1. Python fundamentals and data handling

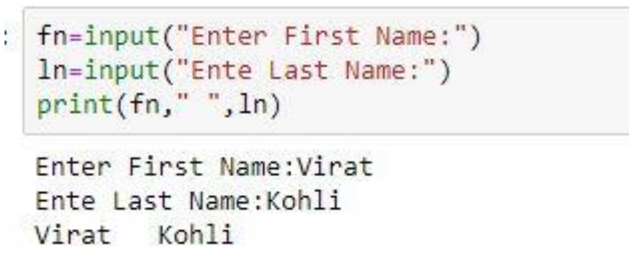
A program has important parts such as variables, statements, expressions, data types, input and output related functions, etc. In this post, you will learn about python programming fundamentals. .

Expressions

A small set of variables, operators, and value is known as expression. It seems like every line of a program can be an expression.

input() function

This function is used accept value from user. Let's look in the following:



```
fn=input("Enter First Name:")
ln=input("Ente Last Name:")
print(fn," ",ln)
```

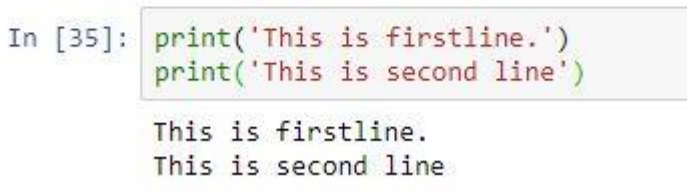
```
Enter First Name:Virat
Ente Last Name:Kohli
Virat   Kohli
```

After understanding how to take input from the user for Python fundamentals class 11 notes understand how you can print the output.

print() function

As I have used print() function in the above codes. It allows generating output on the screen. The syntax of the print() function is as follows:
print("text",[sep=' ' or end='character'])

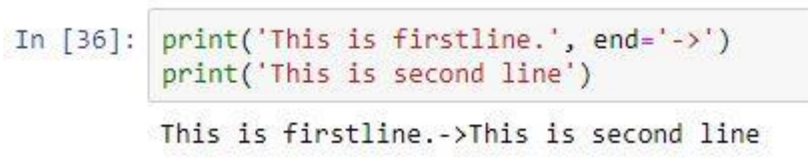
Basically python print() a new line when the print function is used in the program.



```
In [35]: print('This is firstline.')
         print('This is second line')
```

```
This is firstline.
This is second line
```

Sometimes users want to end a line with some specific characters. In this case, the 'end' option is used, have a look:



```
In [36]: print('This is firstline.', end='->')
         print('This is second line')

         This is firstline.->This is second line
```

Now you are familiar with how to accept input and print the output. The next section of Python fundamentals class 11 notes talk about variables.

Variables

Variables are storage names with specific data types, used to store a value for use in the future. It holds a value of any type such as numeric values, letters, or any other value. The value of the variable can be changed at execution time by the user and it keeps changing as per the need of the program.

A variable needs to declare first then it will assign a value or use in input statement and finally hold output or final result. The value of a variable manipulates any time in a program.

When you are going to declare any variable you need to learn about the rules to define them. The next section talk of Python fundamentals class 11 notes talks about the same.

Rules for variable naming

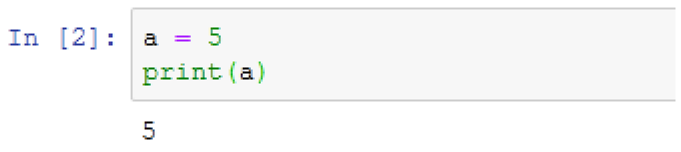
Variables are similar to identifiers. Follow the identifiers naming rules while declaring variables.

