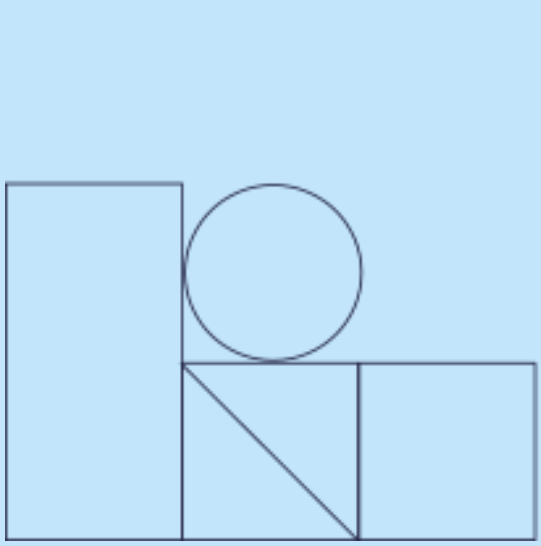
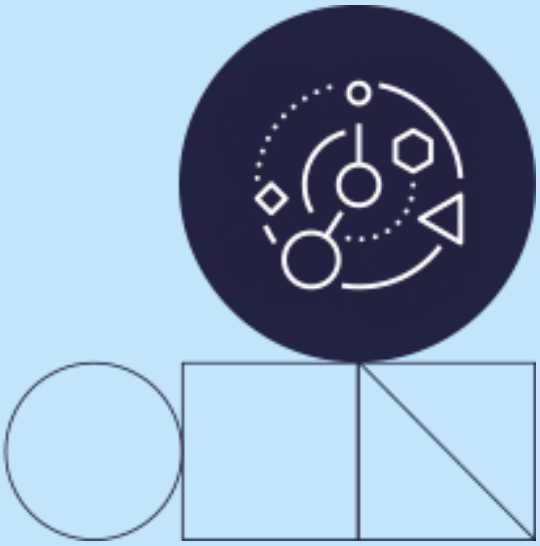
IBM SkillsBuild

IBM

# Noções básicas e sintaxe de Python

## Entrada e saída de dados em Python





# Índice

# Introdução

A seguir, analisaremos primeiro diferentes formas de receber informações dos usuários e, em seguida, mostraremos a saída.

# Como receber informações do usuário em Python

Às vezes, podemos precisar que o usuário introduza um valor por console. Para fazer isso, Python fornece uma função input().

Sintaxe:

```
input('prompt')
```

Onde prompt é uma cadeia opcional (uma mensagem) a ser exibida no momento do pedido de entrada.

Exemplo 1: Peça ao usuário para digitar seu nome.

```
# Entrada input del usuario
nombre = input('Introduce tu nombre: ')

# Salida
print("Hola, " + nombre)
print(type(nombre))
```

Saída:

```
Introduce tu nombre: Angel
Hola, Angel
<class 'str'>
```

**Nota**Python pega o que o usuário entra por meio de um input() como um string. Para convertê-lo para qualquer outro tipo de dado, temos que converter a entrada explicitamente. Por exemplo, para converter a entrada em int ou float temos que usar o método int() y float() respectivamente.

Exemplo 2: Peça um número e acrescente uma unidade a ele.

```
# Entrada por parte del usuario como
número entero
num = int(input('Introduce un número: '))

add = num+1

# Salida
print(add)
```

Saída:

```
Introduce un número: 34
35
```

# Como solicitar múltiplos valores ao mesmo tempo

Podemos aceitar múltiplas entradas de uma só vez, usando o método map()

```
a, b, c = map(int, input("Introduzca los
números: ").split())
print("Los números son: ", end = " ")
print(a, b, c)
```

Saída:

```
Introduce un número: 2 3 4
Los números son: 2 3 4
```

# Entrada de dados em listas, conjuntos, tuplas, etc.

No caso de List e Set, a entrada pode ser retirada do usuário de duas maneiras.

- Solicitando os elementos List/Set um a um usando os métodos append()/add().
- Usando os métodos map() e list() / set().

