Python

Topics: Input and Output

Input and Output

Tutorial

Python Input and Output

With this topic, we begin our series of Python Practice tutorials. Every tutorial describes a specific topic with examples. A problem statement at the end of each tutorial will assess your understanding.

Introduction

Like all high-level languages, Python is easy to read, takes less time to write, and is portable. This versatile programming language has two versions: Python 2 and Python 3. Wiki says: Python 2.x is legacy, Python 3.x is the present and future of the language. That is, Python 2 is no longer in development and all new features will be added in Python 3. Note that, keeping this in mind, the code examples in this tutorial are in Python 3. Wherever Python 2.x code is shown, it will be highlighted.

Execution

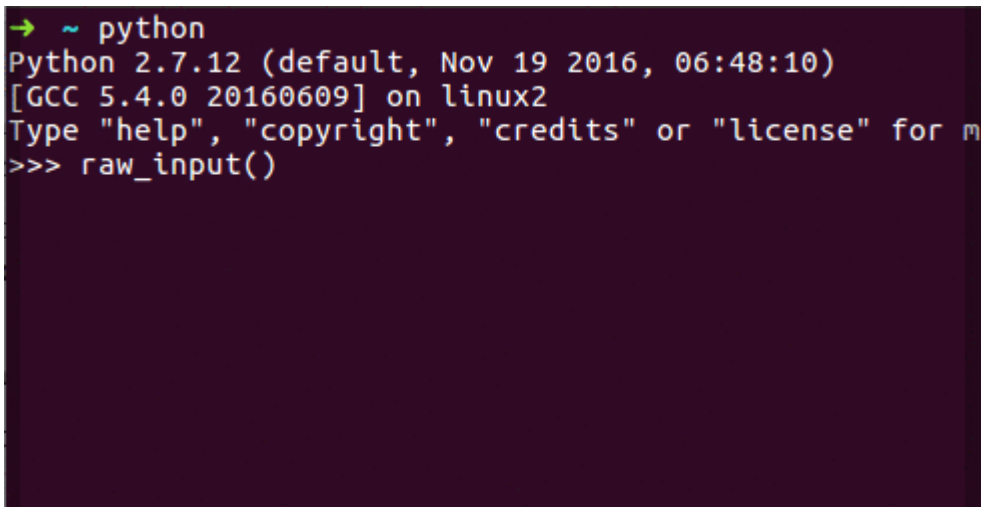
Python executes code top to bottom, when written in the correct syntax. To execute the code in our python tutorials, you will need to install python in your machine as a prerequisite. A small description on how to install Python and get the interpreter running is given here. Once the interpreter is running you can start typing in commands to get the result.

?

Input using the `input( )` function

A function is defined as a block of organized, reusable code used to perform a single, related action. Python has many built-in functions; you can also create your own. Python has an input function which lets you ask a user for some text input. You call this function to tell the program to stop and wait for the user to key in the data. In Python 2, you have a built-in function `raw_input()`, whereas in Python 3, you have `input()`. The program will resume once the user presses the ENTER or RETURN key. Look at this example to get input from the keyboard using Python 2 in the interactive mode. Your output is displayed in quotes once you hit the ENTER key.

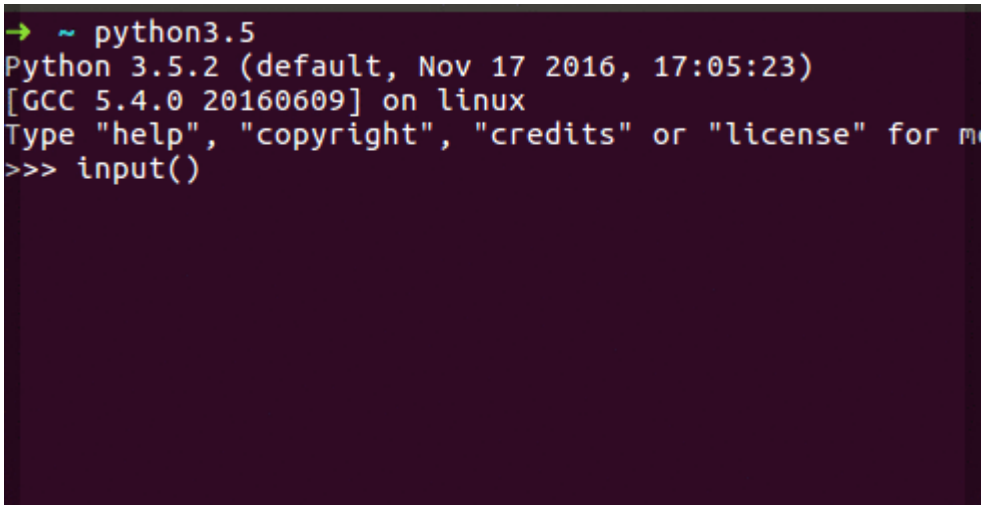
```
>>>raw_input()
I am learning at hackerearth    (This is where you type in)
'I am learning at hackerearth' (The interpreter showing you how the input is
captured.)
```



