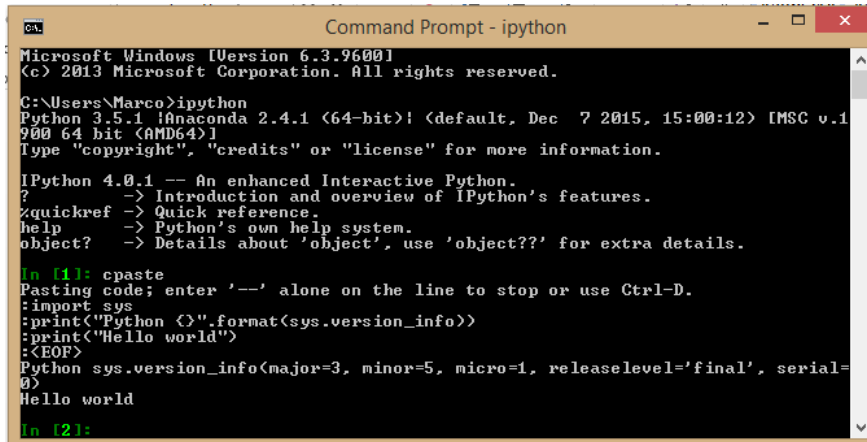


## Marco Rego - Assignment 2

### 3 ways to execute python code

#### Method 1 – Interactive mode



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

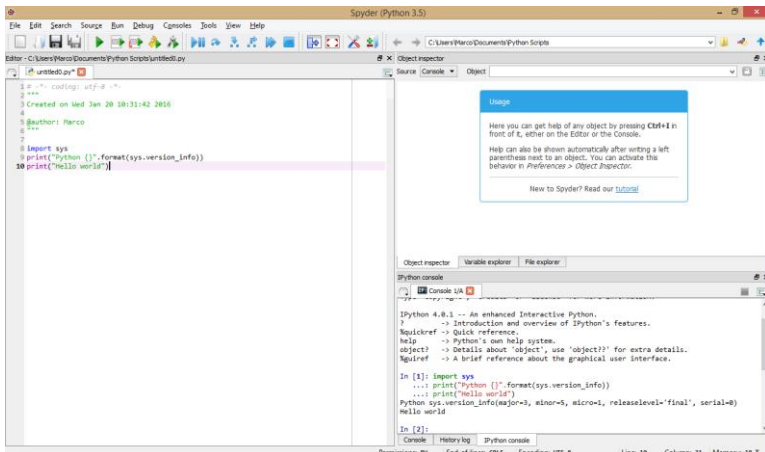
C:\Users\Marco>ipython
Python 3.5.1 |Anaconda 2.4.1 (64-bit)| (default, Dec  7 2015, 15:00:12) [MSC v.1
900 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 4.0.1 -- 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]: cpaste
Pasting code; enter '--' alone on the line to stop or use Ctrl-D.
:import sys
:print("Python {}".format(sys.version_info))
:print("Hello world")
:<EOF>
Python sys.version_info(major=3, minor=5, micro=1, releaselevel='final', serial=
0)
Hello world

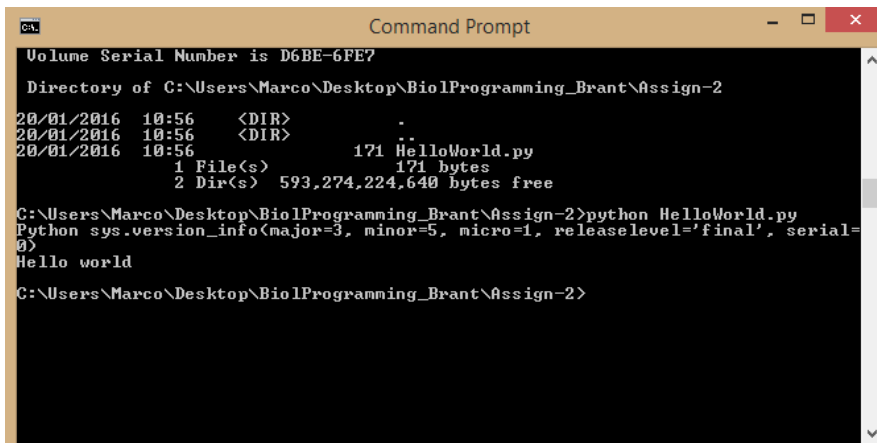
In [2]:
```

#### Method 2 – Text Interpreter (Spyder)



