# Python - entrada do usuário

## Fornecer entrada do usuário em Python

Neste capítulo, aprenderemos como Python aceita a entrada do usuário no console e exibe a saída no mesmo console.

Cada aplicativo de computador deve ter uma capacidade para aceitar entradas do usuário quando estiver em execução. Isso torna o aplicativo interativo. Dependendo de como é desenvolvido, um aplicativo pode aceitar a entrada do usuário na forma de texto inserido no console **(sys.stdin)** , um layout gráfico ou uma interface baseada na web.

## Funções de entrada do usuário Python

Python nos fornece duas funções integradas para ler a entrada do teclado.

- A função de entrada ()
- A função raw_input()

O interpretador Python funciona em modo interativo e com script. Embora o modo interativo seja bom para avaliações rápidas, é menos produtivo. Para execução repetida do mesmo conjunto de instruções, o modo de script deve ser usado.

Vamos escrever um script Python simples para começar.

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

name = "Kiran"
city = "Hyderabad"

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

Salve o código acima como hello.py e execute-o na linha de comando. Aqui está a saída

```
C:\python311> python hello.py
Hello My name is Kiran
I am from Hyderabad
```

O programa simplesmente imprime os valores das duas variáveis nele contidas. Se você executar o programa repetidamente, a mesma saída será exibida todas as vezes. Para usar o programa para outro nome e cidade, você pode editar o código, alterar o nome para "Ravi" e a cidade para "Chennai". Cada vez que precisar atribuir um valor diferente, você terá que editar o programa, salvar e executar, o que não é o caminho ideal.

## A função input()

Obviamente, você precisa de algum mecanismo para atribuir valores diferentes à variável em tempo de execução - enquanto o programa está em execução. **A função input()** do Python faz o mesmo trabalho.

A seguir está a sintaxe da função input() da biblioteca padrão do Python.

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

Quando o intérprete encontra a função input(), ele espera que o usuário insira os dados do fluxo de entrada padrão (teclado) até que a tecla Enter seja pressionada. A sequência de caracteres pode ser armazenada em uma variável de string para uso posterior.

Ao ler a tecla Enter, o programa prossegue para a próxima instrução. Vamos alterar nosso programa para armazenar a entrada do usuário nas variáveis de nome e cidade.

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

name = input()
city = input()

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

Ao executar, você encontrará o cursor aguardando a entrada do usuário. Insira valores para nome e cidade. Usando os dados inseridos, a saída será exibida.

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

Agora, as variáveis não recebem nenhum valor específico no programa. Cada vez que você executa, valores diferentes podem ser inseridos. Então, seu programa se tornou verdadeiramente interativo.

Dentro da função input(), você pode fornecer um texto **de prompt** , que aparecerá antes do cursor quando você executar o código.

```
#! /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)
```

Ao ser executado o programa exibe a mensagem de prompt, basicamente ajudando o usuário o que inserir.

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

## A função raw_input()

A função **raw_input()** funciona de forma semelhante à função **input()** . O único ponto aqui é que esta função estava disponível no Python 2.7 e foi renomeada para **input()** no Python 3.6

A seguir está a sintaxe da função **raw_input()** :

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

Vamos reescrever o programa acima usando a função raw_input():

```
#! /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)
```

Ao executar, você encontrará o cursor aguardando a entrada do usuário. Insira valores para nome e cidade. Usando os dados inseridos, a saída será exibida.

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



## Obtendo entrada numérica em Python

Vamos escrever um código Python que aceite a largura e a altura de um retângulo do usuário e calcule a área.

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

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

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

Execute o programa e insira a largura e a altura.

```
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'
```

Por que você recebe um **TypeError** aqui? A razão é que o Python sempre lê a entrada do usuário como uma string. Portanto, width="20" e height="30" são as strings e obviamente você não pode realizar a multiplicação de duas strings.

Para superar esse problema, usaremos **int()** , outra função integrada da biblioteca padrão do Python. Ele converte um objeto string em um número inteiro.

Para aceitar uma entrada inteira do usuário, leia a entrada em uma string e digite convertê-la em inteiro com a função int() -

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

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

Você pode combinar as instruções input e type cast em um -

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

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

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

Agora você pode inserir qualquer valor inteiro nas duas variáveis do programa -

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

**A função float()** do Python converte uma string em um objeto float. O programa a seguir aceita a entrada do usuário e a analisa em uma taxa variável flutuante e calcula os juros sobre um valor que também é inserido pelo usuário.

```
#! /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)
```

O programa pede ao usuário para inserir o valor e a taxa; e exibe o resultado da seguinte forma -

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



## A função print()

A função print() do Python é uma função integrada. É a função usada com mais frequência, que exibe o valor da expressão Python dada entre parênteses, no console do Python ou na saída padrão **(sys.stdout)** .

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

Qualquer número de expressões Python pode estar entre parênteses. Eles devem ser separados pelo símbolo de vírgula. Cada item da lista pode ser qualquer objeto Python ou uma expressão Python válida.

```
#! /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))
```

A primeira chamada para print() exibe uma string literal e uma variável de string. A segunda imprime o valor de duas variáveis e sua subtração. A função **pow()** calcula a raiz quadrada de um número e o valor quadrado de uma variável.

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

Se houver vários objetos separados por vírgula nos parênteses da função print(), os valores serão separados por um espaço em branco " ". Para usar qualquer outro caractere como separador, defina um parâmetro **sep** para a função print(). Este parâmetro deve seguir a lista de expressões a serem impressas.

Na seguinte saída da função print(), as variáveis são separadas por vírgula.

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

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

O efeito de sep=',' pode ser visto no resultado -

```
Hyderabad,Telangana,India
```

A função print() emite um caractere de nova linha ('\n') no final, por padrão. Como resultado, a saída da próxima instrução print() aparece na próxima linha do console.

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

Duas linhas são exibidas como saída -

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

Para fazer com que essas duas linhas apareçam na mesma linha, defina o parâmetro **end** na primeira função print() e defina-o como uma string de espaço em branco " ".

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

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

A saída de ambas as funções print() aparece em continuação.

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