Introduction to Python

Tutorial 1

Yang

What is python?

- Python is a programming language that lets you work more quickly and integrate your systems more effectively. [1]

You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs.

Let's get started……

[1] https://www.python.org/

Outlines

- Why Python?
- How to start?
- Python Syntax
- Data Structures
- Input and Output
- Matplot

Why Python?

- Python is easy to use, powerful, and versatile, making it a great choice for beginners and experts alike:
  - object-oriented
  - Free
  - Readability counts
  - Program portability
  - Powerful: dynamic, built-in object types, built-in tools, libraries, third-party software, AMM, large program support
  - Component integration

Library

- standard library: OS, SYS, fileinput, time, random, re, etc.
- homegrown libraries and third-party application:

- For scientific computing: >>> import somelibrary

  - Numpy: provide high-performance vector, matrix and higher-dimensional data structures for Python
  - SciPy: based on the low-level Numpy framework and provides a large number of higher-level scientific algorithms
  - matplotlib: an excellent 2D and 3D graphics library for generating scientific figures

Recommend to install Anaconda--aims to simplify package management and deployment

How to start?

- Open the command window:

  Win: CMD; Linux and macOS: Terminal

- Command:

  Windows:

  dir -> Listing the files;

  cd directory -> Moving into a directory;

  Linux and MacOS:

  ls -> Listing the files;

  pwd ->print our the current path

  cd /<absolute path> or cd ~/<relative path> -> Moving into a directory

  cd .. -> up to the parent directory

man command-name -> displays manual explanations for terminal commands

Environment setting & Jupyter

- Managing conda: *conda –version* ->displays the number of the version
  pip/conda list ->check the installed packages (on Python 3)
  conda update conda/--all -> update anaconda or all the packages
- Environment setting:
  conda info -e -> check the environment information
- To install python 2 and set up its environment
  conda create –n py27 python=2.7 anaconda
- To switch between the different environment
  Win: *activate py27* Linux and macOS: *source activate py27*
- To quit from this environment
  Win: *deactivate* Linux and macOS: *source deactivate*
- Jupyter Notebook is an open-source web application
  jupyter notebook

https://conda.io/docs/user-guide/getting-started.html

Assignment (Exercise 1)

```
# this is a symbol for the beginning of a comment
a=2    #integer
b=3.23   # float
print (a, b)
s='What is your name?'
s_1=”What is ¥
your name?”
print (s); print (s_1)
print ('%d' % a)    # this is a statement
print ('%.2f' % b)
s_2=r”What is your name ¥n ?”  # raw string literal
print (s_2)
```

#Pls try to move the 'r', what will happen?

Assignment Magic:

```
>>>x, y, z=1, 2, 3
>>>x, y=y, x
print x, y, z
2 1 3
>>>x=y=z=1
```

Not need declarations

Operators and Expressions(Exercise 2)

```
>>>a+b    #plus
5.23
>>>a*b    #multiply
6.46
>>>3**2   # the same with pow(3,2)
9
>>>b/a    #divide
1.615
>>>b//a   #Floor Division
1
>>>(a==b)==1   #equal to
False
>>>name=input('What is your name? ')
```

Most of the basic operators are the same with C/C++

```
#!/usr/bin/python
#Filename: expression.py
a=2
b=3.23
print ('a multiply b equals to ', a*b)
```

Output:

```
a multiply b equals to 6.46
```

Control Flow (Exercise 3)

Blocks: The Joy of Indentation

the preferable style is to use four spaces or Tab

indented by *the* ***same amount***

The example of the if Statement

```
num=23
guess=input('Enter a number: ')
if guess==num:
    print (' Congratulations, you guessed it.')   # this is another block
elif guess<num:
    #this is the beginning of a new block
    print ('No, it is a little higher than that')
    print ('you have to guess again!')
    #this is the end of the new block
else:
    print ('No, it is a little lower than that')
print ('Done')
```

