Funciones más importantes en Python

October 23, 2024

A continuación, se presenta una lista de las funciones ms importantes en Python, con una explicación de para qué sirven y ejemplos de uso.

1 `print()`

La función `print()` se utiliza para mostrar información en la consola.

```
print("Hola, mundo!")
```

Salida:

```
Hola, mundo!
```

2 `len()`

La función `len()` devuelve la longitud (el número de elementos) de un objeto iterable como una lista, cadena o tupla.

```
mi_lista = [1, 2, 3, 4]
print(len(mi_lista))
```

Salida:

```
4
```

3 `input()`

La función `input()` permite al usuario ingresar datos desde la consola.

```
nombre = input("¿Cuál es tu nombre? ")
print("Hola, " + nombre)
```

Salida:

```
¿Cuál es tu nombre? Juan
Hola, Juan
```

4 `type()`

La función `type()` devuelve el tipo de dato del objeto pasado como argumento.

```
print(type(10))
print(type(3.14))
print(type("Python"))
```

Salida:

```
<class 'int'>
<class 'float'>
<class 'str'>
```

5 int(), float(), str()

Estas funciones convierten un valor a entero, flotante o cadena, respectivamente.

```
num_str = "100"
num_int = int(num_str)
num_float = float(num_str)
print(type(num_int))
print(type(num_float))
```

Salida:

```
<class 'int'>
<class 'float'>
```

6 range()

La función `range()` genera una secuencia de números.

```
for i in range(5):
    print(i)
```

Salida:

```
0
1
2
3
4
```

7 enumerate()

La función `enumerate()` añade un contador a un iterable y lo devuelve en forma de objeto enumerado.

```
lista = ['a', 'b', 'c']
for indice, valor in enumerate(lista):
    print(indice, valor)
```

Salida:

```
0 a
1 b
2 c
```

8 zip()

La función `zip()` toma iterables (como listas o tuplas) y devuelve un iterador de tuplas.

```python
nombres = ['Ana', 'Luis', 'Mar\'{i}a']
edades = [25, 30, 22]
for nombre, edad in zip(nombres, edades):
    print(f"{nombre} tiene {edad} a\~{n}os.")
# Salida:
# Ana tiene 25 años.
# Luis tiene 30 años.
# María tiene 22 años.
```

Salida:

```
Hola, mundo!
```

9 sum()

La función `sum()` suma todos los elementos de un iterable.

```python
numeros = [1, 2, 3, 4, 5]
print(sum(numeros))  # Salida: 15
```

Salida:

```
Hola, mundo!
```

10 max() y min()

Las funciones `max()` y `min()` devuelven el elemento mayor y menor de un iterable, respectivamente.

```python
numeros = [5, 2, 9, 1, 7]
print(max(numeros))  # Salida: 9
print(min(numeros))  # Salida: 1
```

Salida:

```
Hola, mundo!
```

11 sorted()

La función `sorted()` devuelve una lista ordenada de un iterable.

```python
numeros = [5, 2, 9, 1, 7]
print(sorted(numeros))  # Salida: [1, 2, 5, 7, 9]
```

Salida:

```
Hola, mundo!
```

12 open()

La función `open()` abre un archivo y devuelve un objeto de archivo.

```python
with open('archivo.txt', 'r') as f:
    contenido = f.read()
    print(contenido)
```

Salida:

Introducción a Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Se destaca por su simplicidad y legibilidad, lo que lo convierte en una excelente opción tanto para principiantes como para programadores experimentados.

Características principales de Python:

Sintaxis sencilla: Python utiliza una sintaxis clara y concisa, lo que facilita la escritura y comprensión del código.

Multiparadigma: Permite programación orientada a objetos, programación imperativa y programación funcional.

Extensa biblioteca estándar: Python cuenta con una amplia colección de módulos y paquetes que facilitan tareas comunes, como manejo de archivos, procesamiento de datos y creación de interfaces gráficas.

Comunidad activa: Python tiene una gran comunidad de desarrolladores que contribuyen constantemente con bibliotecas, frameworks y recursos de aprendizaje.

Aplicaciones diversas: Python se utiliza en diversos campos, incluyendo desarrollo web, análisis de datos, inteligencia artificial, automatización de tareas y más.

13 `list()`, `dict()`, `set()`

Estas funciones crean una lista, diccionario o conjunto respectivamente.

```
lista = list((1, 2, 3))
diccionario = dict(a=1, b=2)
conjunto = set([1, 2, 2, 3])
print(lista)
print(diccionario)
print(conjunto)
```

Salida:

[1, 2, 3]
'a': 1, 'b': 2
1, 2, 3

14 `map()`

La función `map()` aplica una función a todos los elementos de un iterable.

```
def cuadrado(x):
    return x * x

numeros = [1, 2, 3, 4]
resultado = map(cuadrado, numeros)
print(list(resultado))
```

Salida:

[1, 4, 9, 16]

15 filter()

La función `filter()` construye un iterador a partir de aquellos elementos de un iterable para los que una función devuelve verdadero.

```
def es_par(x):
    return x % 2 == 0

numeros = [1, 2, 3, 4, 5, 6]
pares = filter(es_par, numeros)
print(list(pares))  # Salida: [2, 4, 6]
```

Salida:

```
[2, 4, 6]
```

16 any() y all()

La función `any()` devuelve `True` si al menos un elemento de un iterable es verdadero. La función `all()` devuelve `True` si todos los elementos son verdaderos.

```
lista = [True, False, True]
print(any(lista))
print(all(lista))
```

Salida:

```
True
False
```

17 abs() y round()

La función `abs()` devuelve el valor absoluto de un número. La función `round()` redondea un número al entero más cercano o al número de decimales especificado.

Ejemplo:

```
print(abs(-5))
print(round(3.1416, 2))
```

Salida:

```
5
3.14
```

18 isinstance()

La función `isinstance()` verifica si un objeto es una instancia de una clase o de una subclase.

```
print(isinstance(5, int))
print(isinstance("hola", str))
print(isinstance(3.14, int))
```

Salida:

```
True
True
False
```