

module1a

January 22, 2024

0.1 Math Practice

```
[7]: #####  
    ## EXECUTE THIS CELL BEFORE YOU TEST YOUR SOLUTIONS ##  
    #####  
  
    import imp  
    sol = imp.load_compiled("sol", "./sol.py")  
    from nose.tools import assert_equal
```

0.2 A note on variable initialization

Variable initialization refers to the act of declaring a variable (giving it a name) and assigning it a value. A variable's name is declared on the left side of the = and its value on the right side. This can be done in a single line of code, like the below. Note that a variable's name may not contain spaces or special characters, with the exception of the underscore _.

```
my_variable = 1
```

Once initialized, referring to `my_variable` in subsequent lines of code will actually refer to `my_variable`'s value, which has been set to 1. However, as their name implies, variables can have changing values. For example, we might do something like the below:

```
my_variable = 1  
print(my_variable)      # This will print out the value of my_variable  
# >>> 1
```

```
my_variable = 1 * 2      # We've now changed the value of my_variable to be (1 * 2) = 2  
print(my_variable)  
# >>> 2
```

We can also reference a variable's current value in it's own re-assignment! This can be done if the variable has already been assigned a value previously. For example:

```
my_variable = my_variable * 3  
print(my_variable)  
# >>> 6
```

What the above line is doing is setting the value of `my_variable` equal to the current value of `my_variable`, which we set to 2, multiplied by 3. This ends up being: `my_variable = 2 * 3 = 6`

Calculate the result of 3.93 multiplied by 4901 and save it in a variable named 'q1', then print it out.

```
[8]: # your code here
q1 = 3.93 * 4901
print(q1)
```

19260.93

```
[9]: #####
    ### TEST YOUR SOLUTION ###
    #####
    # Note: All test cases are read-only

    assert_equal(q1, sol.q1)
    print("Success!")
```

Success!

Calculate the result of 215 divided by 6 and save it in a variable named 'q2', then print it out.

```
[10]: # your code here
q2 = 215 / 6
print(q2)
```

35.833333333333336

```
[11]: #####
    ### TEST YOUR SOLUTION ###
    #####

    assert_equal(q2, sol.q2)
    print("Success!")
```

Success!

Calculate the result of 3 divided by 0.3 and save it in a variable named 'q3', then print it out.

```
[12]: # your code here
q3 = 3 / 0.3
print(q3)
```

10.0

```
[13]: #####
    ### TEST YOUR SOLUTION ###
    #####

    assert_equal(q3, sol.q3)
    print("Success!")
```

Success!

Calculate the remainder of 215 divided by 6 and save it in a variable named 'q4', then print it out.

```
[14]: # your code here
q4 = 215 % 6
print(q4)
```

5

```
[15]: #####
### TEST YOUR SOLUTION ###
#####

assert_equal(q4, sol.q4)
print("Success!")
```

Success!

Calculate the value of 9 raised to the 12th power and save it in a variable named 'q5', then print it out.

```
[16]: # your code here
q5 = 9 ** 12
print(q5)
```

282429536481

```
[17]: #####
### TEST YOUR SOLUTION ###
#####

assert_equal(q5, sol.q5)
print("Success!")
```

Success!

Cast 3.5 to an integer and save it in a variable named 'q6', then print it out.

```
[18]: # your code here
q6 = int(3.5)
print(q6)
```

3

```
[19]: #####
### TEST YOUR SOLUTION ###
#####

assert_equal(q6, sol.q6)
```

```
print("Success!")
```

Success!

0.3 Data Types Practice

Calculate the data type of "False" (notice the **quotes** around the word False!) and save it in a variable named 'q7', then print it out.

```
[20]: # your code here
q7 = type("False")
print(q7)
```

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

```
[21]: #####
    ### TEST YOUR SOLUTION ###
    #####

assert_equal(q7, sol.q7)
print("Success!")
```

Success!

Calculate the data type of True and save it in a variable named 'q8', then print it out.

```
[22]: # your code here
q8 = type(True)
print(q8)
```

```
<class 'bool'>
```

```
[23]: #####
    ### TEST YOUR SOLUTION ###
    #####

assert_equal(q8, sol.q8)
print("Success!")
```

Success!

Calculate the data type of the result of 1000 divided by 10 and save it in a variable named 'q9', then print it out.

```
[24]: # your code here
q9 = type(1000 / 10)
print(q9)
```

```
<class 'float'>
```

```
[25]: #####  
      ### TEST YOUR SOLUTION ###  
      #####  
  
      assert_equal(q9, sol.q9)  
      print("Success!")
```

Success!

Cast the value of 6.3 divided by 3.8 to an integer. Save it in a variable named 'q10', then print it out.

```
[26]: # your code here  
      q10 = int(6.3 / 3.8)  
      print(q10)
```

1

```
[27]: #####  
      ### TEST YOUR SOLUTION ###  
      #####  
  
      assert_equal(q10, sol.q10)  
      print("Success!")
```

Success!

0.4 String Practice

Concatenate the strings 'James', 'Brian', and 'Patrick' - store the result in a variable called 'q11', then print it out. Make sure to add a single empty space between the names!

```
[28]: # your code here  
      q11 = 'James ' + 'Brian ' + 'Patrick'  
      print(q11)
```

James Brian Patrick

```
[29]: #####  
      ### TEST YOUR SOLUTION ###  
      #####  
  
      assert_equal(q11, sol.q11)  
      print("Success!")
```

Success!

Make the following string correct, storing it in a variable named 'q12': q12 = "4 % 2 = " + (4 % 2))

```
[33]: # your code here
q12 = "4 % 2 = " + str(4 % 2)
print(str(q12))
```

4 % 2 = 0

```
[34]: #####
    ### TEST YOUR SOLUTION ###
    #####

assert_equal(q12, sol.q12)
print("Success!")
```

Success!

Save the following quote, (including the double quotes), in a variable called 'q13', then print it:
Albert Einstein's best quote is "I have no special talent. I am only passionately curious."

```
[36]: # your code here
q13 = "Albert Einstein's best quote is \"I have no special talent. I am only_
    ↳passionately curious.\""
print(q13)
```

Albert Einstein's best quote is "I have no special talent. I am only passionately curious."

```
[37]: #####
    ### TEST YOUR SOLUTION ###
    #####

assert_equal(q13, sol.q13)
print("Success!")
```

Success!

```
[ ]:
```