Python provides numerous **built-in functions** that are readily available to us at the Python prompt.

Some of the functions like input() and print() are widely used for standard input and output operations respectively. Let us see the output section first.

## Python Output Using print() function

We use the print() function to output data to the standard output device (screen). We can also [output data to a file](https://www.programiz.com/python-programming/file-operation), but this will be discussed later.

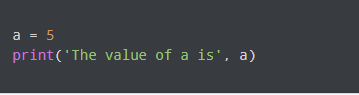
An example of its use is given below.



**Output**



Another example:



**Output**



In the second print() statement, we can notice that space was added between the string and the value of variable a. This is by default, but we can change it.

The actual syntax of the print() function is:

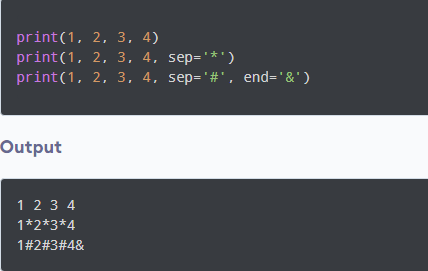


Here, objects is the value(s) to be printed.

The sep separator is used between the values. It defaults into a space character.

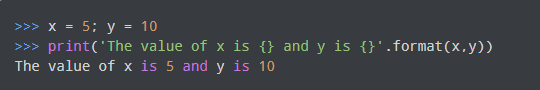
After all values are printed, end is printed. It defaults into a new line.

The file is the object where the values are printed and its default value is sys.stdout (screen). Here is an example to illustrate this.

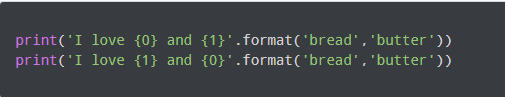


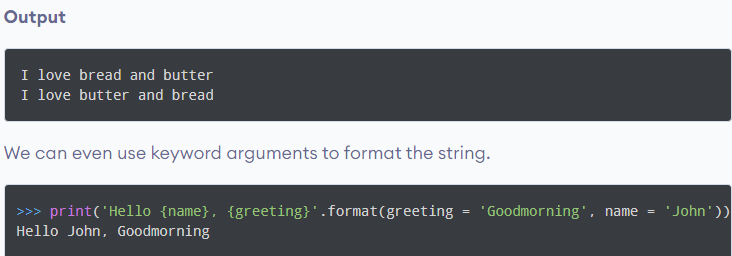
## Output formatting

Sometimes we would like to format our output to make it look attractive. This can be done by using the str.format() method. This method is visible to any string object.

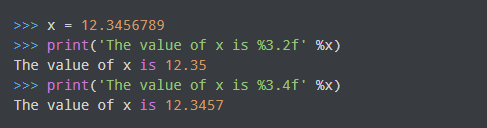


Here, the curly braces {} are used as placeholders. We can specify the order in which they are printed by using numbers (tuple index).





We can also format strings like the old **sprintf()** style used in **C programming language**. We use the % operator to accomplish this.



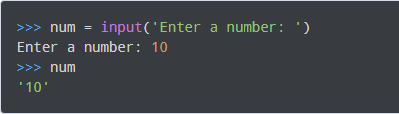
## Python Input

Up until now, our programs were static. The value of variables was defined or hard coded into the source code.

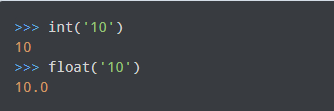
To allow flexibility, we might want to take the input from the user. In Python, we have the input() function to allow this. The syntax for input() is:

**input([prompt])**

where prompt is the string we wish to display on the screen. It is optional.



Here, we can see that the entered value 10 is a string, not a number. To convert this into a number we can use **int() or float()** functions.



This same operation can be performed using the eval() function. But eval takes it further. It can evaluate even expressions, provided the input is a string

