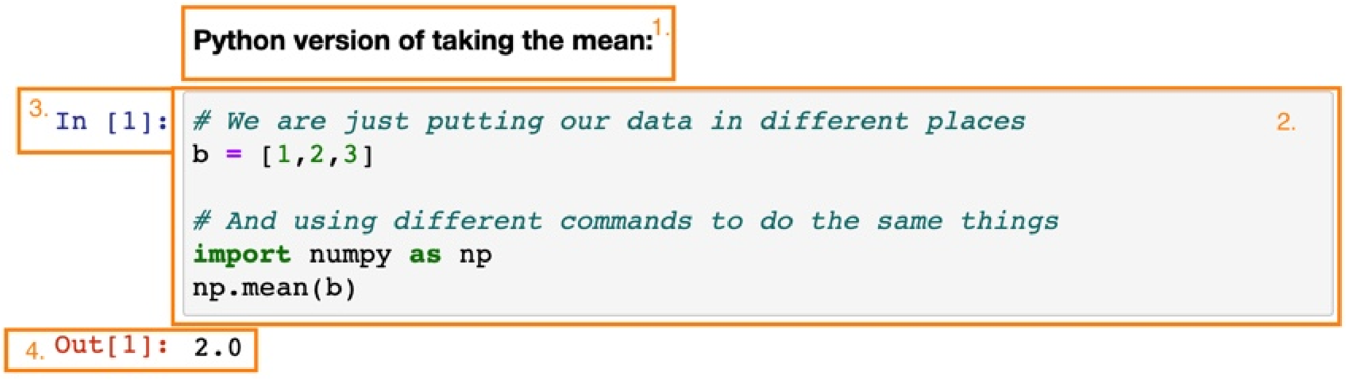
Part one Cheat sheet

Jupyter Notebooks explained:

Anatomy of the Jupyter notebook:

Here is the basic layout of the Jupyter notebook (not including the menus at the top)



```
In [1]: # We are just putting our data in different places
        b = [1,2,3]

        # And using different commands to do the same things
        import numpy as np
        np.mean(b)
Out[1]: 2.0
```

1. Markdown Cell - used to create formatted text using the "markup" language called markdown. No python code is run here
2. Code Cell - Where multiple lines of code can be written, and run at the same time
3. Input Number. If there is a number between the "[]", then the code has been run, the number tells you how many cells have been run in total at the time this cell was run.
4. Output - This shows the same number as the input number, as well as any output that was created on the last line of the code cell. (not no output is created on the last line of code, it will show nothing)

Edit mode vs command mode:

If your curser is blinking within a cell of code (and you can type something into that cell), then you are in "edit mode" if you click to the side, type escape, or ctrl (command) + m, you will go into command mode, where you can more easily add delete and move cells. See here for a list of shortcut keys (also in this folder)

Jupyter Notebook Commands

Here are a few of the more important keyboard shortcutes

Running cells

In edit mode, You can run a command in several ways. (**Note:** mac keys are in parentheses below)

- ctrl (command) + enter (return) - Run that cell, can be markdown or Python
- alt (option) + enter (return) - Run that cell and insert a new cell underneath

Editing cells:

In Edit mode, you can use many commands to navigate the text and write text more easily including copying (ctrl (command) + c), cutting (ctrl (command) + x), pasting (ctr (command) + v), undoing (ctrl (command) + z), redoing (ctrl (command) + shift +z), selecting all (ctrl (command) + a), and saving (ctrl (command) + s).

One nifty one is to comment or uncomment a whole line of code using ctrl (command) + /. many more can be found here

The basic Building Blocks of Python

Basic Data Types:

Numerical data

Integers (`int`), and floats (`float`) are for whole numbers, and numbers with decimal places. Python automatically guesses what you want when you type in a number with or without a decimal point. It will also automatically change datatypes after a calculation.

Strings

to type text data (aka "strings" aka `str`) into your python code wrap it in single quotes (`''`), or double quotes (`""`), you just need to put the same one on either end of the text. Example:

```
"this is text ok you folks"
```

Basic magic powers, or functions!

Using a function in python is like using one in Excel, all you need is the name, some curved brackets, and the data/ object(s) you need to put into the function:

```
function(input_object1,input_object2)
```

The type function:

`type(thing)` - this function tells you what type of thing is in between the brackets, important as not all types of things can do all the things in python 😉. (in Programmer jargon type = a "class", and thing = an "object")

The Length function (`Len`)

the `len` function finds the length of an object, if that python object has a length (examples in include strings, which have a numer of characters as their length, or lists that count the number of items as their length)

```
len('the answer is 16')
# will output 16 because thats how many characters there are
```

type coercion functions:

Every type of thing in python will have a coercion function that will be able to change similar objects into that type. These are always called the class_name(input object):

```
int(1.4) # coerces things into integers
float(1) # coerces things into floats
str(14) #coerces things int strings
```

Methods, functions that have the input in front of them

Methods are a special type of function that is tied to a particular object, its important to note because the look funny, the first input of the function actually goes ***in front*** of the function name when used in python:

```
input_object.method()
```

example string method:

`str.upper` turns a string into a trump tweet... by making all of the letters capitalised 😛.

```
'covfefe'.upper()
# will output `COVFEFE'
```

See how we use a . and then type the name of the function after the input? Methods are super common in python so it's important to watch out for them.

Variables

If you want to save some data or an object and use it again in a later line of code or cell, then you need to assign it to a variable (like giving it a nick name). You can do this using the `=` sign:

```
variable_name = thing
# real example
Garbo = 'Jonathan Lloyd Garber'
```

They are usefull for capturing output of functions, and for putting data into functions:

```
length_of_garbos_name = len(Garbo)
```