Intro to Python

Objectives

1. work with python!

Credit:

things here are a mix of the really excellent Software Carpentry tutorial on Python: http://swcarpentry.github.io/python-novice-inflammation/ (http://swcarpentry.github.io/python-novice-inflammation/) .

and the official python tutorial: https://docs.python.org/3/tutorial/ (https://docs.python.org/3/tutorial/)

And Ryan Abernathys open book, which we will be looking at a lot! https://earth-env-data-science.github.io/lectures/core_python/python_fundamentals.html (https://earth-env-data-science.github.io/lectures/core_python/python_fundamentals.html)

I've made some slight adaptations here and there, but the credit goes to those organizations. I hope I am using this correctly under the licences:

https://earth-env-data-science.github.io/LICENSE.html (https://earth-env-data-science.github.io/LICENSE.html)
https://swcarpentry.github.io/python-novice-inflammation/LICENSE.html (https://swcarpentry.github.io/python-novice-inflammation/LICENSE.html)

Working with Python

There are three main ways to use python.

1. By running a python file in a terminal, e.g:

```
>>>$ python myscript.py
```

2. Through an interactive console (python interpreter or ipython shell)
3. In an interactive notebook (e.g. Jupyterlab)

In this course, we will mostly be interacting with python via Jupyter notebooks like the one we are using right now

Basic Variables: Numbers and Strings

A `variable` is a label for a piece of data

In [2]:
```
# comments are anything that comes after the "#" symbol
# a comment is a note that is not  "run" by python, ie not code

a = 1       # assign 1 to variable a
b = "hello" # assign "hello" to variable b
```

Nothing happened when we ran the code cell, how do we see the variables?

We can use out first built-in python function: `print()`

In general in python a function will have a name (print) followed by parenthesis `()`. Things (called 'arguments') that you put into `()` get passed to the function and used to do whatever the function does.

Let's use `print()` to see our variables

In [3]:
```
# how to we see our variables?
print(a)
print(b)
print(a,b)
```

```
1
hello
1 hello
```

All variables are code 'objects'. Every object has a type (class). To find out what type your variables are use the built in function `type()`

In [4]:
```
# what do you think the output here means?

print(type(a))
print(type(b))
```

```
<class 'int'>
<class 'str'>
```

In [5]:
```
# as a shortcut, iPython notebooks will automatically print whatever is
 on the last line
type(b)
```

Out[5]:
```
str
```

we can check for the type of an object using the built-in keyword `is`, which checks if two things are equal

In [6]:
```
# we can check for the type of an object
print(type(a) is int)
print(type(a) is str)
```

```
True
False
```

note that `True` and `False` are special objects, called booleans, in python. They are useful for all sorts of codeing stuff

We just saw two types of data objects: `int` and `str`

Different code 'objects' have `attributes` and `methods`

these are features of the code object that allow you to do stuff with it

they can be accessed via the syntax `variable.method`

In code notebooks like this one, you can autocomplete if you press `<tab>` to show you the methods available for a code object

There are a million, but here is an example of one for strings which lets you capitalize a word

In [7]:
```
# this calls the method
b.capitalize()
# there are lots of other methods
```

Out[7]:
```
'Hello'
```

in general if you use a method (like `b.capitalize()`) that does an action to your data you will follow it with `()` - think of the `()` as the sign for an action, or function.

you can perform calculations with variables

these calculations act differently for different types of objects.

What happens if we use `+` to add: numbers, strings, and numbers to a string?

In [8]:
```
# binary operations act differently on different types of objects
c = 'World'
print(b + c)
print(a + 2)
print(a + b)
```

```
helloWorld
3

---------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-8-958e0fcf701c> in <module>
      3 print(b + c)
      4 print(a + 2)
----> 5 print(a + b)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Math

Basic arithmetic and boolean logic is part of the core python library.

In [10]:
```
# addition / subtraction
1+1-5
```

Out[10]: -3

In [11]:
```
# multiplication
5 * 10
```

Out[11]: 50

In [12]:
```
# division
1/2
```

Out[12]: 0.5

In [13]:
```
# that was automatically converted to a float
type(1/2)
```

Out[13]: float

In [14]:
```
# exponentiation
2**4
```

Out[14]: 16

In [15]:
```
# rounding
round(9/10)
```

Out[15]: 1

Logic

You can use keywords to perform logical operations in python.

this starts to become the basics of coding

In [17]:
```
# logic
True and True
```

Out[17]: True

In [18]:
```
True and False
```

Out[18]: False

In [19]:
```
True or True
```

Out[19]:
```
True
```

In [9]:
```
(not True) or (not False)
```

Out[9]:
```
True
```

Break time: go to breakout rooms and reflect on this intro in groups. What doesn't make sense? what doesnt?

In [ ]:
```

```