Their rules are as following:

- It must start with an alphabet
- It doesn't contain any space or special character or symbols
- A keyword should not be used as variables
- It should be short and simple

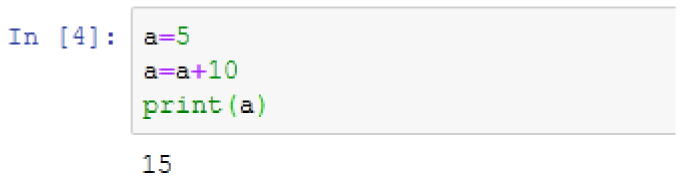
Read about print() function

Now let’s look at this code:



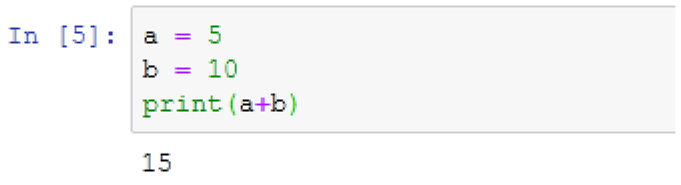
```
In [2]: a = 5
        print(a)

        5
```



```
In [4]: a=5
        a=a+10
        print(a)

        15
```



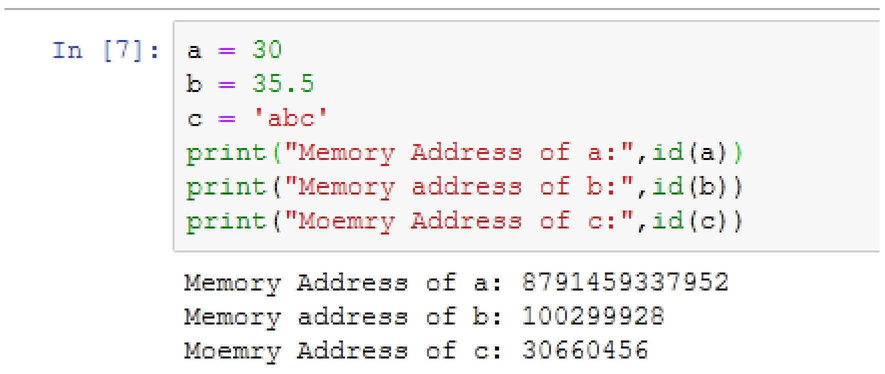
```
In [5]: a = 5
        b = 10
        print(a+b)

        15
```

In the first code variable, a is assigned value 5. In python, when the value is assigned to the variable, the data type of a variable is determined by python itself. So here 5 is an integer. In the second code variable, a is declared with value 5, then value 10 is added in a and then displayed using print() function. In the last code, two different values assigned to two different variables a and b, 5 and 10 respectively, then added both of them along with print() functions.

There are three main properties associated with a variable:

1. **Address:** Memory location address in the memory cell. To know the address of memory location python offers id() function, that accepts a variable name as parameter.

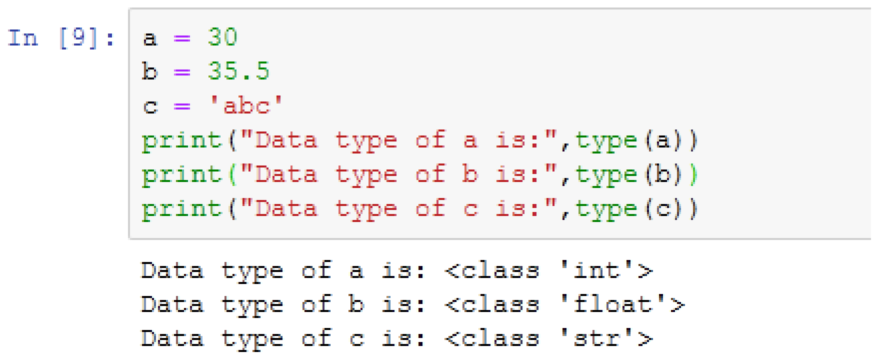


```
In [7]: a = 30
        b = 35.5
        c = 'abc'
        print("Memory Address of a:",id(a))
        print("Memory address of b:",id(b))
        print("Moemry Address of c:",id(c))

        Memory Address of a: 8791459337952
        Memory address of b: 100299928
        Moemry Address of c: 30660456
```

Displaying memory address through id() function

2. **Value:** Assigned by the programmer or changed by the statements in a program.
3. **Datatype**: The type of data such as number, letters or string, etc. To print datatype of a variable, python offers type() function.