```
#!/usr/bin/env python
# coding: utf-8
"""
Created on Wed Jan 20 10:31:42 2016
@author: Marco
"""
import sys
print("Python {}".format(sys.version_info))
print("Hello world")
```

#### Method 3 – Running a Python script in the Terminal



```
Volume Serial Number is D6BE-6FE7

Directory of C:\Users\Marco\Desktop\BiolProgramming_Brant\Assign-2

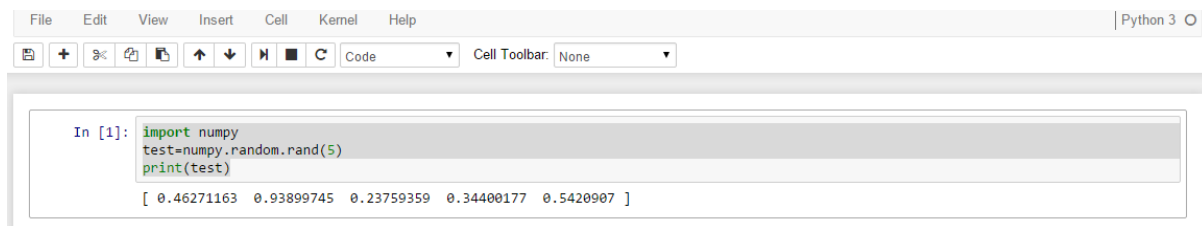
20/01/2016  10:56    <DIR>          .
20/01/2016  10:56    <DIR>          ..
20/01/2016  10:56                171 HelloWorld.py
               1 File(s)                171 bytes
               2 Dir(s)  593,274,224,640 bytes free

C:\Users\Marco\Desktop\BiolProgramming_Brant\Assign-2>python HelloWorld.py
Python sys.version_info(major=3, minor=5, micro=1, releaselevel='final', serial=
0)
Hello world

C:\Users\Marco\Desktop\BiolProgramming_Brant\Assign-2>
```

## Importing modules

When importing just “numpy”, I am creating reference to that module and if I want to execute a function from it, I will have to type “numpy.function()”.



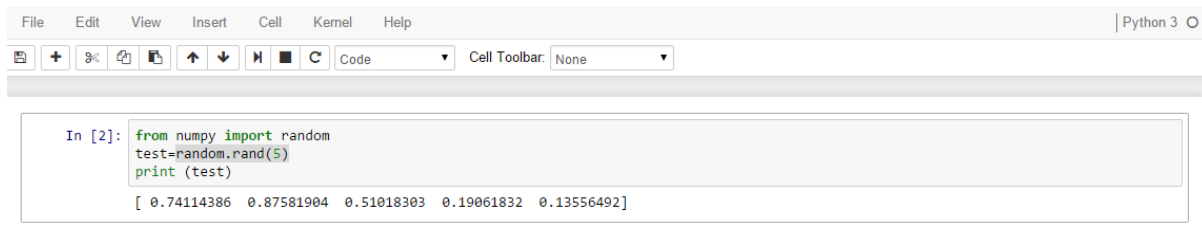
```
File Edit View Insert Cell Kernel Help Python 3
[+] [-] [undo] [redo] [run] [stop] [clear] [code] Cell Toolbar: None

In [1]: import numpy
test=numpy.random.rand(5)
print(test)

