book']
>>> U = ["a","b","c","d"]
>>> V = ["b1","b2","b3"]
>>> U1 = ["a",V,"c","d"] # nested list
>>> U1
['a', ['b1', 'b2', 'b3'], 'c', 'd']
>>> U[0] = "z"
>>> U[2] = "q"
>>> U.sort() # sorting
>>> U
['b', 'd', 'q', 'z']
```

```
>>> s = "book and pen on table"
>>> T = s.split(" ")
>>> T
['book', 'and', 'pen', 'on', 'table']
>>> del T[1], T[2] # Attention!!!
>>> T
['book', 'pen', 'table']
```

Λεξικά (Dictionaries)

```
>>> dict = {"to go": "a verb", "apple": "a noun", "green": "an adjective"}
>>> dict
{'green': 'an adjective', 'apple': 'a noun', 'to go': 'a verb'}
>>> dict["apple"]
'a noun'
>>> dict["apple"] = "not an adverb"
>>> dict
{'green': 'an adjective', 'apple': 'not an adverb', 'to go': 'a verb'}
>>> dict["apple"] = "not an adverb"
>>> dict
{'green': 'an adjective', 'apple': 'not an adverb', 'to go': 'a verb'}
>>> dict["apple"] = "a noun"
>>> dict.values()
['an adjective', 'a noun', 'a verb']
>>> dict.keys()
['green', 'apple', 'to go']
>>> dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
>>> dict
{'Age': 7, 'Name': 'Zara', 'Class': 'First'}
>>> dict['Age'] = 8
>>> dict['School'] = "DPS School"
>>> dict
{'School': 'DPS School', 'Age': 8, 'Name': 'Zara', 'Class': 'First'}
# Another way to update entries
>>> dict2 = {'School': "DPS School"}
>>> dict.update(dict2)
>>> print "Value : %s" % dict
Value : {'School': 'DPS School', 'Age': 7, 'Name': 'Zara', 'Class': 'First'}
# Deleting dictionary elements
>>> del dict['Name']  # remove entry with key 'Name'
>>> dict.clear()      # remove all entries in dict
>>> del dict          # delete entire dictionary
```

Tuples (Λίστες με αμετάβλητα στοιχεία)

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

>>> tup4 = () # The stuty tuple
>>> tup5 = (50,) # To write a tuple containing a single value

>>> print "tup1[0]: ", tup1[0]
tup1[0]:  physics
>>> print "tup2[1:5]: ", tup2[1:5]
tup2[1:5]:  (2, 3, 4, 5)

>>> tup6 = tup1 + tup2 #Updating tuples
>>> print tup6
('physics', 'chemistry', 1997, 2000, 1, 2, 3, 4, 5)

>>> del tup1 # Deleting a tuple

# Notice that replacement tup1[0] = "a" does not work in tuples!
```

Ώρα, Ημερομηνία και Ημερολόγια

```
>>> import time

>>> localtime = time.asctime( time.localtime(time.time()) )

>>> print "Local current time :", localtime

Local current time : Mon Feb 25 12:26:42 2013

>>> import calendar

>>> cal = calendar.month(2013, 2)

>>> print cal

    February 2013
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
```

Συγκρίσεις και Τροποποιήσεις Τιμών (Assignments)

```
# a == b checks whether a = b is True
# or False
>>> 6 == 7, 6 == 6
(False, True)
# a != b checks whether a = b is False
# or True
# <> is similar to !=
>>> 6 != 7, 6 != 6, 3 <> 5
(True, False, True)
>>> 6 > 7, 6 > 6, 6 < 7
(False, False, True)
>>> 6 >= 7, 6 >= 6, 6 <= 7
(False, True, True)

# A simple decision
>>> a = 100
>>>
if (a == 100): print "Value of a is 100"

Value of a is 100
>>>
if (a != 200): print "Value of a is \
not 200"

Value of a is not 200
```

```
# c = a + b will assigne value of a + b
# into c
>>> a = 2 + 3
>>> a
5
# c += a is equivalent to c = c + a
# and respectively with -=, *=, **=,
# /=, %=
>>> a += 4
>>> a
9
>>> a -= 5
>>> a
4
>>> a *= 3
>>> a
12
>>> a **= 2
>>> a
144
>>> a /= 10
>>> a
14
>>> a %= 3
>>> a
2
```

