Lab Exercise 1. Introduction to Python

Why Python? Python is a modern programming language that was developed by Guido van Rossum (now at Google) in 1990 and first released in 1991. While there are a number of programming languages that might be used for a first course in programming, Python offers a number of features that make it particularly applicable:

• It is interpreted, which for the introductory student means that they can type program commands into a console and immediately see the results. This makes learning details of the language much easier

• The syntax of Python is generally simpler than other languages. Python's philosophy is "one way to do it" so that the number of details the student has to learn/remember is reduced. This makes code much more natural and more readable. The focus of learning code development should be on its readability, and python supports this.

• Python provides high level data structures and methods that make many programming tasks much easier. This means that students are productive, writing significant code, much more quickly.

• Python has many libraries which make tasks easier. In particular, there are many libraries specific to a field (biology, chemistry, physics, etc.) that make the language useful for "getting work done". A student who knows Python therefore has a host of software available to them for doing anything from graphics, to gene structure matching, to financial accounting, to music and anything in between

• Python is free! Students can download it from the web for their personal machines. Furthermore, Python does not depend on a particular operating system. Thus students are free to develop their code on Windows, Macintosh OS X or Linux and the code will (for the most part) run anywhere

Where to get it Python is available for any platform (Windows, Mac, Linux) for free download. The main Python site is http://www.python.org and from there you can download a full installation of Python.

What you get When you install Python for your computer, you get a number of features:

- a python "shell", a window into which you can directly type python commands and where interaction between you and programs you write typically occurs.
- a simple editor called IDLE, in which you can type programs, update them, save them to disk and run them. IDLE's interface is essentially the same on any machine you run it because it is a Python program!
- you get access to all the python documentation on your local computer including
  - a tutorial to get you started
  - language reference for any details you might want to investigate
  - library reference for modules you might wish to import and use o other nifty items

Getting Help

IDLE/Python installs a host of help documents. In the Python Shell, if you select Help->Python_Docs you will get a set of web pages in your default browser.
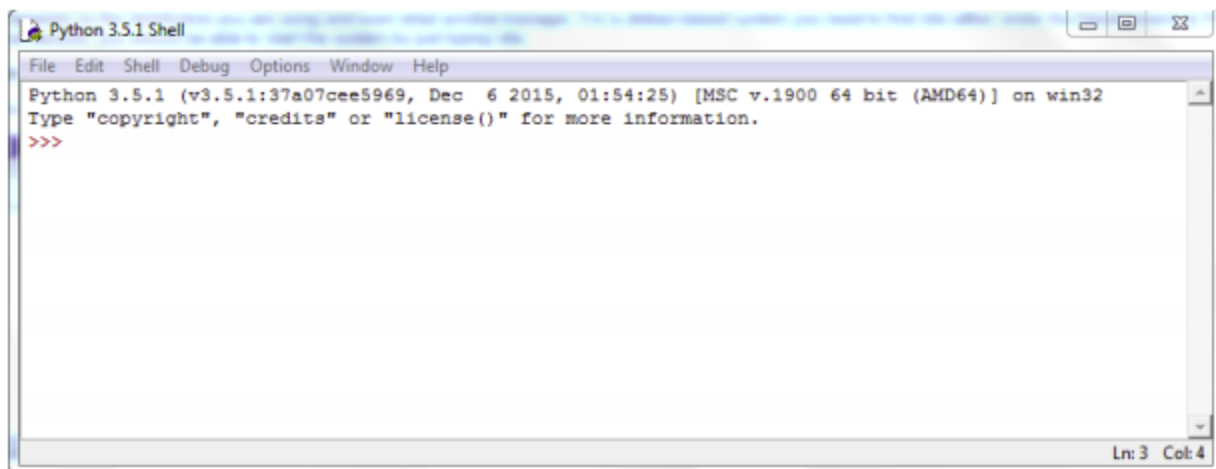
There is a lot of help available. Note that there is a Tutorial available to help you with getting started with Python. There is also a Language Reference for the details of Python and a Library Reference to give you the details of modules you can import.

Finally, www.python.org has a large amount of help available, including Forums.

Getting started To start the Python/IDLE combination on Windows, go to Start Menu -> All Programs -> Python 3.x where x can vary -> IDLE (Python GUI)

For Linux, it depends on the distribution you are using and even what window manager. For a *debian*-based system you need to find IDLE either under the Development or Programming menu (KDE or Gnome). If you just start a terminal session, you should be able to start the system by just typing idle.