## Solicitação de itens de List/Set um a um

Para inserir os elementos da Lista/Set um a um, usaremos o método append() no caso das Listas, e o método add() no caso de conjuntos.

```
List = list()
Set = set()
l = int(input("Introduzca el tamaño de la lista: "))
s = int(input("Introduzca el tamaño del Set: "))
print("Introduzca los elementos de la lista:")
for i in range(0, 1):
    list.append(int(input()))
print("Introduzca los elementos del Set: ")
for i in range(0, 5):
    Set.add(int(input()))
print(list)
print(set)
```

## Usando métodos de mapa() e lista() / set()

```
List = list(map(int, input("Introduzca los elementos de la lista:").split()))
Set = set(map(int, input("Introduzca los elementos del Set: ").split()))
print(List)
print(Set)
```

Saída:

```
Introduzca los elementos de la lista: 2
Introduzca los elementos del Set: 3
[2]
{3}
```

## Entrada de dados em uma tupla

Sabemos que as tuplas são imutáveis, não há métodos disponíveis para adicionar elementos às tuplas. Para adicionar um novo elemento a um tuple, devemos primeiro converter o tupla em uma lista, depois adicionar o elemento à lista e novamente converter a lista em uma tupla.

```
T = (2, 3, 4, 5, 6)
print("Tupla inicial")
print(T)
L = list(T)
L.append(int(input("Introduzca el nuevo elemento: ")))
L = tuple(L)
print("Tupla final")
print(T)
```

Tupla inicial

(2, 3, 4, 5, 6)

Introduzca el nuevo elemento : 77

Tupla final

(2, 3, 4, 5, 6, 77)

# Saída de dados em Python

Python fornece a função **print()** para exibir a saída para os dispositivos de saída padrão.

Sintaxe:

**print**(valor)

Exemplo: saída de impressão Python

```
# Demostración de la función print()
print("GFG")

# Demostración de la función print() con espacios
print('G', 'F', 'G')
```

GFG

G F G

No exemplo acima, podemos ver que no caso da segunda declaração impressa há um espaço entre cada letra e a declaração impressa sempre acrescenta um novo caractere de linha no final da cadeia. Isto porque, após cada personagem, o parâmetro **sep** e no final da cadeia o parâmetro final é impresso. Vamos tentar mudar este parâmetro **sep** e **end**.

Exemplo: Saída Python **Print** com parâmetros de separação e terminação personalizados

```
print("GFG", end = "@")

print('G', 'F', 'G', sep = "#")
```

GFG@G#F#G

# Saída de dados com formato

A saída de dados formatados em Python pode ser feita de várias maneiras.

## Usando caracteres formatados literais

Podemos usar literais de corda formatada, começando uma corda com f ou F antes de abrir vírgulas invertidas ou vírgulas triplas invertidas. Nesta cadeia, podemos escrever expressões Python entre { } que pode se referir a uma variável ou qualquer valor literal.

Exemplo: formatação de cordas Python usando uma corda F

```
# Declaramos una variable
name = "Antonio"

# Salida
print(f'hola {name}!. Qué tal?')
```

Hola Antonio! Qué tal?

## Usando format()

Também podemos usar a função **format()** para formatar nossa produção para que ela pareça apresentável. As chaves { } funcionam como placeholders. Podemos especificar a ordem na qual as variáveis aparecem na saída.

Exemplo: formato de corda Python utilizando o **format()**

```
# Declaramos de variables
a = 20
b = 10

# Suma
sum = a+b

# Resta
sub = a-b

# Salida
print('El valor de a es {} y b es
{}'.format(a,b))
print('{2} es la suma de {0} y
{1}'.format(a, b, sum))
print('{sub_value} es la resta de
{value_a} y
{value_b}'.format(value_a=a,value_b=b,sub
_value=sub))
```

El valor de a es 20 y b es 10

30 es la suma de 20 y 10

10 es la resta de 20 y 10

## Uso do operador %

Podemos usar o operador '**%**'. Os valores de **%** são substituídos por valores de zero ou mais elementos. Formatação usando **%** é semelhante à de **'printf'** na linguagem de programação C.

**%d** – inteiro

**%f** – flutuante

**%s** - cadeia

**%x** - hexadecimal

**%o** – octal

Exemplo:

```
# Entrada de datos
num = int(input("Introduzca un número:
"))
add = num+5

# Salida
print("La suma es %d" %add)
```

Introduzca un número: 2

La suma es 7