CH2 Notes

April 28, 2018

1 Chapter 2 Notes

1.1 Running Helloworld.py

Breaking it down, your computer knows how to interpret the text you give it due to the .py file extension, lettin the computer know it's python, which you have installed on your computer. It then uses a python interpreter (similar to the kernels in Jupyter notebook) taht can run python code

2 Variables

Variables are how you store/access information easily, for example:

```
In [51]: variable ='This is a variable' #stores the value - this is a variable to variable
         print(variable) #accessing the contents of my variable by applying a function to it
```

```
This is a variable
```

2.0.1 editing a variable

You can reassign the value of a variable easily by giving a variable another value.

```
In [52]: variable = "I have changed variable's value by reassigning it to this text"
         print(variable)
```

```
I have changed variable's value by reassigning it to this text
```

2.0.2 variable naming guidelines

MOST IMPORTANT RULE YOU CANT USE SPACES BECAUSE THAT SCREWS EVERYTHING UP, instead use underscores after a word, for example

```
In [54]: this_is_four_words = '4'
```

Rules cont:

1. Variables only contain letters, numbers and underscores

2. You can't start variables with a number
3. Avoid using python built-in words like print, len, str, int, and stuff as a variable name
4. Variable names should be descriptive of the contents, but also concise

2.0.3 Avoiding name errors with variables

What is a name error - most often a typo, or referencing a variable that has not been created yet

```
In [55]: # example one:
         my_name = 'Joby'
         print(My_name) #notice the capital M, python variables are case sensitive unlike SQL
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-55-cff800372110> in <module>()
  1 # example one:
  2 my_name = 'Joby'
----> 3 print(My_name) #notice the capital M, python variables are case sensitive unlike SQ


NameError: name 'My_name' is not defined
```

```
In [56]: #example_two
         print(Joby) #i never made the Joby variable yet, so i can't use it for anything
         Joby = 'Joby'
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-56-be87acddd540> in <module>()
  1 #example_two
----> 2 print(Joby)
  3 Joby = 'Joby'


NameError: name 'Joby' is not defined
```

3 Excercises 2-1, 2-2. See my .py files

4 String data type

Strings are data wrapped in quotes, for example "1" is a string, however 1, without the quotes is an integer.

You can use single quotes, ’’, or double quotes, "", to denote a string.

If you have an apostrophe, or you want to use "" in your string, you would want to wrap the string with the opposite type of quote, so Python does not interpret the text as input.

4.1 Changing cases w/String methods

Review on methods - methods are functions that can be applied to a class, which will be covered later, however every instance of that class type has the same methods.

In particular, we can use some special string methods, showcased below to help formout our" strings.

```
In [2]: #title case capitalizes first letters of each word with some interesting exceptions as

In [3]: name = 'ada lovelace'
        print(name.title())

Ada Lovelace


In [4]: text = "john's Pizza" #the S will be capitalized due to this method!
        print(text.title())

John'S Pizza


In [5]: #upper capitalizes all letters, lower lowercases all letters

In [6]: print(name.lower())

ada lovelace


In [9]: print(name.upper())

ADA LOVELACE
```

4.1.1 Recap of methods

1. .title() - title cases (capitalizes first letter)
2. .upper() - capitalizes all letters
3. .lower() - makes all letters lowercase - a useful method to ensure stable formatting when data cleaning