Working with the shell The good news is that no matter how you start it, you should get a window that looks pretty much like the following
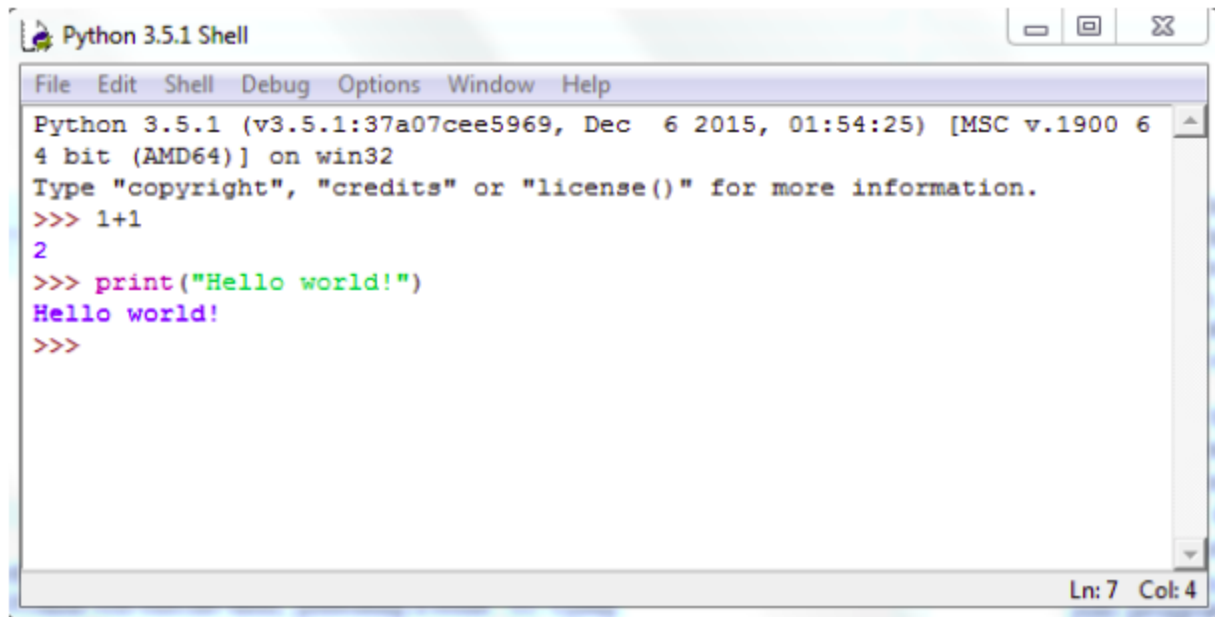

This is the python shell. The shell is interactive, you can type Python commands in the shell and Python will execute them, generating a result. Try typing:

1 + 1 <Enter Key>
print ("Hi World") <Enter Key>
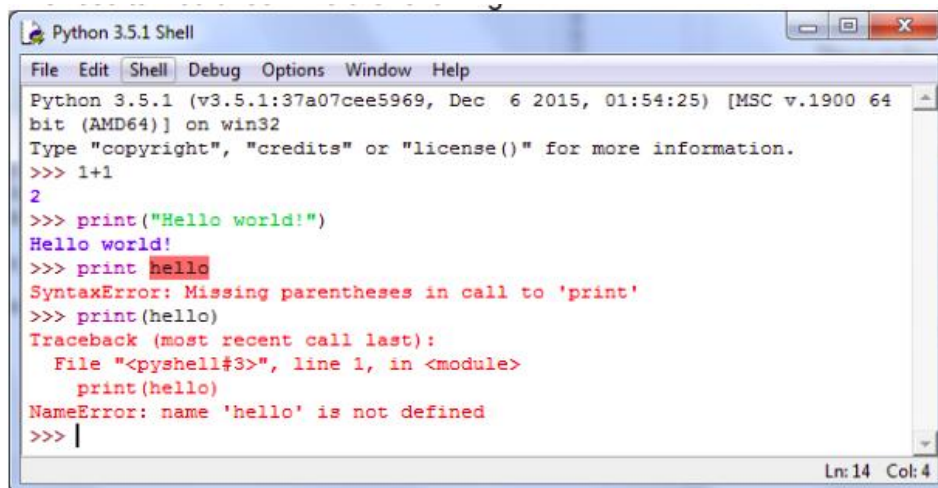
What output do you get? It should look like the window below

What you type shows up in front of the **>>>** prompt. When you type something and hit the Enter key, the result shows up on the next line(s). Sometimes you can get some surprising results, sometimes an error. For example, try entering:

print hello or **print(hello)**

The results would look like the follow

The last shows that an error occurred. **hello** was neither a variable nor did it have quotes around it, so print failed to perform (we'll see more on this later).

