IntroductionPython

February 21, 2017

1 Scientific Programming in Python

Created by:

- Ian Stokes-Rees [ijstokes@continuum.io]

Original github: https://github.com/ijstokes/python-sci-3h.git
Shortened by Jiri Spilka

2 Basics

In [1]:
```
"""
Exercise 1. Introduction to python
"""

import sys
import os.path
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#  ------ bool
bb = True
print bb
```

```
True
```

In [3]:
```
a = 3
b = 7
print 'Results is = ', a
```

```
Results is =  3
```

In [4]:
```
x = 10
y = 10.
```

```
In [5]: print x, y
        print type(x), type(y)

10 10.0
<type 'int'> <type 'float'>


In [6]: d = 7E4
        print d

70000.0


In [7]: # ------- complex numbers

        d2 = 3 + 2j
        print d2, type(d2)

(3+2j) <type 'complex'>
```

2.1 Tuples

```
In [8]: #  ------ tuples
        person = ('Arnost', 'Cech', 20)

        print person
        print type(person)

('Arnost', 'Cech', 20)
<type 'tuple'>


In [9]: print person[0]

Arnost


In [10]: # assigment is not supported for tuple
          # person[0] = 'petr jan'

In [11]: print 'Size of person: ', sys.getsizeof(person)

Size of person:  80
```

2.2 Dictionaries

```
In [12]: d = dict()
         d['karel'] = 50
         d['petr'] = 30
         print d

         # alternatively
         dd = {'key1': 5, 'key2': 3}
         print dd
```

```
{'karel': 50, 'petr': 30}
{'key2': 3, 'key1': 5}
```

2.3 Lists

```
In [13]: l = list([10, 20, 50, 100])
         l.append(50)
         l.pop()
```

```
Out[13]: 50
```

```
In [14]: print 'Count {0}'.format(len(l))
```

```
Count 4
```

```
In [15]: # list slicing, striding
         # this is list not an array
         nums = [3, 7, 2, 8, 5, 12, -5, 4]
         # len(nums)
         nums[7]
         nums[-1]
         nums[-2]
         nums[0]
         nums[3:6]  # half open interval: end index is NOT included
         nums[5]
         nums[1:7:2]
         nums
         nums[7:0:-2]
```

```
Out[15]: [4, 12, 8, 7]
```

```
In [16]: # power operations with power operator
         a
```

```
Out[16]: 3
```

```
In [17]: b
```

```
Out[17]: 7

In [18]: b**a

Out[18]: 343

In [19]: nums

Out[19]: [3, 7, 2, 8, 5, 12, -5, 4]

In [20]: pow(b, a)

Out[20]: 343

In [21]: sum(nums)

Out[21]: 36

In [22]: max(nums)

Out[22]: 12

In [23]: min(nums)

Out[23]: -5

In [24]: range(5)

Out[24]: [0, 1, 2, 3, 4]

In [25]: range(4, 12)

Out[25]: [4, 5, 6, 7, 8, 9, 10, 11]

In [26]: # simple functions: params, defaults, return values, scoping
         def f(x):
             ' a simple polynomial function '
             return 3*x**2 + 8

In [27]: help(f)

Help on function f in module __main__:

f(x)
    a simple polynomial function


In [28]: f(1.5)

Out[28]: 14.75
```

```
In [29]: f(3.7)
```

```
Out[29]: 49.07000000000001
```

```
In [30]: def f_o(x, offset=8):
             ''' a simple polynomial function with a configurable offset
                 offset default's to 8
             '''
             return 3*x**2 + offset
```

```
In [31]: f_o(1.5)
```

```
Out[31]: 14.75
```

```
In [32]: f_o(1.5, offset=10)
```

```
Out[32]: 16.75
```

```
In [33]: f_o(1.5, 10)
```

```
Out[33]: 16.75
```

```
In [34]: pfunkce = f_o
```

```
In [35]: print pfunkce(20)
```

```
1208
```

```
In [36]: # PEP 731 - use of lambda expressions is not recommended
         # g = lambda x: 7*x**3 + 2
         # print g(2)

         def g(x): return 7*x**3 + 2
         print g(2)
```

```
58
```

```
In [37]: with open('bostonarea.dat') as fh:
             for line in fh:
                 parts = line.split()  # will remove leading and trailing whitespace
                 print "found parts:", parts
```