Βρόχοι (Loops) και Δηλώσεις (Statements)

while loop

```
>>> count = 0
>>> while (count < 5):
        print 'The count is:', count
        count = count + 1

The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
```

else used with while

```
>>> count = 0
>>> while (count < 5):
        print count, " is  less than 5"
        count = count + 1
    else:
        print count, " is equal to 5"

0  is  less than 5
1  is  less than 5
2  is  less than 5
3  is  less than 5
4  is  less than 5
5  is equal to 5
```

for loop

```
>>> fruits = ['banana', 'apple',
'mango']
>>> for fruit in fruits:
        print 'This is a:', fruit

This is a fruit: banana
This is a fruit: apple
This is a fruit: mango
```

else used with for

```
>>> for num in range(10,20):
        for i in range(2,num):
            if num%i == 0:
                j=num/i
                print '%d equals %d *
            break
    else:
        print num, 'is a prime number'

10 equals 2 * 5
12 equals 2 * 6
14 equals 2 * 7
16 equals 2 * 8
18 equals 2 * 9
19 is a prime number
```

Nested **for loop**

```
>>> i = 2
>>> while(i < 30):
        j = 2
        while(j <= (i/j)):
                if not(i%j): break
                j = j + 1
        if (j > i/j) : print i,
                                " is prime"
        i = i + 1

2  is prime
3  is prime
5  is prime
7  is prime
11  is prime
13  is prime
17  is prime
19  is prime
23  is prime
29  is prime
```

The **if–else** Conditional Statement

```
>>> x = int(raw_input("Please enter
                                an integer: "))
>>> if x % 2:
        print "x is odd"
    else:
        print "x is even"
```

The **range()** function

```
>>>
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(5, 10)
[5, 6, 7, 8, 9]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> range(-10, -100, -30)
[-10, -40, -70]
```

To iterate over the indices of a sequence

```
>>> a=['Mary','had','a','little','lamb']
>>> for i in range(len(a)):
...     print i, a[i]
...
0 Mary
1 had
2 a
3 little
4 lamb
```

The **pass** statement: It does nothing

The **break** statement

```
>>> for n in range(2, 7):
...     for x in range(2, n):
...         if n % x == 0:
                print n, 'equals', x,
                                '*', n/x
                break
...     else:
...         print n, 'is a prime number'
...
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
```

The **continue** statement

```
>>> for num in range(2, 6):
...     if num % 2 == 0:
            print "Found an even number",
                                    num
            continue
...     print "Found a number", num
Found an even number 2
Found a number 3
Found an even number 4
Found a number 5
```

Συναρτήσεις (Functions)

Σειρά Fibonacci

```
>>> def fib(n):
# write Fibonacci series up to n
...     a, b = 0, 1
...     while a < n:
...         print a,
...         a, b = b, a+b
...
>>> fib(2000)    # call it
0 1 1 2 3 5 8 13 21 34 55 89 144 233
                        377 610 987 1597
```

Σειρά Fibonacci ως λίστα

```
>>> def fib2(n):
...     result = []
...     a, b = 0, 1
...     while a < n:
...         result.append(a)
...         a, b = b, a+b
...     return result
...
>>> fib2(100)     # call it
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Ευκλείδεια απόσταση

```
>>> def dist(X, Y):
        Z = [(x - y)**2 for x, y in
                                zip(X, Y)]
        return sum(Z)**0.5
>>> X = [1, 2, 3, 4]
>>> Y = [10, 9, 8, 7]
>>> dist(X, Y)
12.806248474865697
```

Συντελεστής συσχέτισης Pearson

