Python - User Input

Provide User Input in Python

In this chapter, we will learn how Python accepts the user input from the console, and displays the output on the same console.

Every computer application should have a provision to accept input from the user when it is running. This makes the application interactive. Depending on how it is developed, an application may accept the user input in the form of text entered in the console **(sys.stdin)**, a graphical layout, or a web-based interface.

Python User Input Functions

Python provides us with two built-in functions to read the input from the keyboard.

- The input () Function
- The raw_input () Function

Python interpreter works in interactive and scripted mode. While the interactive mode is good for quick evaluations, it is less productive. For repeated execution of same set of instructions, scripted mode should be used.

Let us write a simple Python script to start with.

Open Compiler

```
#! /usr/bin/python3.11

name = "Kiran"
city = "Hyderabad"

print ("Hello My name is", name)
print ("I am from", city)
```

Save the above code as hello.py and run it from the command-line. Here's the output

```
C:\python311> python hello.py
Hello My name is Kiran
```

```
I am from Hyderabad
```

The program simply prints the values of the two variables in it. If you run the program repeatedly, the same output will be displayed every time. To use the program for another name and city, you can edit the code, change name to say "Ravi" and city to "Chennai". Every time you need to assign different value, you will have to edit the program, save and run, which is not the ideal way.

Learn **Python** in-depth with real-world projects through our **Python certification course**. Enroll and become a certified expert to boost your career.

The input() Function

Obviously, you need some mechanism to assign different value to the variable in the runtime − while the program is running. Python's **input()** function does the same job.

Following is the syntax of Python's standard library input() function.

```
>>> var = input()
```

When the interpreter encounters input() function, it waits for the user to enter data from the standard input stream (keyboard) till the Enter key is pressed. The sequence of characters may be stored in a string variable for further use.

On reading the Enter key, the program proceeds to the next statement. Let use change our program to store the user input in name and city variables.

```
#! /usr/bin/python3.11

name = input()
city = input()

print ("Hello My name is", name)
print ("I am from ", city)
```

When you run, you will find the cursor waiting for user's input. Enter values for name and city. Using the entered data, the output will be displayed.

```
Ravi
Chennai
Hello My name is Ravi
I am from Chennai
```

Now, the variables are not assigned any specific value in the program. Every time you run, different values can be input. So, your program has become truly interactive.

Inside the input() function, you may give a **prompt** text, which will appear before the cursor when you run the code.

```
#! /usr/bin/python3.11

name = input("Enter your name : ")
city = input("Enter your city : ")

print ("Hello My name is", name)
print ("I am from ", city)
```

When you run the program displays the prompt message, basically helping the user what to enter.

```
Enter your name: Praveen Rao
Enter your city: Bengaluru
Hello My name is Praveen Rao
I am from Bengaluru
```

The raw_input() Function

The **raw_input()** function works similar to **input()** function. Here only point is that this function was available in Python 2.7, and it has been renamed to **input()** in Python 3.6

Following is the syntax of the **raw_input()** function:

```
>>> var = raw_input ([prompt text])
```

Let's re-write the above program using raw_input() function:

```
#! /usr/bin/python3.11

name = raw_input("Eneter your name - ")
city = raw_input("Enter city name - ")

print ("Hello My name is", name)
print ("I am from ", city)
```

When you run, you will find the cursor waiting for user's input. Enter values for name and city. Using the entered data, the output will be displayed.

```
Eneter your name - Ravi
Enter city name - Chennai
Hello My name is Ravi
I am from Chennai
```

Taking Numeric Input in Python

Let us write a Python code that accepts width and height of a rectangle from the user and computes the area.

```
#! /usr/bin/python3.11

width = input("Enter width : ")
height = input("Enter height : ")

area = width*height
print ("Area of rectangle = ", area)
```

Run the program, and enter width and height.

```
Enter width: 20
Enter height: 30
Traceback (most recent call last):
  File "C:\Python311\var1.py", line 5, in <module>
   area = width*height
TypeError: can't multiply sequence by non-int of type 'str'
```

Why do you get a **TypeError** here? The reason is, Python always read the user input as a string. Hence, width="20" and height="30" are the strings and obviously you cannot perform multiplication of two strings.

To overcome this problem, we shall use **int()**, another built-in function from Python's standard library. It converts a string object to an integer.

To accept an integer input from the user, read the input in a string, and type cast it to integer with int() function −

```
w = input("Enter width : ")
width = int(w)
```

```
h = input("Enter height : ")
height = int(h)
```

You can combine the input and type cast statements in one –

```
#! /usr/bin/python3.11

width = int(input("Enter width : "))
height = int(input("Enter height : "))

area = width*height
print ("Area of rectangle = ", area)
```

Now you can input any integer value to the two variables in the program –

```
Enter width: 20
Enter height: 30
Area of rectangle = 600
```

Python's **float()** function converts a string into a float object. The following program accepts the user input and parses it to a float variable – rate, and computes the interest on an amount which is also input by the user.

```
#! /usr/bin/python3.11

amount = float(input("Enter Amount : "))
rate = float(input("Enter rate of interest : "))

interest = amount*rate/100
print ("Amount: ", amount, "Interest: ", interest)
```

The program ask user to enter amount and rate; and displays the result as follows –

```
Enter Amount: 12500
Enter rate of interest: 6.5
Amount: 12500.0 Interest: 812.5
```

The print() Function

Python's print() function is a built-in function. It is the most frequently used function, that displays value of Python expression given in parenthesis, on Python's console, or standard output **(sys.stdout)**.

</> Open Compiler

```
print ("Hello World ")
```

Any number of Python expressions can be there inside the parenthesis. They must be separated by comma symbol. Each item in the list may be any Python object, or a valid Python expression.

```
#! /usr/bin/python3.11

a = "Hello World"
b = 100
c = 25.50
d = 5+6j
print ("Message: a)
print (b, c, b-c)
print(pow(100, 0.5), pow(c,2))
```

The first call to print() displays a string literal and a string variable. The second prints value of two variables and their subtraction. The **pow()** function computes the square root of a number and square value of a variable.

```
Message Hello World
100 25.5 74.5
10.0 650.25
```

If there are multiple comma separated objects in the print() function's parenthesis, the values are separated by a white space " ". To use any other character as a separator, define a **sep** parameter for the print() function. This parameter should follow the list of expressions to be printed.

In the following output of print() function, the variables are separated by comma.

</> Open Compiler

```
#! /usr/bin/python3.11

city="Hyderabad"
state="Telangana"
country="India"
print(city, state, country, sep=',')
```

The effect of sep=',' can be seen in the result –

```
Hyderabad,Telangana,India
```

The print() function issues a newline character ('\n') at the end, by default. As a result, the output of the next print() statement appears in the next line of the console.

Open Compiler

```
city="Hyderabad"
state="Telangana"
print("City:", city)
print("State:", state)
```

Two lines are displayed as the output –

```
City: Hyderabad
State: Telangana
```

To make these two lines appear in the same line, define **end** parameter in the first print() function and set it to a whitespace string " ".

Open Compiler

```
city="Hyderabad"
state="Telangana"
country="India"

print("City:", city, end=" ")
print("State:", state)
```

Output of both the print() functions appear in continuation.

```
City: Hyderabad State: Telangana
```