Introduction to Python Tutorial and How to Make Python Scripts

Basic programming Jargon

Terminal: Is a text only window in a graphical user interface (GUI) that emulates a console.It is a text input/output environment, which implements various commands and outputs the results.

Shell: It is a program with text only interface for Linux and other Unix like operating systems

Command line: The space to the right of the command prompt is called the Command line. From the command line users can enter commands.

Basic Unix Commands:
In order to use terminal and access your python programs you will need to know some basic Unix Commands. These can easily be found online but for your convenience Ill refer you to this link:
https://www.tjhsst.edu/ dhyatt/superap/unixcmd.html

Why use Python?

Python is free for everybody! (unlike MATLAB and IDL)

Python is widely used and is the number one tool used in Astronomy

There are lots of programs and Libraries written for python which are called packages

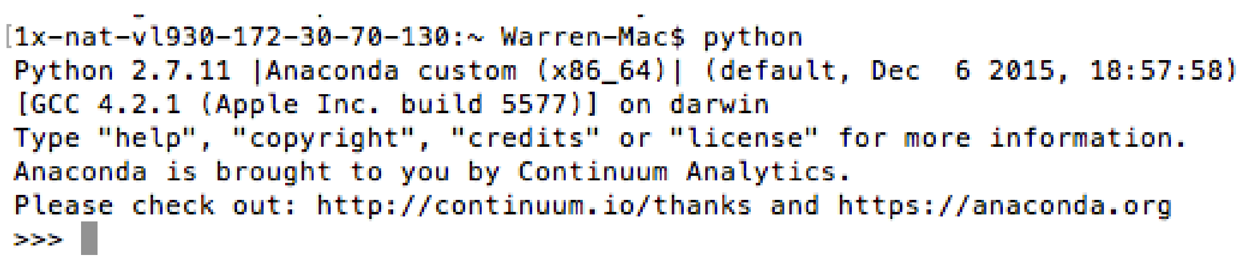
Opening Terminal and Python on a Mac

Step 1: Open finder which is available in the Dock

Step 2: Click on Applications and chose Utilities.

Step 3: Double click on Terminal (Terminal should then appear)

Once youve opened the terminal utility once, you can alternatively simply search in the spotlight finder (upper right hand corner magnifying glass) for terminal to skip a few steps. Once terminal is open, you can open python by simply typing python in terminals command line. Once this is done you're terminal shell should look something like this:



```
1x-nat-vl930-172-30-70-130:~ Warren-Mac$ python
Python 2.7.11 |Anaconda custom (x86_64)| (default, Dec  6 2015, 18:57:58)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> 
```

You have now accessed python!

Python has different versions

There are two major versions of python: Version 2.7 and Version 3.x (You choose any version you want but please note that the latest version of python uses slightly different syntax than the previous versions. Also note that astronomy software may not work on Version 3; for the time being it is advisable to use Version 2.7)

On Macs, a basic version of Python is already installed. Basic Python doesnt include many software packages, so we have to install different python packages and then import them into python to enhance its capabilities. For the type of programming you will be doing in Astronomy, I recommend downloading Astroconda which is a free python package which has many important packages like numpy for example. To download Astroconda go to:

http://astroconda.readthedocs.io/en/latest/installation.html and follow the directions.

Python Variable and Data Structure

Integer (called int)

Real (called float for floating point)

Complex (called complex)

Logical (called bool for boolean)