In Python 3.x, you need to use input().

```
>>> input()
I am learning at hackerearth.
'I am learning at hackerearth.'
```

?



You can always tell your user what to input by printing a prompt. There is no difference between `input` in Python 3 and `raw_input` in Python 2 except for the keywords.

Output using the `print()` function

To output your data to the screen, use the `print()` function. You can write `print(argument)` and this will print the `argument` in the next line when you press the `ENTER` key.

> **Definitions to remember:** An argument is a value you pass to a function when calling it. A value is a letter or a number. A variable is a name that refers to a value. It begins with a letter. An assignment statement creates new variables and gives them values.

This syntax is valid in both Python 3.x and Python 2.x. For example, if your data is "Guido," you can put "Guido" inside the parentheses `( )` after `print`.

```
>>> print("Guido")
Guido
```

More on using input

To capture the input in your program, you will need a variable. A variable is a container to hold data. (You will learn more on variables in a later tutorial.) You can take the input and assign it to a variable. This is done using the = operator before the input keyword and then putting the variable name before the = operator. For example, when you give a sentence "generic input" as the input, this gets assigned to a variable, say, `my_var`. You can then print the value stored in `my_var`. Let us understand this with the following example:

```
>>> # take an input and assign it to a variable
```

```
>>> # take an input and assign it to a variable
>>> beautiful_number = input() # The data you key in the next line which is 6 will be assigned to beautiful_number
6
>>> print(beautiful_number) # the next line will print the value in beautiful_number after you press enter or return
'6'
```

Give a helpful hint during the prompt

It is often a good idea to tell the user what to input. You can do this by putting the hint in quotes inside the input parentheses. The hint will come in the next line and will wait for the user input. You can then type the input and when you hit the ENTER key, it will capture the input. In this example, "tell me a beautiful number" is the hint. This gets printed in the next line when asking for the input. If you type 6, this will be assigned to the variable `beautiful_number` which we can print later.

```
>>> beautiful_number = input("tell me a beautiful number ")
tell me a beautiful number 6
>>> print(beautiful_number)
'6'
```

More on using print

Say you want to print a specific string (a sequence of characters such as letters, punctuation marks, numbers, and letters) N number of times. The (asterisk) * operator performs repetition on strings. You can print "5" six times. Inside the print parentheses, put â€œ5â€ followed by * and the number of times you want â€œ5â€ repeated.

```
>>> print("5"*6)
555555
```

You can separate the output using the comma delimiter. By default, this adds a space between the output items. For example, the sequence of numbers `5,6,7` separated by comma `,` gets printed with a space between one number and the next.

```
>>> print(5,6,7)
5 6 7
```

To change the output to what you want, use the keyword arguments sep and end to print ( ). When separating the output with a comma delimiter, you can also define the separation format using the `sep` keyword.

```
>>> print('LOVE', 30, 82.2)
LOVE 30 82.2
>>> print('LOVE', 30, 82.2, sep=',')
'LOVE', 30, 82.2
```

By default, `print` goes to a new line at the end. You can change this by using the keyword `end` as shown in the example below.

```
>>> print('LOVE', 30, 82.2, sep=',', end='!!\n')
'LOVE', 30, 82.2!!
```

For example, you can print the letters in the word "python" and all the letters will come in a new line.

```
>>> for i in "python":
...     print(i)
...
p
y
t
h
o
n
```

You can change this default implementation. You can have a colon `:` between the letters instead of a new line.

```
>>> for i in "python":
...     print(i, end=":")
...
p:y:t:h:o:n:
```

Printing the result of a calculation

Say you can assign the number `7` to a variable `population` and if you write the logic `population * 7` inside the parentheses of `print`, it will just do the calculation up front and print the result.

?

```
>>> population = 7
>>> print("Population in 2050: ", population * 1.28) # making the calculation in place
Population in 2050: 8.96
```

Contributed by: Joydeep Bhattacharjee

View all comments