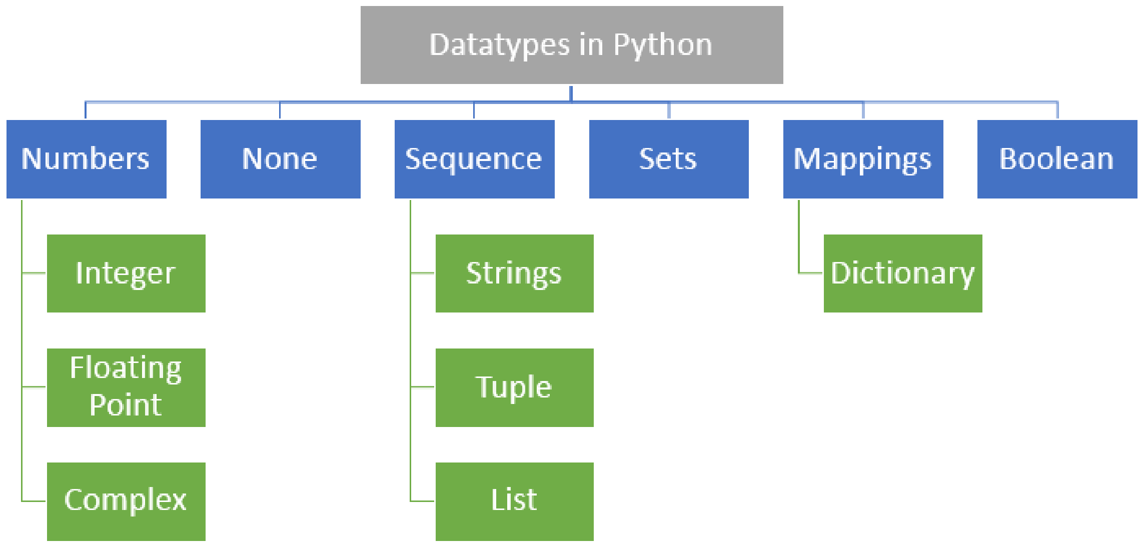
```
In [9]: a = 30
        b = 35.5
        c = 'abc'
        print("Data type of a is:",type(a))
        print("Data type of b is:",type(b))
        print("Data type of c is:",type(c))

        Data type of a is: <class 'int'>
        Data type of b is: <class 'float'>
        Data type of c is: <class 'str'>
```

Displaying data type of variable

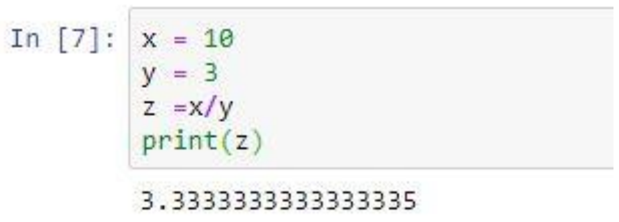
The next section of Python fundamentals class 11 notes talks about data types in python.

Datatypes in python



Python DataTypes

Numbers: It holds numeric values in a program. Python allows following built-in data types for numbers:
Integer: A Numeric value without decimal places is considered as an integer. The inbuilt class 'int' represent these type of numbers. Integers can be of any length in python. Ex. : 12345, -45465, 234, -456, 10, 0
Float: Real numbers having decimal or floating points are considered are float. Ex.: 567.90, 234.90, 3456.898989 etc. Python prints a large value post decimal places by default. Have a look on the following **code:**

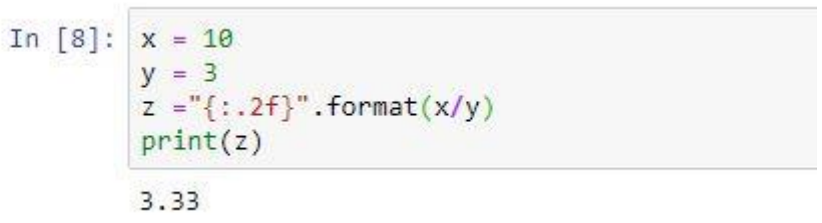


```
In [7]: x = 10
        y = 3
        z =x/y
        print(z)

        3.3333333333333335
```

Using float in program

To print desired digits in decimal value post point/dot, use str.format() function. Suppose I want to print 2 decimal place value post point/dot in the above result, I have done in the following manner:

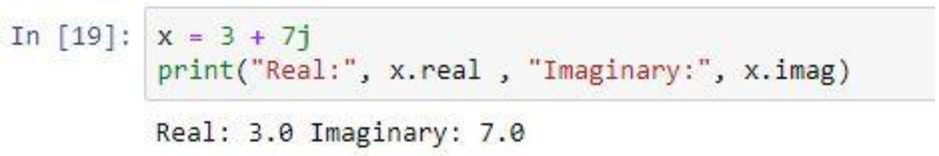


```
In [8]: x = 10
        y = 3
        z ="{:.2f}".format(x/y)
        print(z)

        3.33
```