String an array of characters

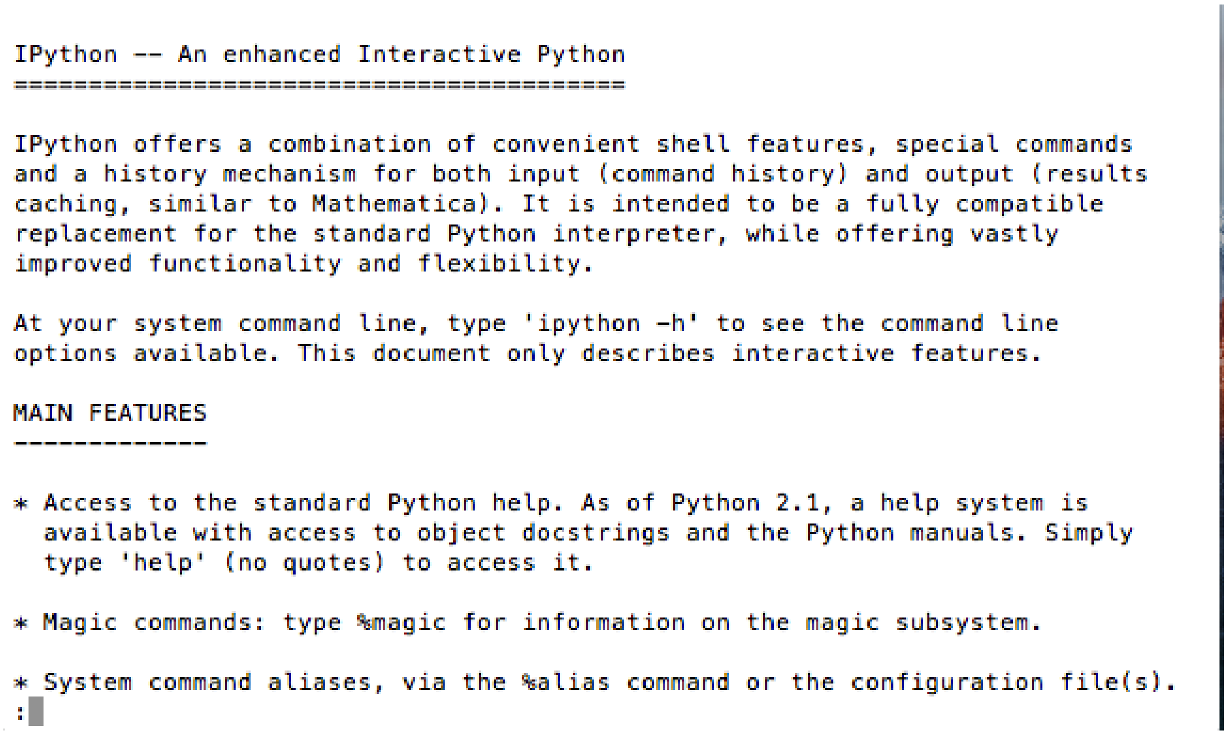
List an array of variables (of any kind)

Tuples a kind of list that is immutable

Dictionaries a list of key and value pairs.

What Is IPython?

IPython is an interactive shell built within python. Ipython can do a lot more than the standard python prompt and does take the standard python commands.You can run a python program in IPython by simply typing IPython and the filename in terminals command line. Ipython is great tool for helping you debug your script and it will tell you where the problems are in your script.

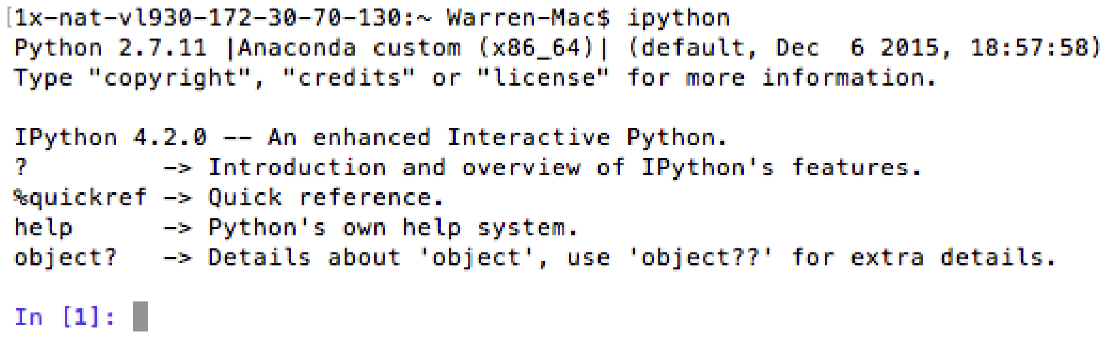


```
IPython -- An enhanced Interactive Python
=========================================

IPython offers a combination of convenient shell features, special commands
and a history mechanism for both input (command history) and output (results
caching, similar to Mathematica). It is intended to be a fully compatible
replacement for the standard Python interpreter, while offering vastly
improved functionality and flexibility.

At your system command line, type 'ipython -h' to see the command line
options available. This document only describes interactive features.

MAIN FEATURES
-------------

* Access to the standard Python help. As of Python 2.1, a help system is
  available with access to object docstrings and the Python manuals. Simply
  type 'help' (no quotes) to access it.

* Magic commands: type %magic for information on the magic subsystem.

* System command aliases, via the %alias command or the configuration file(s).
:
```

