Python Math

I. OVERVIEW

Computer programming languages can be divided into three general categories:

- High-level vs low-level (higher abstraction, easier to code vs harder but faster running)
- Interpreted vs compiled (translates code line-by-line vs processes entire program first)
- General-purpose vs domain-specific (useful across many problem areas vs specialized)

Python is a high-level, interpreted, general-purpose programming language that can be used for almost any type of application. It is easy to learn and quite popular, even being taught at top universities as a first programming language [1]. And Python is widely-used in industry. [2]

You can use Python's interpreter in either script mode or shell mode. Script mode is where the interpreter executes your source code line-by-line from a file. The animated examples simulate using script mode. Shell mode allows you to type expressions directly into the Python shell and have the interpreter execute them immediately. Section III below demonstrates using IDLE (a special shell) as a calculator, though you can run *any* Python commands from IDLE.

Our scope is to write short computational scripts using algorithms [3] that illustrate selected algebra and coding concepts. Teaching a whole programming language, however, is better left to the many good resources already available online. We encourage interested students to start with a Python Tutorial [4] and read up to the section called 'More Control Flow Tools' (**if** statements and **for** statements). The concepts and skills you'll learn by studying Python won't be wasted. They they are universal to almost all programming languages.

We also recommend that you actually install Python on your home Windows machine and practice using the shell and writing programs on your own. You'll learn a heck of lot more by going through the hands-on mechanics yourself. You can download Python from the home page for free. [5]

For the advanced student, consider buying a Raspberry Pi [6]. It comes with Python installed, and you'll also own a full-blown computer for only $40! But its most interesting feature is that it has electrically programmable General Purpose Input/Output (GPIO) pins. These pins, in turn, can control other electrical devices, such as LEDs, switches, and motors. Think of the amazing DIY and Maker projects you can build! A Pi could open up a whole new world of electronic and programming hobbies. An example of a Pi controlling three LEDs is given in the 'Raspberry Pi (Python)' animation at the top of this webpage. A circuit schematic is given on the last screen of that animation.

II. PYTHON MATH OPERATORS AND FUNCTIONS

These operators work natively for types **int** (integer) and **float** (floating point).

+	addition
-	subtraction
*	multiplication
/	division
//	integer division
**	raise to a power
%	remainder
==	equality comparision operator
!=	inequality comparison operator
>	greater than operator
<	less than operator
>=	greater than or equal to operator
<=	less than or equal to operator

Below is a partial list of functions contained in the 'math' module. The Python math page has the full list. [5] A module is an external file containing Python statements and definitions. You must *import* a module (e.g. 'import math') before gaining access to its functions.

math.exp(x)	e**x (raises e to the x power)
math.log10(x)	base 10 log of x (the common logarithm of x)
math.log(x)	base e log of x (the natural logarithm of x)
math.log(x,base)	the change-of-base formula: $log_{base}(x) = log_a(x)/log_a(base)$
math.pow(x, y)	x**y (either variable can be an int or a float)
math.sqrt(x)	square root
math.pi	the constant pi (3.141592653589793)
math.e	the constant e (2.718281828459045)
math.degrees(x)	convert radians to degrees
math.radians(x)	convert degrees to radians
math.sin(x)	sine (in radians)
math.cos(x)	cosine (in radians)
math.tan(x)	tangent (in radians)
math.asin(x)	arc sine (in radians)
math.acos(x)	arc cosine (in radians)
math.atan(x)	arc tangent (in radians)
math.hypot(x, y)	hypotenuse = math.sqrt(x*x + y*y)

Examples of all the above operators and functions are given (in order) in Section III.

III. EXAMPLES OF MATH OPERATORS AND FUNCTIONS IN IDLE

IDLE is Python's Integrated DeveLopment Environment. Here's how IDLE works: Enter your valid Python commands at the shell prompt >>>, then press the Enter key. The result of your command is shown on the next line in blue, followed by a new shell prompt on the following line. If, however, you entered a nonvalid command, or tried using a function before first importing the math module that contains it, you will get a Traceback error message. The 'import math' instruction is shown in IDLE below, and also in the 'Quadratic (Python)' and 'Raspberry Pi (Python)' animations at the top of the webpage. A Traceback message is shown below. The # symbol represents a comment. Comments are ignored by the Python interpreter. All operators and functions work the same in actual Python programs as they do within the IDLE environment. The examples below follow the order given in Section II.