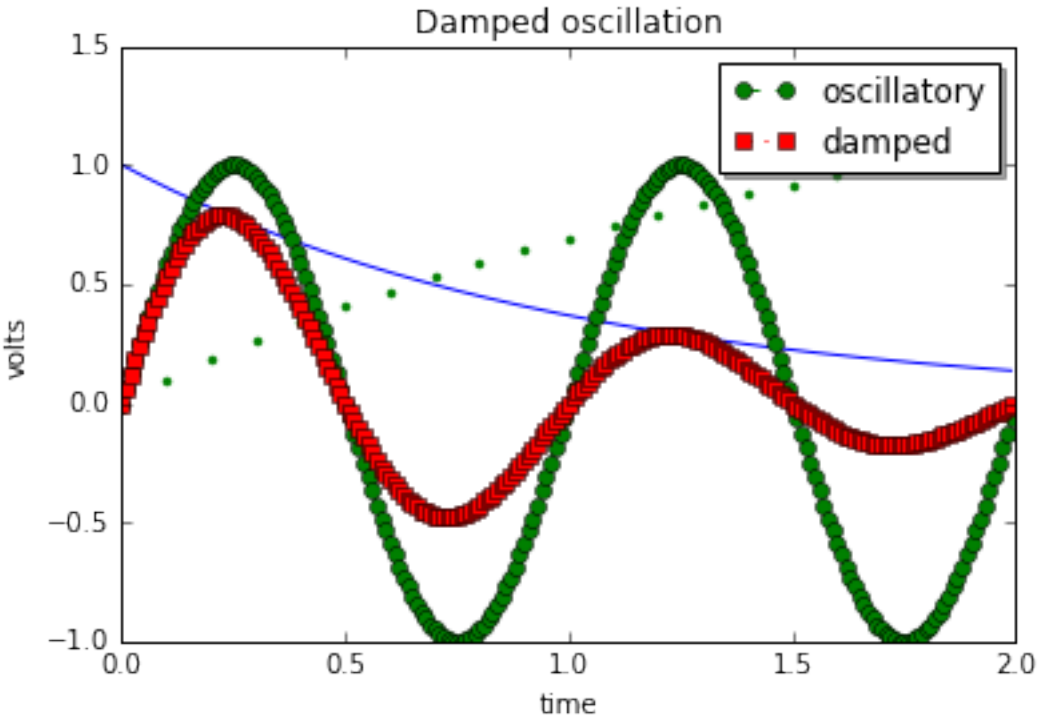
```
found parts: ['0', '0', 'Cambridge', '110000']
found parts: ['4', '-2', 'Boston', '650000']
found parts: ['2', '2', 'Somerville', '80000']
found parts: ['0', '-4', 'Brookline', '60000']
found parts: ['-4', '-2', 'Newton', '90000']
found parts: ['-4', '2', 'Waltham', '60000']
found parts: ['1', '4', 'Medford', '60000']
```

In [49]:
```
# ----- matplotlib a numpy

plt.hold()
t1 = np.arange(0.0, 2.0, 0.1)
t2 = np.arange(0.0, 2.0, 0.01)
```

In [50]:
```
# note that plot returns a list of lines.  The "l1, = plot" usage
# extracts the first element of the list into l1 using tuple
# unpacking.  So l1 is a Line2D instance, not a sequence of lines
l1, = plt.plot(t2, np.exp(-t2))
l2, l3 = plt.plot(t2, np.sin(2 * np.pi * t2), '--go', t1, np.log(1 + t1), '.')
l4, = plt.plot(t2, np.exp(-t2) * np.sin(2 * np.pi * t2), 'rs-.')

plt.legend( (l2, l4), ('oscillatory', 'damped'), loc='upper right', shadow=True)
plt.xlabel('time')
plt.ylabel('volts')
plt.title('Damped oscillation')
plt.show()
```



In [48]:
```
x = np.arange(-2*np.pi, 2*np.pi, 0.01)
type(x)
```

Out[48]:
```
numpy.ndarray
```

```
In [41]: x.shape
```

```
Out[41]: (1257,)
```

```
In [42]: x.size
```

```
Out[42]: 1257
```

```
In [43]: x.dtype
```

```
Out[43]: dtype('float64')
```

```
In [44]: nums = np.arange(3, 12, 2)
         nums.dtype
```

```
Out[44]: dtype('int64')
```

```
In [46]: # random noise
         An = 0.1
         noise = An * np.random.randn(x.size)
         y = np.sin(x)

         OFFSET = 0.001
         signal = y + noise + OFFSET

         plt.figure()
         plt.plot(x, signal, '.')
         plt.show()
```