Start IPython:

Step 1: open terminal

Step 2: in the command line enter: IPython and the following image should appear



```
1x-nat-vl930-172-30-70-130:~ Warren-Mac$ ipython
Python 2.7.11 |Anaconda custom (x86_64)| (default, Dec  6 2015, 18:57:58)
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

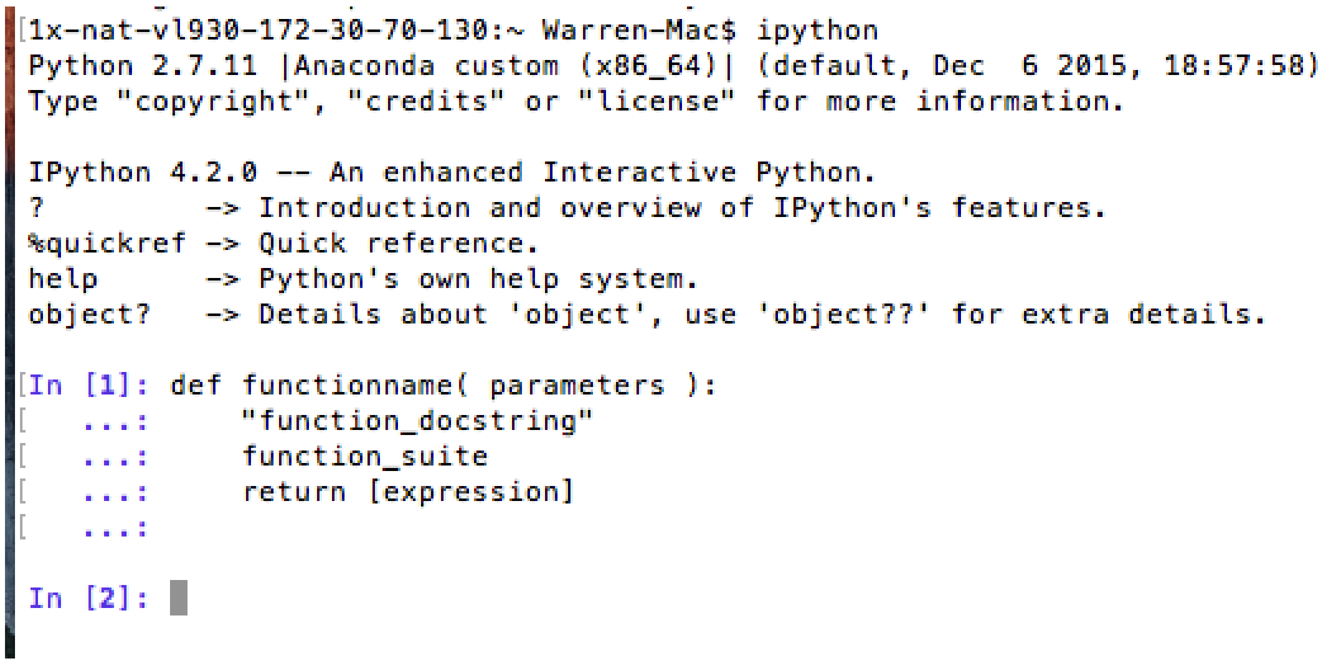
User Defined Functions

Python has many built in functions such as print, but you can make your own function which is called a user defined function. Functions are extremely useful when you have to re-use your code various times.

In python, user defined functions are made with the keyword def followed by the name you want to give your function in parenthesis. Outside the parenthesis should be colon (:).

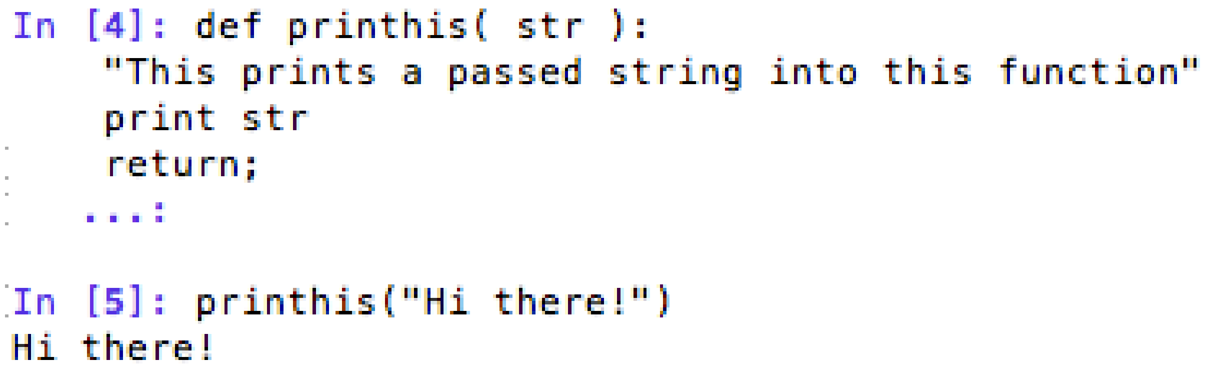
The arguments of your function should be listed in the parenthesis. Any equations you wish to add to your function should be indented and entered in the command line under your function at the end of your function enter the command return which exits the function.

Example 1:



```
1x-nat-vl930-172-30-70-130:~ Warren-Mac$ ipython
Python 2.7.11 |Anaconda custom (x86_64)| (default, Dec  6 2015, 18:57:58)
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: def functionname( parameters ):
   ...:     "function_docstring"
   ...:     function_suite
   ...:     return [expression]
   ...:

In [2]:
```

Example 2: In the Example below I created a function called printhis" and then called the function to execute it.

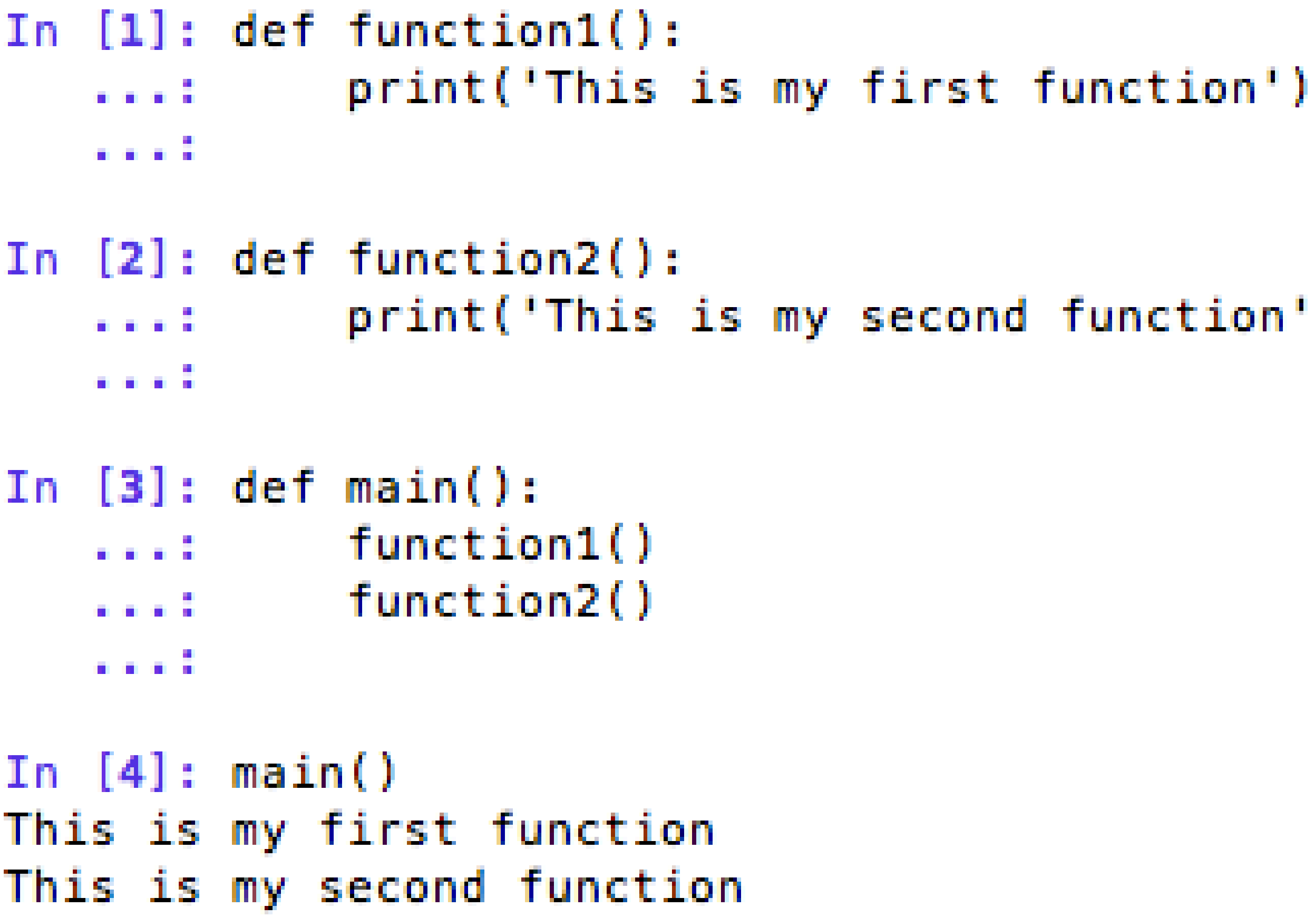