Result in two decimal places value after a point

Complex Number: It has two parts: i) Real Number and ii) Imaginary Part. It is available in the form of 'x + yJ' or 'x + yj', where x is a float number (real number), yJ is the imaginary part, small letter j indicates the square root of an imaginary number -1. Example:



```
In [19]: x = 3 + 7j
         print("Real:", x.real , "Imaginary:", x.imag)

         Real: 3.0 Imaginary: 7.0
```

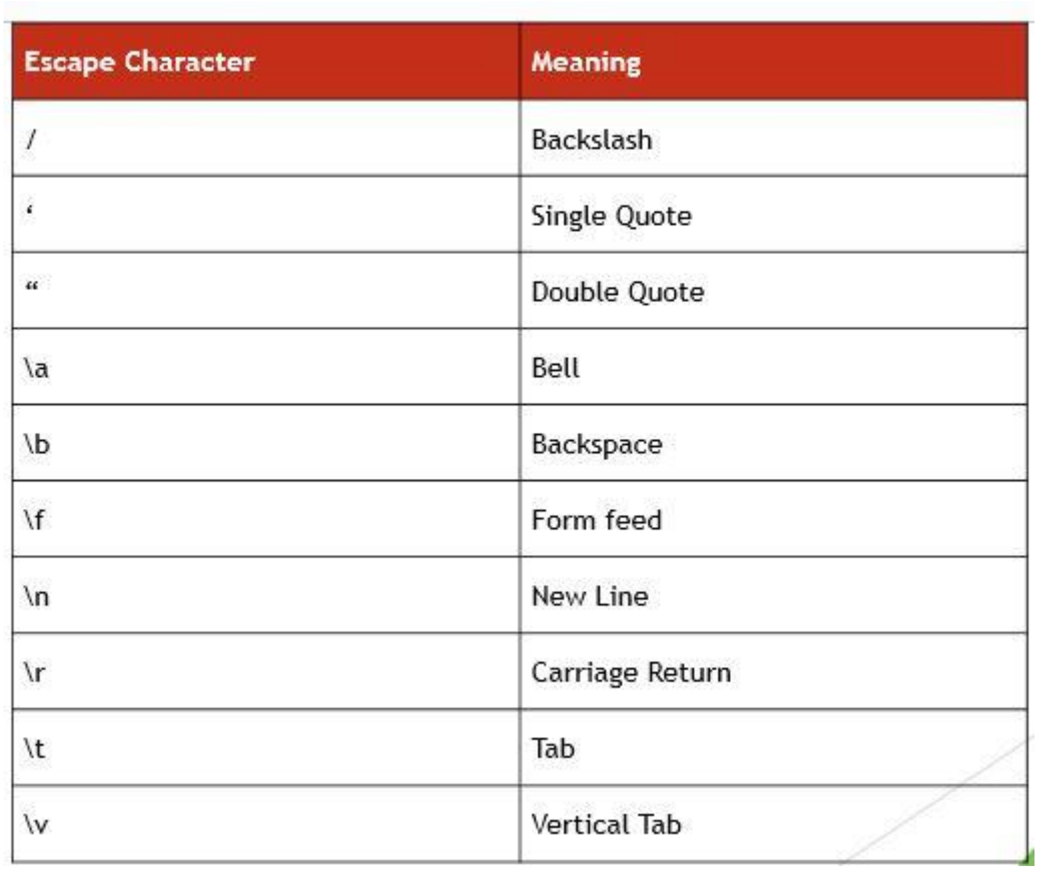
Use of complex number in python

None: None is a special data type of python. When no other data type is required to accepted then None is given as a value. It displays nothing when a variable is assigned to None.

After this in python fundamentals class 11 notes, the following data types can be used.

Sequence: It accepts values as in a sequence or specific patters. Python offers the following sequence data types:
str (String): It is a sequence of characters that is a combination of letters, numbers, and special symbols. A string is enclosed with a quotation. It can be enclosed with single, double, or triple quotes. Single line text is enclosed either with single or double quotes whereas multi-line text is enclosed with triple quotes.

