Upsilon Lab *Introduction to Python for Mac*

an Official UCLA Physics & Astronomy Department Sponsored Organization

Tony Alarcon, Manager-TS

February 25, 2018

“A lab for undergraduates, by undergraduates.”

1 Summary

Python is a high-level scripting language that is fully object-oriented, utilized for general programming purposes. It contains simple to use syntax and the code is dynamically interpreted, making it very flexible. Python provides a variety of function tools used for basic scientific research and engineering, which is supported by comprehensive, open-source, contributions from scientists for their own work. There are currently two major versions of Python available: Python 2 and Python 3. To learn which one best fits your needs, you may refer to this page.

2 Installing Python

You may already have the programming language since versions of Mac OS X 10.8 or later features Python 2.7 pre-installed in your system. You may quickly check by opening Terminal located in the *'Utilities'* folder in *'Applications'* , and type the following:

```
python -V
```

Note that the capitalization in V is important in this step. The command will return the version of Python installed in your system. The Apple version of Python is installed in System/Library/Frameworks/Python.framework and /usr/bin/python, respectively; **these should never be modified**. If you've never downloaded Python before, I highly encourage you to do so, as it will provide you with a framework which includes executables and libraries. You may download it here.

If everything went well, we can now check to see if some useful modules are installed. In particular, lets check if we have numpy, scipy, and matplotlib. We will open python from terminal by simply typing python in the command line, as follows:

```
python
```

You will be prompted with an introduction of the version being used along with some information. Now, to check if you have the modules, we will use the inbuilt help() function:

```
help('numpy')
```

If you have numpy installed, you will be prompted with documentation about the module, you may hit the key Q to interrupt and go back to the python interface. Do the same for all three modules. If you are missing one or more modules, you can install them by downloading the content from respitory. To exit the python interface hit Ctrl-Z and you can install numpy simply by typing

```
python -m pip install --user numpy
```

This installs packages for your local user, and does not write to the system directories. Do the same for the rest of the modules if you do not have them installed.

Now, to test the modules. The easiest and fastest way is to make a script on a text editor, before proceding, please refer to section 3. You can then run scripts from the Terminal window, first save the script with the .py extension in any desired folder and use Terminal to move to the location of the file you want to execute. For example, if you saved myscript.py in your Desktop in a folder named python, then to move locations, you would execute:

```
cd ~/Desktop/python
```

To run your script, execute:

```
python myscript.py
```

2.1 Matplolib

The matplotlib library produces publication-quality figures such as scatterplots, histograms, power spectra, power chart, etc., that allows for effortless customizable properties. There is also a procedural pylab that provides a MATLAB-like interface, giving you full control of axes properties, text annotation, line styles, and more. Let's plot a mathematical function to test matplotlib, type and execute the following:

```
from pylab import *

x = []
y = []

for i in range(100):
    angle = .1 * i
    sang = sin( angle )
    x.append(angle)
    y.append(sang)

plot( x , y , 'r' )
show()
```

We start by importing the functions in pylab and proceed to create two empty lists that will represent our x and y axis data points. The range() function generates a list of 100 numbers, that is passed on to the for loop where the integer 'i' runs from 0 to 99. Therefore, for every value in 'i' we multiply it by '.3', store the value in a temporary variable named 'angle' and added to our list 'x' by using the function 'append()'. In addition, the sine of each angle is computed and stored in a temporary variable named 'sang' and similarly added to our list 'y'. It is important to note that the for loop runs only for commands that are indented. The 'plot()' commands takes the x and y coordinates as their first and second arguments, respectively. The last statement 'show()' instructs matplotlib to display the figures.

2.2 Numpy

The program above is fine in the sense that it introduces a variety of functions and syntax; however, since our objective is to use python as a tool for learning physics, we need data types like arrays and matrices, such as those provided by the Numpy library. Let's plot the same sine function as above, along with some lissagous figures, but this time utilizing the power of Numpy. To test numpy, type and execute the following:

```
from numpy import *
from pylab import *

x = linspace(0.0, 2*pi, 100)

plot(x, sin(x), 'r')                # 'r' stands for red
plot(sin(x), cos(x), 'y')           # 'y' stands for yellow
plot(sin(2 * x), cos(x), 'b')       # 'b' stand for blue. The default setting.

show()
```

The first statement imports all of the functions from numpy. The 'linspace()' function generates an array of 100 equi-spaced values, starting from 0 and ending with 2π. It is also possible to specify the color of the plot by adding a third argument inside the 'plot()' function. Note that anything followed after a hashtag, '#' is ignored by the compiler, we use the hashtag to make comments inside our program. In this program we managed to plot three figures, meanwhile eliminating the need of a for loop; essentially reducing the complexity of the program, allowing us to concentrate on physics.