```
In [4]: def printhis( str ):
    "This prints a passed string into this function"
    print str
    return;
   ...:

In [5]: printhis("Hi there!")
Hi there!
```

Multiple functions in a script

You can have several functions in a script and in order to get them to do anything you have to call each and everyone of them. An easier way to do this is to put all your user defined functions in a main function, then you can just call the main function.

Example 2:



```
In [1]: def function1():
   ...:     print('This is my first function')
   ...:

In [2]: def function2():
   ...:     print('This is my second function')
   ...:

In [3]: def main():
   ...:     function1()
   ...:     function2()
   ...:

In [4]: main()
This is my first function
This is my second function
```

Conditional statements in general

Conditional statements allow us to check conditions and change the behavior of the program. The simplest form of a condition statement is the if statement. The if statement has form:

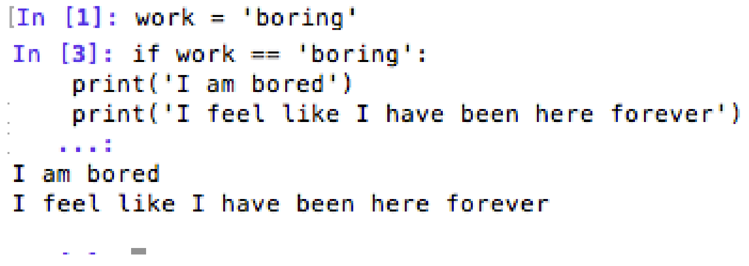
if (condition) (statement)

The condition is a **boolean expression** which is defined as a logical statement that is either true or false. Boolean expressions can compare data of any type as long as both parts of the expression have the same basic data type. You can test data to see if it is equal to, greater than, or less than other data. The ¡ statement ¿ is usually a **block**. A block collects the sequence of statements and declarations into a single statement.

Python If statement Syntax

In Python, the If statement starts with the command if followed by a key word and a colon(:) The line where your statement begins must be indented. It is customary in python to use four spaces for indenting.

Here is an example of an if statement using Ipython:



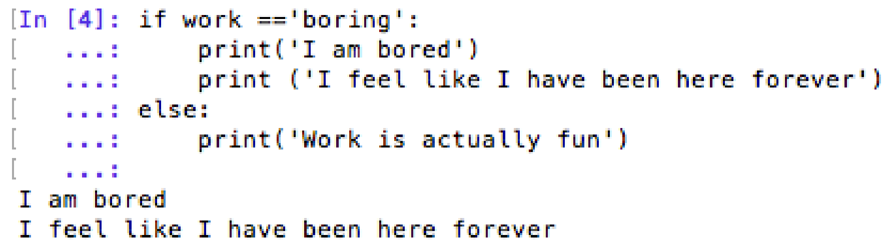
```
In [1]: work = 'boring'
In [3]: if work == 'boring':
    print('I am bored')
    print('I feel like I have been here forever')
   ...:
I am bored
I feel like I have been here forever

