Variable Types and Basic Arithmetic

Monika Baloda

Importing Libraries Sometimes the name of the libraries are long and writing it repeatedly may consume time. To make it short we can give a short name to the library for the purpose of our code. For example, if we write *import math as m* then instead of *math.pi* we can just write *m.pi,* this may appear to be a very small point here but when code becomes long then it helps. For example, suppose we import pyplot function from matpltlib library.

```
import matplotlib.pyplot
matplotlib.pyplot.plot(x)
```

So instead of *matplotlib.pyplot.plot(x)*, we could have just have wrote *plt.plot(x),* if we would have imported the library as follows:

```
import matplotlib.pyplot as plt
```

Variables and Types

Python is a user friendly language, we need not to declare a variable type before we start. The language automatically assigns it to appropriate data types. For example if we write *a=1*, it will assign *a* to be an *int* type. But we write *a="1",* then it will assign *a* to be a string data type. In this part, we cover the type of data used in python:

1. *int* : this datatype stores integer values i.e. 1,2,3,4,6, -4, -199, …
2. *float* : this data type stores numbers with fractional values as well including integers. e.g. 1.23, 0.03, -8.92,... Python even supports complex numbers.
3. *boolean* : This data type takes only two values: true and false. This is often useful in checking conditional or logical statements.
4. *string* : String data type allows characters, texts, even numbers in it. We cannot do arithmetic on the numbers stored as string. We define it with quotes i.e. *name="Monika",* here *name* is a string type variable with its value being *Monika*. One can use single quotes in place of double quotes here, it doesn't matter.

Casting of Variable types:

We can actually convert the data type into each other. For example if numbers are stored in strings, as described above, we can write *int("2")* this will give us an *int* type number. We can similarly convert appropriate integers to booleans, if we write *bool(0)* it will give us a boolean with *false* value stored in it.

Basic Arithmetic:

Similar to what we do in pen and paper math, we can do 2+2 arithmetic in python. The plus sign here is called operator. The multiplication operator is * and not *x.* The division operator is /. The equal to operator is == and not just single =, the single = is called assignment operator i.e. it sets value given on the right hand side to the left hand side.

While the == signs checks whether the left hand side is equal to the right hand side and gives output in terms of boolean format.
I list down all the basic operators here:

1. + : addition
2. - : subtraction
3. * : multiplication
4. / : division
5. % : gives the remainder when we divide the two integers e.g. 5%2 is equal to 1
6. // : floor division discards the fractional part e.g. 5//2 is equal to 2, while 5/2 is 2.5
7. ** : this gives raised to the power. e.g. 5**2 is equal to 25 i.e. 5 multiplied two times
8. == equals: `5 == 5` yields `True`
9. != : does not equal: `5 != 5` yields `False`
10. `>` greater than: `5 > 4` yields `True`
11. `>=` greater than or equal: `5 >= 5` yields `True`. Similarly, we have `<` and `<=`.

Dynamic types

Unlike some languages, the variable type can change dynamically in python. All the languages will run 1+1 but python can even run *1+ True,* the output will be 2. What just happened here? Python converts the boolean True into integer and gives it a value equal to one. Similarly if we write *1 > True* then the output will be a boolean type *false*. So the output type depends on the operator we are using.

String Functions:

It is easy to play with strings in python. We directly add two strings. string*2 repeats the string two times. *.upper()* changes the string to upper case letters, similarly *.lower()* changes the string to a lower case string.