A simple example:

```
if True:
    print ('Yes, it is true')
```

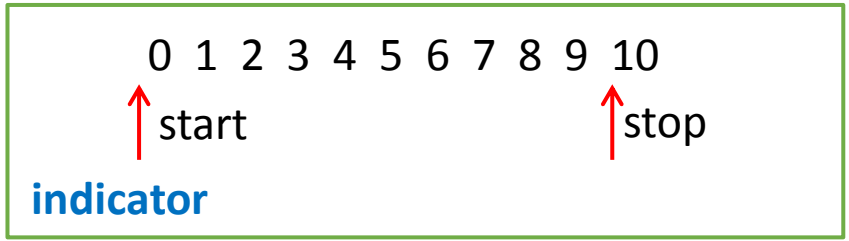
The while Statement (Exercise 4)

```
num=23
run =True
While run:
    guess=input('Enter a number: ')
    if guess==num:
        print (' Congratulations, you guessed it.')
        run=False # this causes the while loop to stop
    elif guess<num:
        print ('No, it is a little higher than that')
    else:
        print ('No, it is a little lower than that')
else: print ('The while loop is over.')
# you can do anything else you want to do here
print ('Done')
```

or you can use *break* here

The for Statement (Exercise 5)

```
>>>words = ['this', 'is', 'an', 'ex', 'parrot']
>>>for word in words:
        print (word)
>>>numbers=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>for number in numbers:
        print (number)
>>>for i in range(0, 10):
        print (i)
>>>[x*x for x in range(10) if x%3==0]
[0, 9, 36, 81]
```



Parallel Iteration

```
names = ['anne', 'beth', 'george', 'damon']
ages = [12, 45, 32, 102]
for i in range(len(names)) :
    print (names[i], 'is', age[i], 'years old')
else: print ('Did not find anyone!')
```

Functions (Exercise 6)

Example (save as func_local.py):

```
x = 50

# function definition
def func(x):
    print ('x is', x)
    x = 2
    print ('Changed local x to', x)

func(x)
print ('x is still', x)
```

Example (save as func_return.py):

```
def maximum(x, y):
    '''Prints the maximum of two numbers,
    if two numbers are not equal.'''
    if x>y:
        return x
    elif x==y:
        return 'The numbers are equal'
    else:
        return y

print (maximum(2, 9))
print (maximum.__doc__)
```

Modules

```
import module_using_name
import numpy    #example

from module_using_name import some functions
# for example
from math import sqrt

from math import sqrt as sq

from mymodule import say_hi, __version__   # you can define your own modules
```

Data Structures

List:

```
>>>[0, 2, 4, 6, 8, 10]
```

Tuple:

```
>>>tuple([1, 2, 3, 4, 5])
```

Dictionary:

```
>>>ab={ 'Swaroop'    : 'swaroop@swaroopch.com',
            'Larry'        : 'larry@wall.org',
            'Matsumoto' : 'matz@ruby-lang.org',
            'Spammer'   : 'spammer@hotmail.com'
            }
      print “Swaroop's address is “, ab['Swaroop']
```

Other Data Structures (Exercise 7)

```
import numpy as np
import pandas as pd
_1dlist=[1, 2, 3]
_2dlist=[[1, 2, 3], [4, 5, 6]]

#numpy
_1darray=np.array(_1dlist)
_2darray=np.array(_2dlist)

#pandas
_series=pd.Series(_1dlist, name='n')
_dataframe=pd.DataFrame(_2darray,
columns=list('xyz'), index=['a', 'b'] )
```

```
>>>_1dlist
[1, 2, 3]
>>>_2dlist[1][0]
4

>>>_1darray
array{[1, 2, 3]}
>>>_2darray[:, -1]
array{[3, 6]}

>>>_series

>>>_dataframe
```

```
   n
0  1
1  2
2  3
Name: n, dtype: int64
```

	x	y	z
0	1	2	3
1	4	5	6

	x	y	z
a	1	2	3
b	4	5	6

Input and Output (Exercise 8)

```
>>>f=open(r'somefile.txt', 'r+')
>>>f.readline()
>>>f.write('Hello, world!')
>>>f.close()
```

```
>>>with open(r'somefile.txt', 'r+') as f:
        s=f.read()
        print s
>>>dataList=[]
>>>dataLine=s.strip().split(' ')
>>>dataList=[float (data) for data in dataLine]
```