Making a program

Typing into the shell is useful, but the commands that you write there are not saved as a file and therefore cannot be reused. We need to save our commands in a file so we can run the program again and again, and more importantly improve it!

To open a file, go to the shell window, click on `File -> New File` A second window will appear into which you can type python commands. This window is an editor window into which we can type our first program.

There is a tradition in computer science. The first program you run in a new language is the Hello World program. This program does nothing but print the words "Hello World". It is a tradition because it does very little except focus on the mechanics of writing your first program and running it. Look at the wikibooks page http://en.wikibooks.org/wiki/List_of_hello_world_programs for more than 200 programming language examples of hello world

In python helloWorld is very easy. Type the following in the "Untitled" window

print ("Hello World")

The phrase after print should be in quotes. Having done that, we should save our first program so that we can use it again. How and where you save a file differs depending on your operating system. For Windows, you left click in the Untitled Window the menu `File->Save As`.

This is on the way to saving the file, not quite completed. It shows the file dialogue and there are a few things to note. First, we type the name of the file at the bottom of the dialogue. The name I have chosen is ***helloWorld.py*** The ***.py*** file extension is important to add. If you do not, then all the nice colouring and formatting you get in the IDLE editing window will be lost. Second, note the location on your computer.

Once saved, you will notice that the title of the editing window changes to the

file name you used to save your program. You can then run your module to see what your program produces. To run the program, select the editing window menu `Run->Run Module`. This results in new output showing up in your python shell

Congratulations, you wrote your first python program.

Some more programming hints Every time you make a change to a file, IDLE asks you to save the file before you run it. If you try to run it before you save, it will display a dialogue box. At that point, you will then be able to run the program.

After some time, you will come to find two keyboard shortcuts very useful.

After you save the file for the first time, the keyboard combination Ctrl-S will resave the file under the present file name. The key F5 will then run the program. So the combo Ctrl-S followed by F5 will help you clean up errors as you go.

Lab Exercise 1a. Getting started with PyCharm

PyCharm is an Integrated Development Environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems such as *git*, and supports web development with Django. PyCharm is developed by JetBrains. It is cross-platform working on Windows, Mac OS X and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm also has an Educational edition which you can install on your own computers and use for coursework.

https://www.jetbrains.com/pycharm-edu/download/

In PyCharm, you organise your work into "projects". There are various

types of projects, but for our purposes, we are interested in "Pure Python" projects only. When you create a project, it has to know where the "interpreter" is. This is the Python interpreter or a "Virtual Environment" which holds the code interpreter and associated additional package libraries. You don't need to be concerned with virtual environments for the moment.

Tasks

1. Open PyCharm.
2. Create a new project (File->New Project). Replace "untitled" in the project name with something meaningful to you. Set the interpreter.
3. You'll see two directories – one with the name of your project and "External Libraries". Your code will live in the directory named for your project.
4. Create a new file (Right-click the project directory→New→New Python File and give it a name, for example helloWorld.py. You now have an editor into which you can type your code.
5. Enter some code - For example `print("Hello World!"`. The code will be saved automatically,
6. Run your module. You can run it by clicking on the green arrow next to your code, by right clicking the file and clicking 'Run' or by using the shortcut 'Ctrl+Shift+F10'. After running you should have a screen like the one below.

Project ▾   ⊕ ÷ ✱ —   helloWorld.py ×

▾ Test ~/PycharmProjects/Test   1 ▶ print("Hello World")
   ▶ ■ venv   2
      helloWorld.py
   ▶ ⅠⅠⅠ External Libraries
      Scratches and Consoles

Run: 🐍 helloWorld ×   ✱ —

▶   ↑   Hello World
■   ↓
           Process finished with exit code 0
📌
🗑

   🐍 Python Console   ⌨ Terminal   ▶ 4: Run   ⚪ Event Log
                                                     2:1 LF UTF-8 4 spaces Python 3.5 (Test)