**Lecture 2 (week 2.1)**

Tuesday 13th 2016

Numbers and Strings: basic operators and manipulations

Sections 1.1 to 1.8 in <http://greenteapress.com/thinkpython2/html/thinkpython2002.html>

Sections 2.1 to 2.9 in <http://greenteapress.com/thinkpython2/html/thinkpython2003.html>

Dividing by integers

The floor division operator, //, divides two numbers and rounds down to an integer.

Try:

**In[#]: 1//2** returns **0** (divides 1 by 2, which is 0.5, rounded down is 0)

**In[#]: -1//2** returns **-1** (divides -1 by 2, which is -0.5, rounded down is -1)

Boolean variable

Python can compares numbers to give logical (boolean type) results:

**In[#]: 1 < 2**

**True**

**In[#]: 1 < 1**

**False**

**In[#]: 1 <= 1** 1 =< 1 gives a syntax error

**True**

**In[#]: 1 == 1.0** 1 = 1 gives an error; mixing types is OK

**True**

**In[#]: 1 != 1.0**

**False**

**In[#]: 3 >= 2 > 1** Many languages don't support this one

**True**

You can save a comparison into a boolean variable:

**In[#]: a = 2.0 > 1**

**In[#]: a**

**True**

String slicing

Python strings are just sequences of individual characters and share their manipulation properties with other Python sequences which we will introduce later (*lists* and *tuples*). The sequence of individual characters of a string is also called an *array*, which means that they are labeled and can be accessed individually as explained below. Here are some examples how to cut/slice strings:

**In [41]: toto="hello world"**

**In [42]: toto[1]**

**Out[42]: 'e'**

**In [43]: toto[0]**

**Out[43]: 'h'**

**In [44]: toto[0:4]**

**Out[44]: 'hell'**

**In [45]: toto[1:4]**

**Out[45]: 'ell'**

**In [46]: toto[:4]**

**Out[46]: 'hell'**

**In [47]: toto[4:]**

**Out[47]: 'o world'**

**In [48]: toto[0:8]**

**Out[48]: 'hello wo'**

**In [49]: toto[0:8:2]**  Same as above, but jumps forward two positions

**Out[49]: 'hlow'**

Python built-in Functions

Python has only a very few built-in function which you can call from the prompt interpreter. They are listed there:

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

For now we will use a very small number of them, such as **str()**, **type()**, or **print()**

Python Math module

The standard functions like sqrt(x), sin(x), exp(x), log(x) are not included in Python.

**In[#]: sqrt(2)**

**---------------------------------------------------------------------------**

**NameError Traceback (most recent call last)**

**<ipython-input-12-40e415486bd6> in <module>()**

**----> 1 sqrt(2)**

**NameError: name 'sqrt' is not defined**

But they are in the module called math. To access them, you must first import the math module, then preface the function names with math. (note the dot!). The function name is called a “method” of the math module.

**In [13]: import math**

**In [14]: math.sqrt(2)**

**Out[14]: 1.4142135623730951**

**In [15]: math.sin(20.4)**

**Out[15]: 0.9997929001426692**

Here is a list of math functions (i.e. methods) you can use with the **math** module:

<https://docs.python.org/3/library/math.html>

String Methods

Methods have a very deep and precise meaning in Python. For now, you will use methods as a function that applies to a certain object. If the object is a string, there are many functions (called “methods” in python) you can apply to it. The syntax is always:

**In[#]: myobject.method(parameters)**

would run the method **method** on object **myobject** using the optional parameters **parameters**. For example, imagine you want to run some methods on a string (your object). Start by defining the string:

**In [58]: charvar="hello world"**

**In [59]: charvar**

**Out[59]: 'hello world'**

then try out the **capitalize** and **count** methods:

**In [60]: charvar.capitalize()**

**Out[60]: 'Hello world'**

**In [61]: charvar.count('a')**

**Out[61]: 0**

**In [62]: charvar.count('o')**

**Out[62]: 2**

**In [63]: charvar.count('o',2,4)**

**Out[63]: 0**

The exhaustive list of string methods are given there:

<https://docs.python.org/2/library/stdtypes.html#string-methods>

Python has an extensive “help” system. For instance you can also access all methods of a String with a **help** command line by typing:

**In[#]: help(str)**

Where **str** refers to strings. The deep reasons behind what this help command is doing will appear clear much later in the course (for the expert, **str()** is a *class*). For now, just use this as a standard “help” function.

The object does not have to be a string. In fact it can pretty much be anything, for now we will explore methods on strings and numerical variables.

Python Script Files

When your task requires more than a few lines, or may be useful in the future, you should use a text-editor to make a *script* file, instead of typing everything every time. It's most common to call your script file **filename.py** where **filename** is some name of your choosing.

You run the script from the command line by typing

**In[#]: run filename.py**

If your script file has no syntax errors, and does not produce any runtime errors, you will see the results.

You will experiment with scripts during the lab today.

Note that you can also run a Python script from command line:

**ipython filename.py**

Syntax and runtime errors

If python does not understand what you have typed, you get a syntax error:

**In [16]: 1 2**

**File "<ipython-input-16-f303b7d2f2b8>", line 1**

**1 2**

**^**

**SyntaxError: invalid syntax**

If the syntax of what you type is correct but fails to run it, you get a runtime error:

**In [17]: hello,world**

**---------------------------------------------------------------------------**

**NameError Traceback (most recent call last)**

**<ipython-input-17-757228086dc1> in <module>()**

**----> 1 hello,world**

**NameError: name 'hello' is not defined**