Python also offers a special escape character sequence to print some non-graphic character as following:



Escape Character	Meaning
/	Backslash
‘	Single Quote
“	Double Quote
\a	Bell
\b	Backspace
\f	Form feed
\n	New Line
\r	Carriage Return
\t	Tab
\v	Vertical Tab

Escape character sequence in Python

Boolean: Boolean holds either True or False value. Any statement that has either True or False result, has a boolean data type.

Datatypes like tuples, list, dictionary, sets will be explained later in specific posts.

So now you are familiar with data types you need to assign a value for your variables. In the next section, we will talk about the variable assignment in Python fundamentals class 11 notes.

Variable Assignment

The assignment operator (=) is used to assign a new value to a variable. It takes a general form like L-value = R-value where
L-value is always a variable
R-value is a value assigned to a variable, R-value can be a value or expression
Ex. x = 3, where L-value is x and R-value is 3.

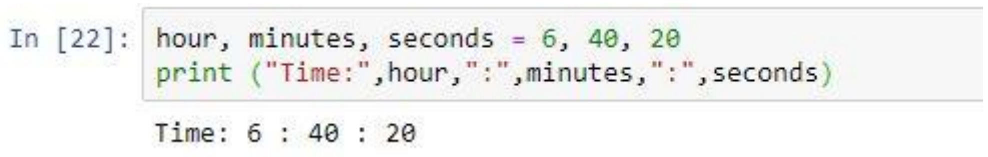
Python allows multiple assignments in a statement. The next section of Python fundamentals class 11 notes focuses on multiple assignments.

Multiple Assignment

Python allows multiple variable assignments in one line. There are two ways to assign multiple values:

Assign multiple values to multiple variables

The values of the variable assigned by separating them with commas. Ex.:



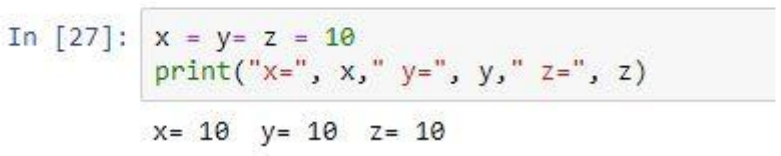
```
In [22]: hour, minutes, seconds = 6, 40, 20
         print ("Time:",hour,":",minutes,":",seconds)

         Time: 6 : 40 : 20
```

Multiple variable assignments

Assign the same value to multiple variables

The same value assigned by using an assignment operator (=) multiple times.



```
In [27]: x = y= z = 10
         print("x=", x," y=", y," z=", z)

         x= 10  y= 10  z= 10
```

Assign a same value to multiple variable

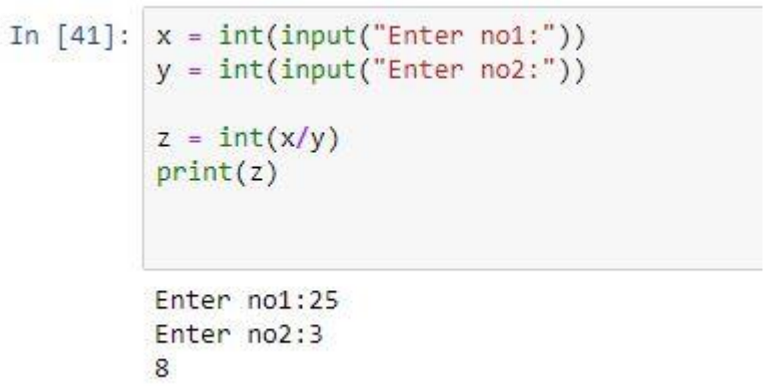
In the next section of python fundamentals class 11 notes, we will discuss data type conversion.

Datatype Conversion

It is also known as typecasting. There are two types of type conversion.

Explicit

This type of typecasting is done by the program to change the result value data type. It uses the target data type as a function and a variable is passed to convert it. Consider this example:



```
In [41]: x = int(input("Enter no1:"))
         y = int(input("Enter no2:"))

         z = int(x/y)
         print(z)

         Enter no1:25
         Enter no2:3
         8
```

Implicit

It is by default conversion. As in the above program, the division operator converts the result into float implicitly.