IDLE

```
Python 3.5.2 Shell
File   Edit   Debug   Options   Window   Help
-------------------------------------------------------------------------------
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> 3+2        # adding two (or more) integers (ints) results in another int
5
>>> 3+2.0       # adding an int to a float (2.0) results in a float
5.0
>>> 3-2       # subtracting two ints results in another int
1
>>> 3.0-2       # subtracting when one (or more) number is a float results in a float
1.0
>>> 3*2       # multiplying two ints results in another int
6
>>> 3*2.0       # multiplying when one (or more) number is a float results in a float
6.0
>>> 3//2       # dividing two ints using // (integer division) results in another int
1
>>> 3//2.0       # dividing when one number is a float using // results in a float
1.5
>>> 3/2       # dividing two ints using / results in a float (ver 3.x and above)
```

```
1.5
>>> 3.0/2        # dividing using / when one number is a float results in a float
1.5
>>> 3**2         # raising an int to an int power results in an int
9
>>> 3.0**2       # if either number is a float, the result is a float
9.0
>>> 3**2.0       # if either number is a float, the result is a float
9.0
>>> 5**0.5       # square root of 5 (5 raised to the one half power)
2.23606797749979
>>> 3%2          # remainder–when both are ints, the remainder is an int
1
>>> 3%2.0        # remainder–when either is a float, the remainder is a float
1.0
>>> 3==2         # equality operator–checks whether two numbers are equal...
False
>>> 3==3         # ...and never changes either number
True
>>> 3!=2         # inequality operator...and never changes either number
True
>>> 3!=3         # inequality operator...and never changes either number
False
>>> 3>2          # greater than operator
True
>>> 3<2          # less than operator
False
>>> 3>=2         # greater than or equal to operator
True
>>> 3<=2         # less than or equal to operator
False
>>> math.exp(2)      # Calling function before importing math module causes error
Traceback (most recent call last):
    File "<pyshell#30>", line 1, in <module >
        math.exp(2)
NameError: name 'math' is not defined)
>>> import math       # import the math module
>>> math.exp(2)       # e**2 using function from math module
7.38905609893065
```

```
>>> math.log10(2)        # log of 2 to the base 10
0.3010299956639812
>>> math.log(2)         # log of 2 to the base e
0.6931471805599453
>>> math.log(100,5)      # Example: log5100 = log10100/log105 = math.log(100, 5)
2.8613531161467867
>>> math.pow(2,3)         # 2**3, raising 2 to the 3rd power
8.0
>>> math.sqrt(121)         # square root of 121
11.0
>>> math.pi         # returns the constant pi
3.141592653589793
>>> math.e         # returns the constant e
2.718281828459045
>>> math.degrees(1)         # convert radians to degrees (1 rad ≈ 57.3 deg)
57.29577951308232
>>> math.radians(90)         # convert degrees to radians (90 degrees ≈ 1.6 radians)
1.5707963267948966
>>> math.sin(90)         # We meant degrees so this is wrong. (Python uses radians.)
0.8939966636005579
>>> math.sin(math.radians(90))    # Convert to radians, sin(90) = 1.
1.0
>>> math.cos(math.radians(90))     # Convert to radians, cos(90) = 0.
6.123233995736766e-17
>>> math.tan(math.radians(45))     # Convert to radians, tan(45) = 1.
0.9999999999999999
>>> math.asin(math.radians(45))     # Convert to radians, asin(45) = 0.90.
0.9033391107665127
>>> math.acos(math.radians(45))     # Convert to radians, asin(45) = 0.66.
0.6674572160283838
>>> math.atan(math.radians(45))     # Converting to radians, atan(45) = 0.66.
0.6657737500283538
>>> math.hypot(3, 4)    # The classic 3-4-5 triangle.
5.0
```

 

V. REFRENCES AND RESOURCES

[1] "Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities," By Philip Guo, Communications of the ACM, July 7, 2014
http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introduc

[2] Python is widely used in industry:
https://www.python.org/about/success/

[3] Algorithm (Wikipedia article):
https://en.wikipedia.org/wiki/Algorithm

[4] This is the Python Tutorial:
https://docs.python.org/3/tutorial/index.html

[5] This is the Python home page:
https://www.python.org/

How to install Python on Windows for newbies:

On the Python home page, find where it says 'Latest:' (it's under 'Download'). Then select '3.x.x', where x represents the latest point release of version 3. (Point releases change frequently, so we don't use numbers here.) After clicking 3.x.x takes you to the the next page, look under 'Files' and select 'Windows x86 executable installer'. Follow the instructions from there.)

[6] This is the Raspberry Pi website:
https://www.raspberrypi.org/