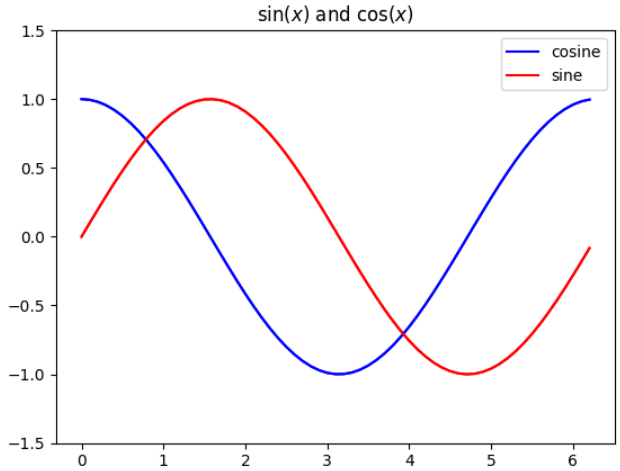
```
import numpy
>>>data = np.genfromtxt('foo.txt', delimiter=',')
>>>data=np.savetxt('foo.txt', fmt='%2.3f', delimiter=',')
```

```
import pandas
>>>data=pd.read_csv('somefile.csv')
>>>data.to_csv('somefile.csv', index=False)
```

Matplotlib —plot (Exercise 9)

Pyplot in Matplotlib Provides a MATLAB-like plotting framework.

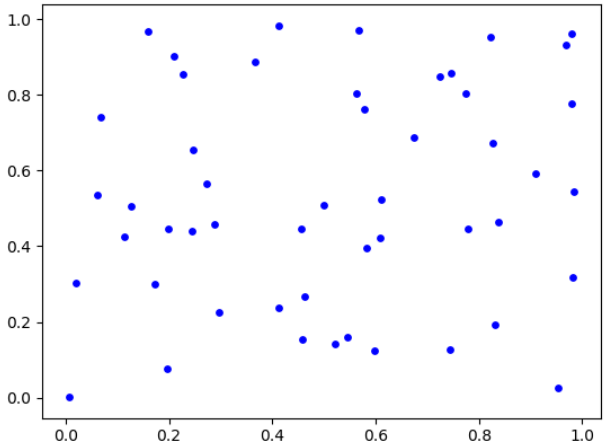
```
>>>import matplotlib.pyplot as plt
>>>x=np.arange(0,5,0.1)
>>>y, y1=np.sin(x), np.cos(x)
>>>plt.plot(x, y, c='r', label='sine')
>>>plt.plot(x, y1,  c='b', label='cosine')
>>>plt.title(r'$¥sin(x)$ and $¥cos(x)$')
>>>plt.legend()
>>>plt.ylim(-1.5, 1.5)
>>>plt.show()
>>>path=os.getcwd()
>>>plt.savefig(path+'¥¥fig1.png')
```



Matplotlib —scatter (Exercise 10)

Make a scatter plot of x vs y

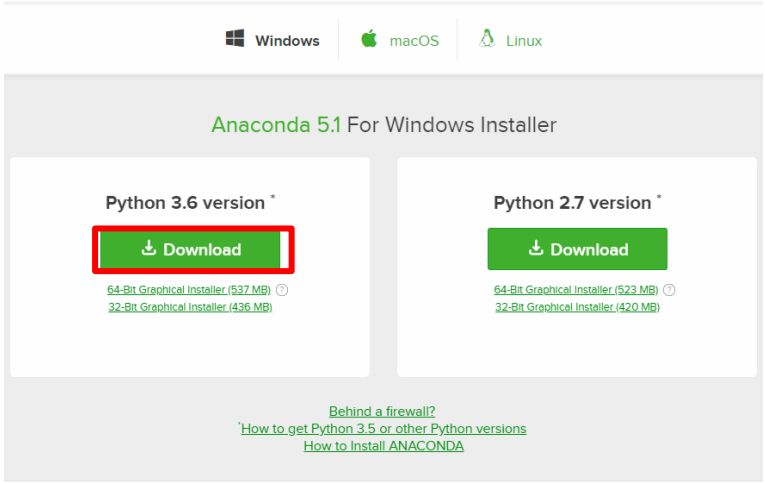
```
>>>import matplotlib.pyplot as plt
>>>x=np.random.rand(50) # 50 Random values in a given shape
>>>y=np.random.rand(50) #random samples from a uniform distribution over [0, 1)
>>>plt.scatter(x, y, s=15, c='blue', marker='o')
```