2.3 Scipy

As an extension to the scientific tools provided by numpy, the Scipy library offers a collection of mathematical algorithms and convenience functions that allows a high-level command for data manipulation and visualization that rivals systems such as MATLAB, IDL, Octave and Scilab [1]. As a brief example, we will employ the most commonly used special function: the gamma function, $\Gamma(z)$, defined via a convergent improper integral, for complex numbers with a positive real part:

$$\Gamma(z) = \int_0^\infty s^{x-1} e^{-x} dx$$

Type and execute the following, to test scipy:

```
from scipy.special import gamma
print gamma(.05)
```

3 Text Editor

Before you start writting code in python, you need to choose a text editor. IDLE (Integrated Development and Learning Environment) is the default text editor bundled with Python. It offers multi-window text editing with syntax highlights, smart indent and autocompletion. No download is necessary since it comes with Python.

However, if you have multiple versions of Python in your system, you often want to have control on which version you want your script to run. An easy method is to use the python interpreter from Terminal. To do that you need another text editor. The one for my personal use is BBEdit, I especially like the FTP/SFTP support feature that allows you open/save/transfer scripts from remote locations and is useful in astrophysics applications. You may download it here. Other popular choices are Sublime Text and Aquamacs. Nonetheless, you may choose whichever text editor you like best.

4 Python Basics

The following are sample codes designed to develop a sense of intuition with basic functions and statements in Python. Let's begin with a classical introductory program, namely the Hello World program.

```
print 'Hello World'
```

In Python 2.7, print is a statement that instructs python to display the text enclosed in single quotes, ' ', these data types are called *strings*.

Besides strings, other data types include *integers*, *booleans*, *floats* and many more.

```
# integer
some_whole_number = 1

# float
some_decimal = .5

# boolean
true_boolean = True
false_boolean = False
```

As you may have noticed by now, we use the assignment operator =, to assign any data type value on the right, to a chosen variable on the left side of the operand.

Now, lets try something more elaborate:

```
if 10 > 5
  print '10 is greater than 5'

if True:
  print 'Hello World'
```

The if statement is a control flow tool that evaluates if an expression is either True or False. If proven to be true, the **indented** block will be executed. If false, Python will ignore that block and move on. Another example of a conditional statement is the for loop statement, which is used when you want to iterate some block of code over a given sequence. For example:

```
# Prints out the numbers 0,1,2,3,4
for x in range(5):
    print(x)
```

The range(N) function creates a list, and is considered shorthand for 0, 1, 2, ..., N-1.

Now that you have familiarized yourself with some of Python commands, try to guess the output of the following programs[1] for further practice.

Code_1.py

```
a = 1
b = 1.2
c = 2 + 3j
d = 'i am a string'
e = [a, b, c, 'hello']
print d, b * c, e
```

Code_2.py

```
s = 'hello world'
m = [1, 2.3, 33, 'ha']

```

[1]Ajith (2008) Python in Physics Education (version 1.0) . http://www.iuac.res.in/ elab/phoenix/docs/Physics/pyphy.pdf.

```python
for k in s:
  print k

for k in m:
  print m
```

Code_3.py

```python
x = range(10)
print x

for i in x:
  if i > 8:
    print i
```

If you feel ready, attempt these exercises below. The problems are in order of increasing difficulty.

1. Write an program that determines the area of a rectangle, given the width and length as their arguments.
2. Write a program that takes a string and converts all characters into either upper or lower case. For example, if the string is CYCLOTRON then the output would be cyclotron, and vise-versa.
3. Write a program that prints out a comma seperated sequence of the numbers in between 10 and 100 which are a multiple of 5 but not a multiple of 50 or 25. The output should be given in a single line.