An Introduction to Python

William Vigor and Clyde Fare

Philosophy of the Course

- Hands on: only this short introductory talk. No one wants to listen to lectures about how to program.
- 3 Workshops:

1. Introducing the basics of python.
2. Using Python for science and data analysis.
3. Advanced Python.

Part I

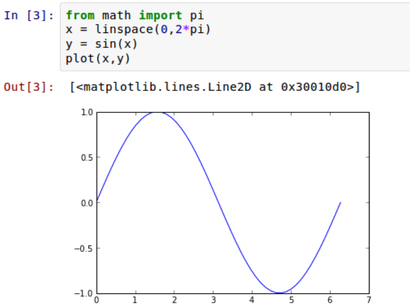
Introduction: What is Programming ?

What is programming ?

- A Computer Program :

“A sequence of instructions that a computer can interpret and execute ”

- In scientific programming these instructions are used to either analyse experimental data or for computational simulation.
- Scientific simulations run on many of the fastest computers in the world. (10,000’s of CPUs)



```
In [3]: from math import pi
        x = linspace(0,2*pi)
        y = sin(x)
        plot(x,y)

Out[3]: [<matplotlib.lines.Line2D at 0x30010d0>]
```

Why Programming is Hard

- Computers are dumb - they can only follow the simplest of instructions.
- Computers are obedient but have no empathy.

" Computers are good at following instructions, but not at reading your mind. " - Donald Knuth

"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realised that a large part of my life from then on was going to be spent in finding mistakes in my own programs." - Maurice Wilkes

Why Learn to program

- Experimental equipment is now commonly computerised.
- A powerful tool to analyse experimental data more efficiently.
- Write computer simulations to complement experimental data.
- A useful transferable skill.

Why Learn to program

- Make science reproducible:
- A program run once with the same inputs should produce the same results if run again.

Part II

Why Python ?

Python Versus other Languages

- Easy to learn but powerful.
- Python's syntax is designed to be readable.

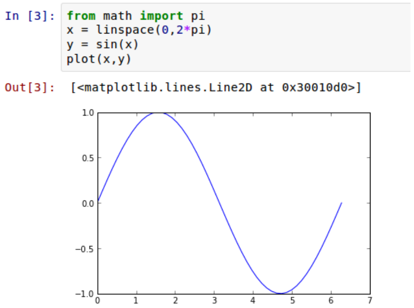
```
for i in [0, 1, 2, 3, 4, 5]:
        print(i)
```

```
#include ⟨ stdio.h ⟩
int main(void)
{
  for (int i=0; i<6; i++)
  {
     printf( "%i \n", i);
  }
  return(0);
}
```

- No need to worry about low level machine details (e.g. memory allocation in C).
- Easy to run - no need for compilation.

Python Versus other Languages

- Batteries included: No need to reinvent the wheel:
- Many open source scientific and other libraries available.
- e.g. Matplotlib for plotting



```
In [3]: from math import pi
        x = linspace(0,2*pi)
        y = sin(x)
        plot(x,y)

Out[3]: [<matplotlib.lines.Line2D at 0x30010d0>]
```

Python is Open source

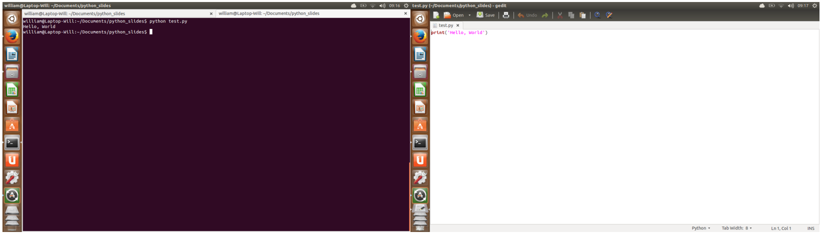
- No need to buy a license, can use it at home even after you leave Imperial.
- Compatible across multiple platforms Linux, Mac, Windows.
- The Python community is very active a large group of people are continually developing new features.
- If a feature is not present then you can add it in yourself might be useful for others.
- You can look at the code to check it is doing what you think it's doing.

Part III

How to can we run Python

From the Command Line

- Put python code into .py file using a text editor.
- Run on the command line: python test.py


william@Laptop-Will: ~/Documents/python_slides
test.py (~/Documents/python_slides) - gedit
Open
Save
Undo
test.py
Python
Tab Width: 8
Ln 1, Col 1
INS

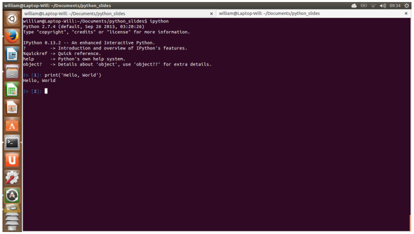
```
william@Laptop-Will:~/Documents/python_slides$ python test.py
Hello, World
william@Laptop-Will:~/Documents/python_slides$
```

```python
print('Hello, World')
```

- Useful for scripts and finished programs.

Using the IPython Interactive environment

- To launch from the command line: IPython
- Interactive session consists of a number of cells which are ran consecutively.

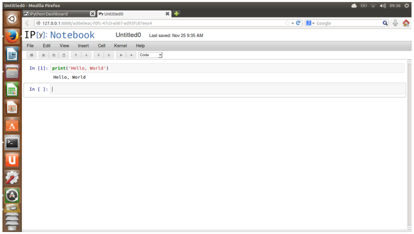

william@Laptop-Will: ~/Documents/python_slides
william@Laptop-Will: ~/Documents/python_slides
william@Laptop-Will: ~/Documents/python_slides
09:34

```
william@Laptop-Will:~/Documents/python_slides$ ipython
Python 2.7.4 (default, Sep 26 2013, 03:20:26)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.2 -- 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]: print('Hello, World')
Hello, World

In [2]: 
```

- Useful for quickly testing out simple ideas which can be put into a .py file later.
- Unfortunately once IPython is closed the code is lost.

Using the IPython Interactive Notebook

- To launch from the command line: ipython notebook –pylab=inline
- Interactive session consists of a number of cells which can be run in any order.
- Can insert plain text and latex notes, links to web pages, images and videos.



```
In [1]: print('Hello, World')
        Hello, World

In [ ]:
```

Using the IPython Interactive Notebook

- Can be used to make a Lab Notebook to analyse data.
- Graphs generated from the code can be directly placed into the notebook.

Other Resources

- General python: http://doc.python.org/2/
- Scipy: www.scipy.org
- Numpy: www.numpy.org
- Further tutorials:
- Anaconda distribution: https://store.continuum.io/cshop/anaconda/

Tips

- Write code which is easy to read. (Code should be written for Humans not computers)
- The simplest solution to a problem is almost always the best.
- Ask questions. It is easy to have misconceptions about programming.