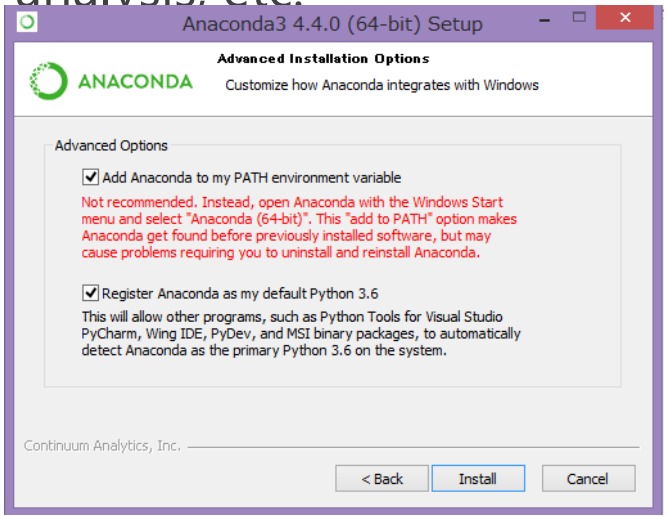


Reference

Anaconda(Win)

- Anaconda: a scientific computing environment of Python for installing and managing a lot of packages including science, Mathematic, engineering and data analysis, etc.





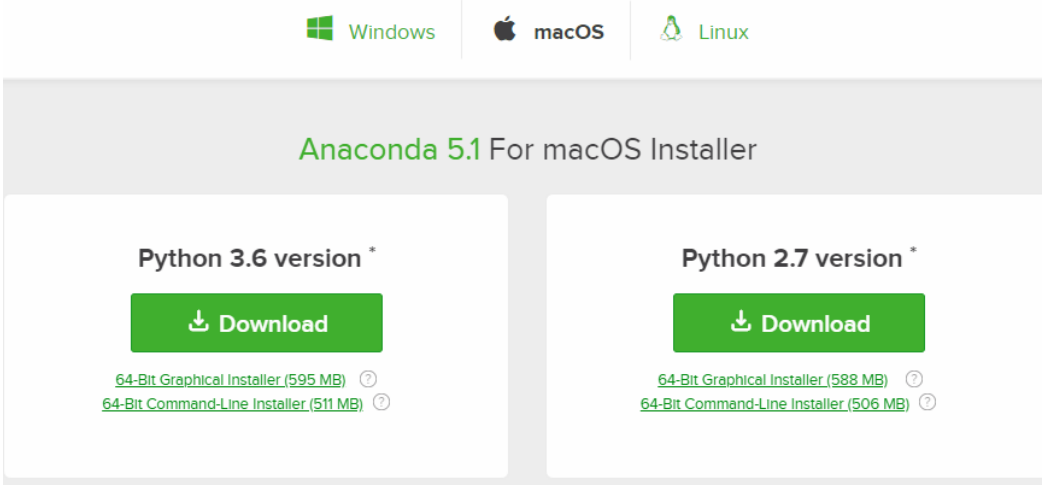
- IDE(Integrated Development Environment) for Python: an application to facilitate application development

for example: Vim, Eclipse with PyDev, Sublime Text, PyCharm, Emacs, etc.

Anaconda(macOS)

- Anaconda for macOS has the graphical installer (“wizard”) and the command line installer (“manual”)

  https://docs.anaconda.com/anaconda/install/mac-os



- For conmmand-line installer: *bash ~/Downloads/Anaconda3-5.1.0-MacOSX-x86_64.sh*
- The installer prompts “Do you wish the installer to prepend the Anaconda install location to PATH in your /home/<user>/.bash_profile ?” recommend “yes”.

Pycharm

- IDE(Integrated Development Environment) for Python: an application to facilitate application development

  e.g. : Vim, Eclipse with PyDev, Sublime Text, PyCharm, Emacs, etc.
- Pycharm: download from https://www.jetbrains.com/pycharm/.