**Python course - Day 1, Lecture 1 Intro - Fabian.**

Structure will be hour lecture then exercises, with final day hack-a-thon.

**Notes:**

Code directly in jupyter notebook. Start new python 2.

Type in directly. []: means coding. If it says [\*] it is still processing a prior command.

Presss Shift + Enter to submit. Enter on it’s own is line break.

Markdown is notation and can be formatted easily.

# Comments ‘’’ is also used and continues across multiple lines until you close it.

Kernel is running in background and periodically may need restarting if code does work.

In python 2.7, division is different to python 3. In 2: 1 / 2 = 0, not 0.5. It does floor division. It decimals. It is normal in Python 3, but in 2, it needs to be 1.0/2. 1.0 interpreted as float when 1 is interpreted as integer. To replicate floor division in python 3, you would use // instead of /.

Power is \*\* and can do comparisons with true false like R. 8<4 = FALSE

Additional module for mathematical/numbers if package NumPy. Good idea to use packages where possible, as they are probably implemented better that you would write them, by experts developed and improved over time.

To add package, use ‘import’ e.g. ‘import math’ import math button.

If you use math.[Tab] drops down tool tip to see all packages.

Can also import just specific function from module, e.g. ‘from math import sqrt’ can then just reference sqrt rather than math.sqrt.

You can also import \* form a module, but wathc out for ‘namespace polluation’ as they may overwrite one of your functions or lead to unintended fucntions.

**Variables:**

Name for a particular object. E.g. x, x = 5 . Equals assigns them.

Can obviously assign them in more useful ways. Be deliberate about it for readability.

Can rename them directly by overtyping in the notebook.

Illegal to use numbers are variable names, or to start a vrting name with number e.g. 1test is illegal. The encourages way to do it is to separate words with underscore, e.g. test\_result. But other conventions are fine such as testResult, just be consistent.

Python accepts multi-assignment: x = y= z=20, three variable set to 20.

Data types. Implicit, dynamically types language. Text and numbers are different types. Numbers are green in Ipythonnotebook, no decimal is int, decimal = float, speech marks = strings e.g.

2 + 2 = 4 but “2” + “2” = ‘22’

Can use function ‘type’ to tell you what type it is: e.g. a = 2.0, type(a) = float.

**Functions:**

Functions have names and arguments. You would use name then () and arguments between the brackets.

Standard (non-packages) functions:

<https://docs.python.org/2/library/functions.html>

Useful to know to avoid naming conflicts.

Can also use help() . Blank gives you how to use help.

To build you own function, you start with definition ‘def’ [name] (arguments) : starts block,

Python will then indent the code meaning it is within the call. Python Idents in-line with execution order. e.g.

Def area(width, height):

return area \* width

Can enter help message between three ‘. E.g. :

def area(width, height):

'''A function to calculate the area'''

return width \* height

Main errors you are likely to encounter are syntax error (e.g. un-expected end of statement), zerodivision error. Also logic errors: python won’t complain about these, but be aware of making them, e.g. even\_number = 57

**Strings:**

White space characters are supported. Print string is ‘print’ (variable), and ipython notebook accepts just the variable name, but other implementations may not. There is an older, but common, convention in python 2 (not 3) where you could use print without (). Supposed to be about it being easy to use, but python 3 made it more regular.

Can use len(), but remember that white space is included.

Can assess certain parts of the string: e.g. ‘for message variable’: message[1] for character, but remember first index is 0, not 1. Remember message[0] If use with print function, can specify formatting and remove ‘’. Can use ranges as well e.g :1-5, but remember the start of the string is inclusive and the end is exclusive.

To get end of strng, you could do len() -1, remember the zero index.

Cab index the string form opposite end: message[-1]