   ...
```

As explained earlier, the expression after the if statement is called the condition. If it is true, then all the indented statements get executed. What happens if the condition is false, and work is not equal to 'boring'? In a simple if statement like this, nothing happens, and the program continues on to the next statement.

If else statement:

The if else statement is useful when you want one action to happen when a statement is true and something different to happen if the statement is false.

Example:



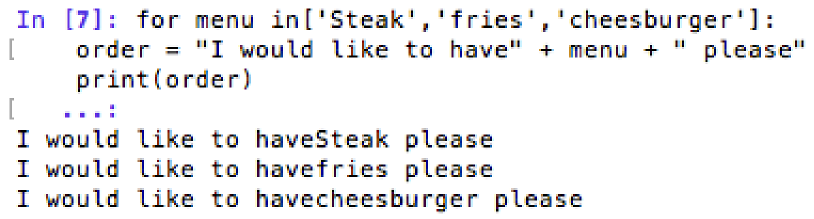
```
In [4]: if work =='boring':
   ...:     print('I am bored')
   ...:     print ('I feel like I have been here forever')
   ...: else:
   ...:     print('Work is actually fun')
   ...:
I am bored
I feel like I have been here forever
```

for loops:

A for loop is a control flow statement that defines iteration, it allows code to be carried out many times. Every programming language uses for loops (the syntax will vary with the programming language). The general form a for loop is:

for variable in sequence:
statements

Example



```
In [7]: for menu in['Steak','fries','cheesburger']:
    order = "I would like to have" + menu + " please"
    print(order)
   ...:
I would like to haveSteak please
I would like to havefries please
I would like to havecheesburger please
```

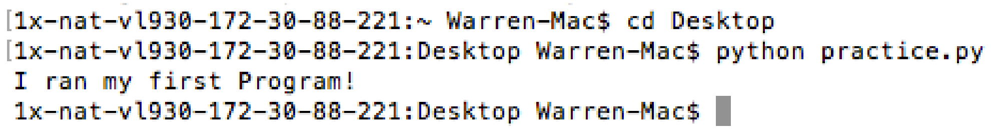
Please note: There are also while loops which I do not talk about in this tutorial. If you would like more information on while loops please refer to pythons official documentation:

How to write a program in python:

To write a program in python you first need an editor. There are various editors for python, and its completely up to your preference. Here is a link that has a list of python editors **(https://wiki.python.org/moin/PythonEditors)**. Some of the most common editors are emacs, gedit , and vim. To write a program you need to open your editor and save the file as a .py file. This will set the environment of your editor. Once this is complete you can start writing your script. if you want to run your the program, you will need to open terminal and change the directory to where your python program is saved. Once you are in the right directory, you can enter in the command line python, followed by your programs file name. This will run your program.

Example:

Using the editor macvim, I created a file called practice.py and saved it to my desktop. In my program, practice.py, I wrote the command : Print I ran my first program. To run this program I open terminal and changed the directory to where I had the filed save (Desktop). I then ran my program which output the line I ran my first program!

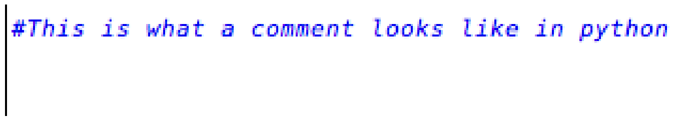


```
[1x-nat-vl930-172-30-88-221:~ Warren-Mac$ cd Desktop
[1x-nat-vl930-172-30-88-221:Desktop Warren-Mac$ python practice.py
 I ran my first Program!
 1x-nat-vl930-172-30-88-221:Desktop Warren-Mac$
```

How to Document your script

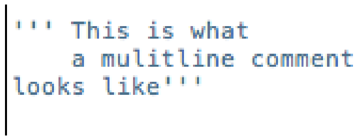
Commenting your script is extremely important! Every program you create should be commented so that it clearly describes the purpose of the code. Every line in your script should be commented describing what that line does. Another person should be able to understand your code by simply reading your comments. This will help you greatly when you have to use previously written programs.

Every programming language has a different method of commenting. In Python, you can comment out a line by using the pound symbol **#** . This indicates to Python that specific line of code should not be evaluated. Anything written after the pound symbol is commented out, and anything written before the pound symbol is not.



