Introduction to Python Programming

Andrew J. Pounds, Ph.D.

Exercise Three: Mathematical Operations

You have probably guessed that Python is a really powerful language for doing mathematics. However, to quote Spider Man, "With great power comes great responsibility." In this exercise we will cover some of the basics that you must know in order to do things properly.

1. PEMDAS

Python does follow the PEMDAS rules. For example to code 2^2+3^2 you would type:

```
2**2+3**2
```

with the result being 13. To code 3(22+(3-42)) you would type

```
3*(2**2+(3-4**2))
```

And should get and answer of -27.

2. Integer Division and Remainders

This is a strange one. Imagine you are doing the following calculation:

```
y=2 / 3
```

To your shock the value is always zero and when you do the calculation

```
y = 3 / 2
```

The answer is always 1. The issue here is that the value 2. And 2.0 are different thatthe value of 2 in the computer. The first two are stored as real (with a decimal point) numbers and the last one is strictly an integer. If we do a calculation with ONLY integers then the result will be an integer as shown above.

Integers are used ubiquitously in computer science because they take less space than reals, are faster to calculate. There is also a special operation that can be applied to integers called the modulus (or remainder). The symbol for the mod operator is the percent sign.

For example:

```
>>> 8 % 3
2
>>> 8 % 4
0
```

This is strange output until you realize that 8 is evenly divisible by 4. If you divide 8 by 3 it will go into 8 twice with a remainder of 2. Python does allow for taking the modulus between floating point numbers, i.e. -- 3.4 % 2.1, but the use for such methods typically are reserved for esoteric mathematical operations that will not be needed in this class.

3. Trigonometric, Transcendental, and Special Funtions

Python has a rich library of functions for trigonometric, transcendental, and special functions. It even has symbols for many of the constants that one might need access to in performing higher math. To access these functions you MUST include the following statement at the top of your program

```
import math
```

Once this is done you can enter to use the functions in this library. For example

```
>>> math.cos(math.pi)
-1.0
```

The preceding “math.” is important because it tells python that you are using a function FROM THE LIBRARY. The math library works on real (not complex) numbers. If you want to work on complex numbers you would need the cmath library -- but that can also greatly complicate the process of debugging your program.

To see the full list of functions in the math library for python version 2, go to this link.