[ 0.46271163  0.93899745  0.23759359  0.34400177  0.5420907 ]
```

When using “from numpy import \*”, I will be creating name references to all functions “from numpy”. This means that I just need to write the function name to run that function. I will not need to type numpy.function, if I do that I will receive an error message.

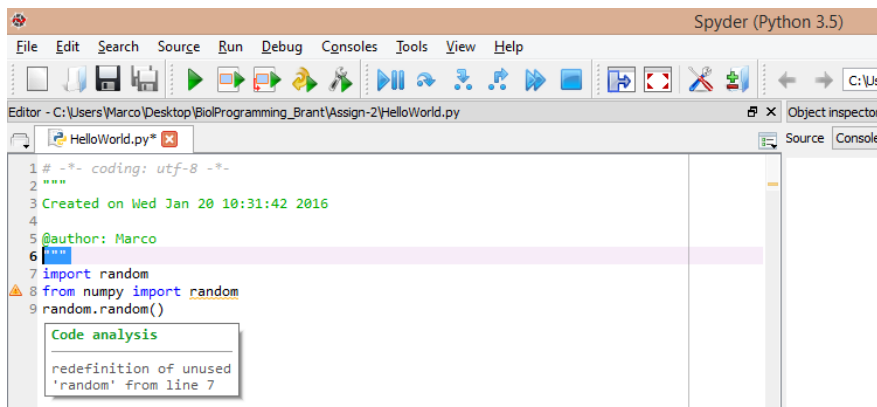
When using from numpy import random I will create a name reference just for the numpy function “random”. This will make me loose the name reference that I had for the stdlib “random” function and from now on, every time that I call the function “random” I will be using the numpy function



```
File Edit View Insert Cell Kernel Help Python 3
[+] [-] [undo] [redo] [run] [stop] [clear] [code] Cell Toolbar: None

In [2]: from numpy import random
test=random.rand(5)
print (test)

[ 0.74114386  0.87581904  0.51018303  0.19061832  0.13556492]
```



## Paths

1. C:\Users\Username\Desktop\File.txt
2. C:\Windows\File.txt
3. PATH is an environment variable that specifies directories where executable programs are located (Wikipedia - [https://en.wikipedia.org/wiki/PATH\\_\(variable\)](https://en.wikipedia.org/wiki/PATH_(variable))).
4. I was not able to locate the Pythonpath setting/variable in my computer (Windows 8.1). PYTHONPATH sets the search path for **importing** python modules (<http://stackoverflow.com/questions/7850908/what-exactly-should-be-set-in-pythonpath>)

## Modules

The csv module deals with .CSV file reading and writing. This file format is commonly used for spreadsheets and databases applications. I already worked with this module and it is a good and straightforward tool to write and read csv files in general. I think this module might be useful for me during this class because with it I will be able to easily manipulate data from spreadsheets, joining different tables and matching columns with similar fields between them. This module has specific functions just to read files (csv.reader) or to write new csv files (csv.write). Other functions will be useful to set and discover which type of csv file are you dealing with, since different applications might use slightly different configurations for their files, such as different types of separators (commas, tabulations, etc.). Even with csv files with different types of delimiters and quoting characters this package does an excellent job in hiding these nasty details. With this module I will be able to manage relatively large tables in an easy and fast way. Bellow there is a code example obtained at the module website (<https://docs.python.org/3/library/csv.html#module-csv>). In this example the code is opening a file named “names.csv” and it is creating new rows inside of it, adding different entries in the original file.

```
import csv

with open('names.csv', 'w') as csvfile:
    fieldnames = ['first_name', 'last_name']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    writer.writeheader()
    writer.writerow({'first_name': 'Baked', 'last_name': 'Beans'})
    writer.writerow({'first_name': 'Lovely', 'last_name': 'Spam'})
    writer.writerow({'first_name': 'Wonderful', 'last_name':
'Spam'})
```