```
#This is what a comment looks like in python
```

The above method only comments out one line of code. If you want to comment out multiple lines use triple quotes strings:



```
''' This is what
    a mulitline comment
looks like'''
```

How to Automatically Import Packages In a Python Script

As mentioned earlier, the basic Python that comes with your Mac doesnt include many software packages, so we have to install different python packages and then import them into Python to enhance its capabilities. There are a few python packages that you will probably always use like numpy and matplotlib. You can automatically import these packages in two ways.

1) You can make a python script called common_imports.py . In this script, import the packages that you will use frequently. Now when writing ever you write a new python script, just write the line the following line to import all the packages you normally use from commontext_imports import*

2) Use the environment variable PYTHONSTARTUP which is defined from the official Documentation as:

If this is the name of a readable file, the Python commands in that file are executed before the first prompt is displayed in interactive mode. The file is executed in the same namespace where interactive commands are executed so that objects defined or imported in it can be used without qualification in the interactive session. You can also change the prompts sys.ps1 and sys.ps2 and the hook sys.__interactivehook__ in this file.

This method will automatically import the packages you want in interactive python while the first method will import the packages you want in a python script.

Python Exercise 1

Write a script in Python to evaluate the following equation to derive the array Q and plot the result. (Please do this in two ways, one of them by using numpy the other by using a loop.)

$$Q = Xc^3 + Yc^2 + Zc$$

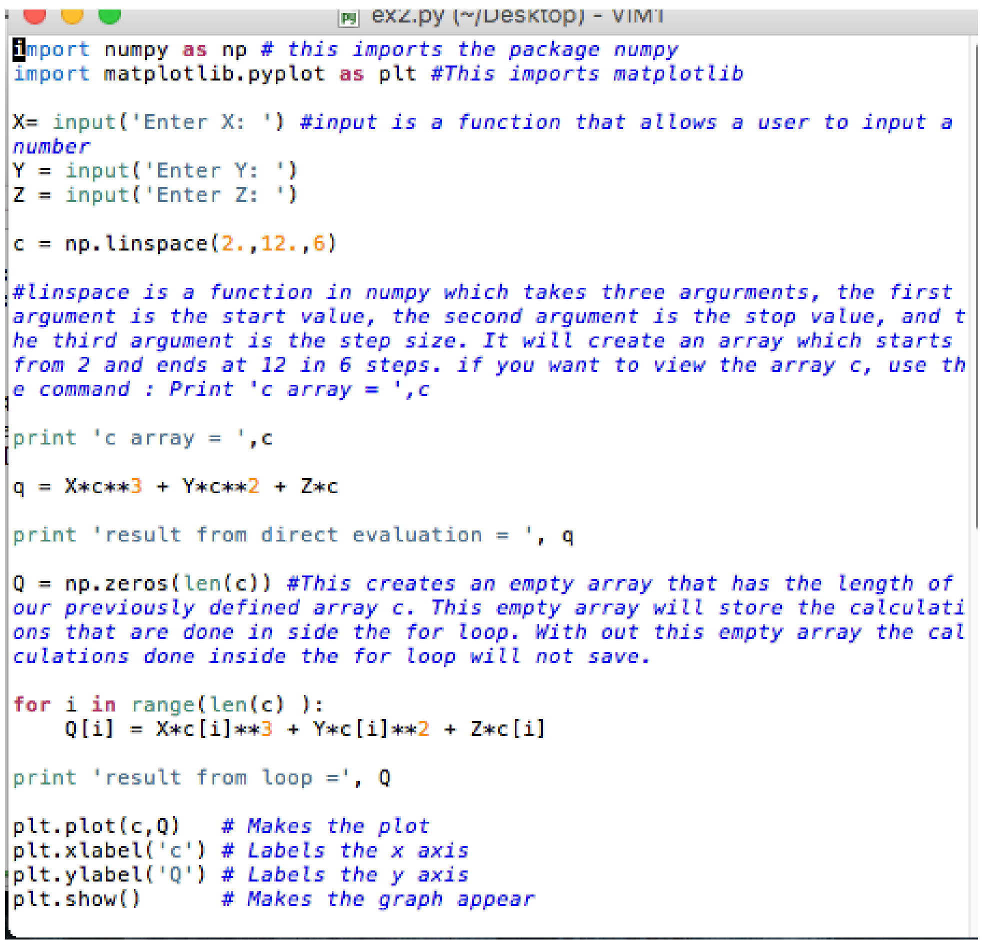
where the constants X,Y, and Z are values that the user inputs and the array c is : c = [2,4,6,8,10,12].

Hints: Use the python package linspace which you will have to import from numpy. To graph youll have to import matplotlib.pyplot. You may also use the Python Cheat sheet in the link below:

https://ipgp.github.io/scientific_python_cheat_sheet/

Solution: Starts on next page

Your script should look similar to this:



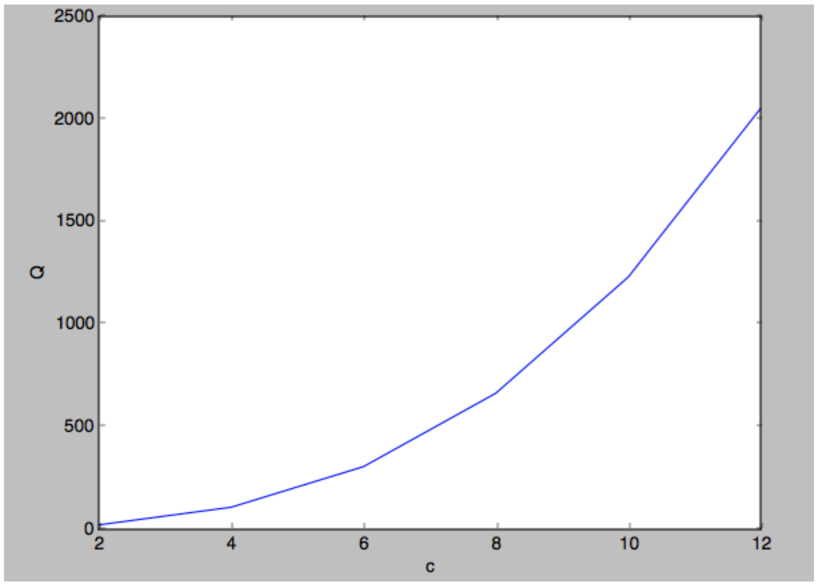
```
import numpy as np # this imports the package numpy
import matplotlib.pyplot as plt #This imports matplotlib

