Input and Output Operation In Python

INPUT AND OUTPUT OPERATIONS

Hello there, everyone. I'm impressed by your determination to continue to excel in the Python programming language.

The Python language's input and output operations are covered in this module.
We've always assigned values to variables on our own, but now you can interact with your program by providing input during the execution. I'm sure it will make you feel powerful. You can also format the output to your requirements.

So let's quickly get started.
Python has a lot of built-in functions that we may use right from the Python prompt. Some of the functions, such as input() and print(), are commonly used for common input and output operations.

Let's see what the input function is?

At some point in the software, a developer may want to encourage user input. Python has a function called input().

As a result, if the user enters 3 as the value of the variable 'a,' the value of the variable 'a' will be 3.

The programmer, on the other hand, has not specified what is an in this example. The user must be aware of what is expected in the program as input. You can specify the type of input you want by providing a display string to the input function. For example, in the code, you can use a display string to show the context, such as "entering your name."

You'll now learn a few more advanced level input functions.

By default, Python treats all input as a string data type. We must convert the input using explicit type conversion to convert it to any other data type. For example, to convert an input to an int or a float, you need to use the int() and float() methods, respectively.

Let's take a look at this program. Let's say the user entered the number 30. In this situation, you must convert the user input to an integer using the int type data conversion. You may now increase the input by 1 to get the output 31.

Let's look at some more examples to help clarify things. num is a variable in the first code. You can indicate the type of user input you prefer by using a display string.

The input is treated as a string in the second set of code, thus the type conversion function int() is used to convert it to an integer. You can also use the float type conversion function to convert the input into a float.

OUTPUT - PRINT FUNCTION

Welcome back to the video, you will now learn more about output operations in the Python language.

Let's start with the print output function and how to change or format it.

The print() method is the easiest way to generate output. The print() function is used to display the results to a standard output device like the screen.

Now you will learn the syntax of the standard output function.

print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)

Print function has 5 parameters such as objects, separator, end character, file, and flush.

Let's take a closer look at each one with an example.

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

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

After all values are printed, the specified end symbol is printed. If we don't specify any symbol, by default it creates 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.

The flush() method in Python file handling clears the internal buffer of the file. The flush makes sure that any output that is buffered goes to the destination.

For example, a= hello world, print a, the output will be a string which is hello world.

You can alter the output by using the separator function. The separator or sep is used between the values. It defaults into a space character.

Note that if you don't specify a separator, the default separator is space.

If you specify commas as a separator, for example, the result will include commas as a separator.

The output will be shown with * as a separator if you use * in the sep parameter.

You can use anything in quotes as a sep function such as @, ! or & as separators.

Similar to the separator function you can define how the output should end. The end functions specify what to print at the end. Remember that it defaults into a new line if nothing is specified.

For example here asterisks are used as separators and the output ends with & symbol by using the end function.

Now the File (file) is the object where the values are printed and its default is sys.stdout (screen).

This module comes to a close; I'm sure you'll want to explore around with these functions to see how they may be used to improve the output.

OUTPUT FORMATTING

Excited to see here! I hope you're enjoying the output function, and in this module, you will learn about output formatting functions again.

Sometimes you may want to format your output to make it more appealing.

The str.format() method can be used to perform this.

String formatting is the process of dynamically putting things during execution into a string and displaying it. Curly brackets are used to do this.

For instance, suppose your age is 20.

To make the result more readable, you can specify where you wish the variable to appear in a string.

As a result, the output will show that I am twenty years old.

Unlike the ordinary print function, you can specify where the value of a variable should be displayed.

Let's look at a couple more examples to help you understand.

Here x and y are two variables with values assigned.

The string formatting functions can be used to describe the location of values in a string, such as "the value of x is 15 and y is 20."

Let's look at an example that is a little more advanced.

By assigning values starting at 0, you can define string formatting places.

The first string format function with a value of 0 will have coding, whereas the second string format function with a value of 1 will have programming.

In this example, the output format can be changed simply altering the string placement.

Change the values of the string format function to see the difference between the two outputs.

Similarly, even if the name and greetings variables are defined in a different order, they will appear in the correct order as described by the string formatting function.

Remember that the output is determined by the use of curly brackets.

Good morning, everyone! I admire your desire to learn more about the Python programming language.

you will continue learning about output formatting in this module, using the escape sequence and the float operation.

First, let's look at the Escape Sequence. It's a string of characters with a non-literal character interpretation (typically preceded by an escape character). A backslash is used as an escape character in most programming languages. A backslash indicates that a new line will be ignored. For example, if you write ten lines with backslash, they will all display on a single line.

A backslash followed by a n denotes a new line. This is how you start a new line.

Backslash with t is the keyboard's counterpart of the tab key.

String formatting can also be done with the percent operator.

The decimal position in a floating point number is determined using the string modulo operator (percent). Percentage x. (dot) is the syntax.

Let's see this example.

x is assigned with the value 12.3456789.

print('The value of x is %.2f' %x)

The value of x is 12.35.

The modulo operator truncates the float number to the first 2 digits after the decimal point by rounding to the nearest digit,

print('The value of x is %.4f' %x)

In this case, the modulo operator rounds the value to four decimal places.

The value of x is 12.3457

print('The value of x is %.5f' %x)

The value of x is 12.34568

In the last example, the modulo operator rounds the value to five decimal places.

SPLIT METHOD

Well, it's great to see you!!!

In this module you will learn how to use the split method and the map function in output formatting.

Let's have a look at the split method.

The split() method splits a string into a list of elements.You can specify the separator, default separator is any whitespace. Basically it creates a list by breaking the given string by the specified separator.

Syntax for the split method is this.

string.split(separator, maxsplit)

The separator splits the string as per the specified separator. If it is not provided then any white space is a separator.

maxsplit is a number, which tells us to split the string into a maximum of the provided number of times.

Example:

txt = "I Love Python Programming"

x = txt.split(" ")

print(x)

The string in this example was split into the

'I', 'Love', 'Python', 'Programming' by creating a list.

We've come a long way and now you can work with several functions and a bit more advanced code.

Let's have a look at the map function now.
Python's map() is a built-in function that allows you to process and transform all the items in an iterable without using explicit looping constructs, a technique commonly known as mapping.

When you need to apply a transformation function to each item in an iterable and transform them into a new iterable, map() comes in useful.

Look at this example program.

```
x, y, z = map(int,input("enter numbers: ").split())
print(x+y+z)
enter numbers: 1 2 3
```

We used the input function in this example to take user input, which was then separated based on space. To convert the input to an integer, use the int function. Finally, the values are assigned to the variable using the map function.

This is the order in which the compiler will compute the output.

I hope this map function is understandable and makes you proud of what you've learned so far.