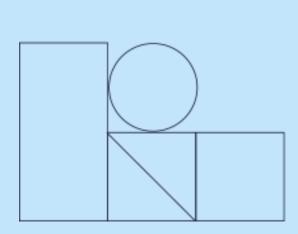
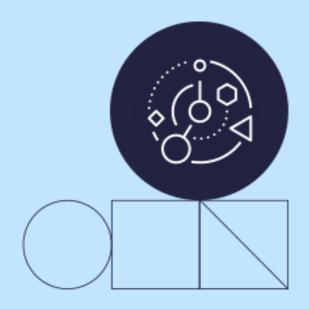
Conceptos básicos y sintaxis de Python

Entrada y salida de datos en Python





Índice	
Introducción	3
Cómo recibir información del usuario en Python	4
Cómo pedir varios valores de una sola vez	2
Entrada de datos en listas, conjuntos, tuplas, etc.	2
Solicitando elementos de List/Set uno por uno	Ę
Uso de los métodos map() y list() / set()	Ę
Entrada de datos en una tupla	Ę
Salida de datos en Python	Ę
Salida de datos con formato	6
Uso de literales de cadena formateados	6
Usando format()	6
Uso del operador %	7

Introducción

A continuación, veremos primero diferentes formas en las que podemos recibir información de los usuarios y luego mostrar la salida.

IBM.

Cómo recibir información del usuario en Python

A veces, podemos necesitar que el usuario introduzca un valor por consola. Para hacer esto, Python proporciona una función input().

Síntaxis:

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

Donde prompt es una cadena opcional (un mensaje) que se mostrará en el momento de realizar la petición de entrada.

<u>Ejemplo 1</u>: Pedir al usuario que introduzca su nombre.

```
# Entrada input del usuario
nombre = input('Introduce tu nombre: ')
# Salida
print("Hola, " + nombre)
print(type(nombre))
```

Salida:

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

Nota: Python toma todo aquello que el usuario introduzca por medio de un input() como un string. Para convertirlo a cualquier otro tipo de datos, tenemos que convertir la entrada explícitamente. Por ejemplo, para convertir la entrada a int o float tenemos que usar el método int() y float() respectivamente.

Ejemplo 2: Solicitar un número y sumarle una unidad.

```
# Entrada por parte del usuario como
número entero
num = int(input('Introduce un número: '))
add = num+1
# Salida
print(add)
```

Salida:

Introduce un número: 34
35

Cómo pedir varios valores de una sola vez

Podemos tomar múltiples entradas a la vez, usando el método map()

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

Salida:

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

Entrada de datos en listas, conjuntos, tuplas, etc.

En el caso de List y Set, la entrada puede tomarse del usuario de dos maneras.

- Solicitando los elementos List/Set uno por uno usando los métodos append()/add().
- Usando los métodos map() y list() / set().

Solicitando elementos de List/Set uno por uno

Para introducir los elementos de la Lista/Set uno por uno usaremos el método append() en el caso de las Listas, y el método add() en el caso de los 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)
```

Uso de los métodos map() y list() / 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)
```

Salida:

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

Entrada de datos en una tupla

Sabemos que las tuplas son inmutables, no hay métodos disponibles para agregar elementos a las tuplas. Para agregar un nuevo elemento a una tupla, primero deberemos convertir la tupla en lista, luego agregaremos el elemento a la lista y nuevamente convertiremos la lista en una 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)
```

Salida de datos en Python

Python proporciona la función **print()** para mostrar la salida a los dispositivos de salida estándar.

Sintaxis:

print(valor)

Ejemplo: salida de impresión de Python

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

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

En el ejemplo anterior, podemos ver que en el caso de la segunda declaración de impresión hay un espacio entre cada letra y la declaración de impresión siempre agrega un carácter de nueva línea al final de la cadena. Esto se debe a que después de cada carácter se imprime el parámetro sep y al final de la cadena se imprime el parámetro final. Intentemos cambiar este parámetro sep y end.

<u>Ejemplo</u>: Salida de Python <u>Print</u> con parámetro personalizado de separación y finalización

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

GFG@G#F#G

Salida de datos con formato

La salida de datos con formato en Python se puede hacer de diversas formas.

Uso de literales de cadena formateados

Podemos usar literales de cadena con formato, comenzando una cadena con f o F antes de abrir comillas o comillas triples. En esta cadena, podemos escribir expresiones de Python entre { } que pueden referirse a una variable o cualquier valor literal.

Ejemplo: formato de cadenas de Python usando una cadena F

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

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

Hola Antonio! Qué tal?

Usando format()

También podemos usar la función **format()** para formatear nuestra salida para que se vea presentable. Las llaves {} funcionan como marcadores de posición. Podemos especificar el orden en que aparecen las variables en la salida.

<u>Ejemplo</u>: formato de cadena de Python usando la función **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 del operador %

Podemos usar el operador '%'. Los valores de % se reemplazan con cero o más valores de elementos. El formateo usando % es similar al de 'printf' en el lenguaje de programación C.

```
%d – entero
%f – flotante
%s - cadena
%x - hexadecimal
%o – octal
```

Ejemplo:

```
# 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