X= input('Enter X: ') #input is a function that allows a user to input a
number
Y = input('Enter Y: ')
Z = input('Enter Z: ')

c = np.linspace(2.,12.,6)

#linspace is a function in numpy which takes three argurments, the first
argument is the start value, the second argument is the stop value, and t
he third argument is the step size. It will create an array which starts
from 2 and ends at 12 in 6 steps. if you want to view the array c, use th
e command : Print 'c array = ',c

print 'c array = ',c

q = X*c**3 + Y*c**2 + Z*c

print 'result from direct evaluation = ', q

Q = np.zeros(len(c)) #This creates an empty array that has the length of
our previously defined array c. This empty array will store the calculati
ons that are done in side the for loop. With out this empty array the cal
culations done inside the for loop will not save.

for i in range(len(c) ):
    Q[i] = X*c[i]**3 + Y*c[i]**2 + Z*c[i]

print 'result from loop =', Q

plt.plot(c,Q)   # Makes the plot
plt.xlabel('c') # Labels the x axis
plt.ylabel('Q') # Labels the y axis
plt.show()      # Makes the graph appear
```

When you call your script in terminal it should look like this:

```
Enter X: 1
Enter Y: 2
Enter Z: 3
c array =  [  2.   4.   6.   8.  10.  12.]
result from direct evaluation =  [   22.   108.   306.   664.  1230.  2052.]
result from loop = [   22.   108.   306.   664.  1230.  2052.]
```

Solution continues on the next page

Your graph should look like this if you used the above values for X Y and Z:



2500
2000
1500
1000
500
0
Q
2
4
6
8
10
12
c