Fundamental of Python usage

Lecture 2: Fundamental of Python

M.Sc. Riyadh Seed Agid

Salahaddin University – Erbil

riyadh.agid@su.edu.krd

Basic Math

Python is a basic calculator out of the box. Here we consider the most basic mathematical operations: addition, subtraction, multiplication, division and exponentiation. We use the function: print to get the output. For now we consider integers and float numbers. An integer is a plain number like 0, 10, or -2345. A float number has a decimal in it. The following are all floats: 1.0, -9., and 3.56.

Note the trailing zero is not required, although it is good style.

Example
Addition & subtraction :

```
>>> print 2 + 4
6                          #result

>>> print 8.1 -5
3.0999999999999996      # result
```

Basic Math

Multiplication is equally straightforward

```
>>> print 5 * 4
20

>>> print 3.1 * 2
6.2
```

Division is almost as straightforward, but we have to remember that integer division is not the same as float division. Let us consider float division first.

```
>>> print 4.0/2.0
2.0

>>> print 1.0/3.1
0.322580645161
```

Basic Math

```
#Now, consider the integer versions:
>>> print 4/2
2
>>> print 1/3
0
# The first result is probably what you expected, but the second may come as a surprise. In integer division the remainder is discarded, and the result is an integer.
```

Exponentiation is also a basic math operation that python supports directly.

```
>>>print 3.**2
9.0
>>>print 3**2
9
>>>print 2**0.5
1.41421356237
```

Other types of mathematical operations require us to import functionality from python libraries.

```
>>> x = 3.
>>> y = 2
>>> Print(x**y)          #same as 3.*3
9.0
```

Python Arithmetic Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Modulus %

```
>>> x = 5
>>> y = 2

>>> print(x % y)
1                #reminder
```

floor division //

```
x = 15
y = 2

print(x // y)
7
#the floor division // rounds the result down to the nearest whole
number
```

Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

```
>>> x = 6
>>> y = 3
>>> print(x == y)          #?
False                      # returns False because 6 is not equal to 3
```

```
>>> x = 5
>>> y = 3

>>> print(x > y)

True = # returns True because 5 is greater than 3
```

```
>>> x = 5
>>> y = 3

>>> print(x >= y)

True                  # returns True because five is greater, or equal, to 3
```

Advanced Mathematical Operators

The primary library we will consider is module: numpy, which provides many mathematical functions, statistics as well as support for linear algebra. We begin with the simplest functions.

```
>>> import numpy as np
>>> print np.sqrt(2)
1.41421356237
```

```
#Here is the exponential function.
>>> import numpy as np
>>> print np.exp(1)
2.71828182846
```

Advanced Mathematical Operators

There are two logarithmic functions commonly used, the natural log function: numpy.log and the base10 logarithm function:numpy.log10.

```
>>> import numpy as np

>>> np.log(10)

2.302585092994046
```

```
>>> print np.log10(10)
1.0
```

Advanced Mathematical Operators

Truth Testing

You can test if one value is equal to another by typing

```
>>> 4==5                    #Equal ??
```

This will return "False" as the numbers are not equal.
You can also use inequalities to test a value:

```
>>> 4<5                     #Less than
True
>>> 4<=5                    #Less than or equal to
True
>>> 4>4                     #Greater than
False
>>> 4>=4                    #Greater than or equal to
True
```

Notice that the "=" sign in greater than or equal to and less than or equal to always comes second, "=<" will not work.

Advanced Mathematical Operators

#There are two methods to raise a number x to the power n, use ** or the function pow(x,n)

```
>>> 3**2
```

np.pow(Base, power)

```
>>> np.pow( 3, 2)
```

These should both give the same answer of 9.

You can use fractional powers in exactly the same way, for example

```
>>> 9**(1./2)
>>> np.pow(9, 1./2)
>>> 9**0.5
```

These should both give an answer of 3.0.

Advanced Mathematical Operators

However,

```
>>> 9**1./2
```

will not. In this case you are calculating half of 9.

Very large or very small numbers are often written in scientific notation, ie. 2.5 x 10^6. In Python you can do this in two ways. Either using normal powers:

```
>>> 2.5*10**6
```

Or using e, which represents the "times ten to the power of" part:

```
>>> 2.5e6
```

Advanced Mathematical Operators

Python can handle trigonometry in much the same way as your calculator.

Note that by default, it works in radians! So

```
>>> np.sin(30)
```

does not give 0.5 as you might expect. Looking at the variable explorer, you see that pi is already defined so you can do

```
>>> np.sin(pi/6)
```

Note that this doesn't give exactly 0.5 but, as with the exponential calculation before, the difference is tiny and will generally not affect the results of any calculation you are doing.

Thank you