```
>>> from itertools import imap
>>> def pearsoncorrelation(x, y):
  n = len(x)
  sum_x = float(sum(x))
  sum_y = float(sum(y))
  sum_x_sq = sum(map(lambda x: pow(x,2),
                                      x))
  sum_y_sq = sum(map(lambda x: pow(x,2),
                                      y))
  psum = sum(imap(lambda x, y: x*y, x,
                                      y))
  num = psum - (sum_x * sum_y/n)
  den = pow((sum_x_sq - pow(sum_x,2)/n)
      * (sum_y_sq - pow(sum_y,2)/n),0.5)
  if den == 0: return 0
  return num / den
>>> pearsoncorrelation(X, Y)
-1.0
```

Δημιουργία Κλάσεων (Classes)

```
>>> class Student:
   'Common base class for all students'
   stuCount = 0

   def __init__(self, name, year):
      self.name = name
      self.year = year
      Student.stuCount += 1

   def displayCount(self):
     print "Total Students %d"
                % Student.stuCount

   def displayStudent(self):
      print "Name : ", self.name,  ",
                 Year: ", self.year
>>> stu1 = Employee("Zara", 2008)
>>> stu2 = Employee("Mary", 2009)
```

Προσθήκη–Αφαίρεση Χαρακτηριστικών (Attributes)

```
>>> stu1.grade = 7
# Add an 'grade' attribute.
>>> stu1.grade = 8
# Modify 'grade' attribute.
>>> del stu1.grade
# Delete 'grade' attribute.
```

Built-In Class Attributes

```
>>> print *.__doc__
>>> print *.__name__
>>> print *.__module__
>>> print *.__bases__
>>> print *.__dict__
```

Destroying Objects Class

```
>>> def __del__()
```

Class Inheritance

```
>>> class SubClassName (ParentClass1[,
                        ParentClass2, ...]):
   'Optional class documentation string'
   class_suite
```

Παράδειγμα: Επαναλήψεις της Τετραγωνικής Απεικόνισης

Τετραγωνική Απεικόνιση $x_{t+1} = 4x_t(1 - x_t)$

```
class QuadMap:

    def __init__(self, initial_state):
        self.x = initial_state

    def update(self):
        "Apply the quadratic map to
                    update the state."
        self.x = 4*self.x*(1-self.x)

    def generate_series(self, n):
        """
        Generate and return a trajectory
        of length n, starting at the
        current state.
        """
        trajectory = []
        for i in range(n):
            trajectory.append(self.x)
            self.update()
        return trajectory
```

```
>>> q = QuadMap(0.2)
>>> q.x
0.20000000000000001
>>> q.update()
# Equivalent to QuadMap.update(q)
>>> q.x
0.64000000000000012
>>> q.generate_series(5)
# The *second* parameter (i.e., n)
# in the method definition is bound to 5
[0.64000000000000012,
0.92159999999999986,
0.28901376000000045,
0.82193922612265036,
0.58542053873419597]
>>> q.x = 0.4  # Reset the state to 0.4
>>> q.generate_series(5)
[0.40000000000000002,
0.95999999999999996,
0.15360000000000013,
0.52002816000000029,
0.9983954912280576]
```

Παράδειγμα: Τιμές Παραγώγων Πολυωνύμων

```
## Author: John Stachurski

class Polynomial:

    def __init__(self, coefficients):
        """
Creates an instance of the Polynomial
class representing p(x) = a_0 x^0 + ...
+ a_N x^N, where a_i = coefficients[i].
        """
        self.coefficients = coefficients

    def evaluate(self, x):
        y = 0
        for i, a in
            enumerate(self.coefficients):
            y += a * x**i
        return y

    def differentiate(self):
        new_coefficients = []
        for i, a in
            enumerate(self.coefficients):
            new_coefficients.append(i*a)
        # Remove the first element,
        # which is zero
        del new_coefficients[0]
        # And reset coefficients data to
        # new values
        self.coefficients =
                    new_coefficients
```

```
>>> data = [2, 1, 3]
>>> p = Polynomial(data)
# create instance of Polynomial
>>> p.evaluate(1)
6
>>> p.coefficients
[2, 1, 3]
>>> p.differentiate()
# Modifies coefficients of p
>>> p.coefficients
[1, 6]
>>> p.evaluate(1)
7
```