Todo lo que Debes Saber sobre print en Python

Publicado el 29 de junio de 2024

Clase

La función incorporada print puede parecer básico al principio, pero ten en cuenta que será una herramienta que usarás de múltiples maneras a lo largo de tu código. Desde el icónico “Hola mundo” hasta mensajes de depuración y presentación de resultados, print es la puerta de entrada a la comunicación de tus programas con el mundo exterior.

Iniciar con un simple “Hola mundo” no solo es una tradición en la programación, sino también un momento crucial donde tu código cobra vida. Es la primera línea de código que demuestra que tu entorno de desarrollo está configurado correctamente y que estás listo para empezar a crear.

Aprenderás a aprovechar al máximo la función incorporada print en Python. Desde formatos avanzados hasta el manejo de caracteres especiales y secuencias de escape, descubrirás cómo print puede ser una herramienta poderosa y versátil en tu caja de herramientas de programación.

**1. Uso básico de**print

El uso más sencillo de print consiste en pasar el texto que deseas mostrar entre comillas. Este código imprimirá “Nunca pares de aprender” en la consola, siendo una excelente forma de probar si tu entorno de Python está configurado correctamente.

print("Nunca pares de aprender")

Resultado:

Nunca pares de aprender

**2. Uso de la coma en**print

La coma dentro de la función print se usa para separar varios argumentos. Al hacerlo, Python añade automáticamente un espacio entre los argumentos. Esto es diferente a concatenar cadenas con el operador +, que no añade espacios adicionales.

print("Nunca", "pares", "de", "aprender")

Resultado:

Nunca pares de aprender

Por otro lado, al concatenar cadenas con el operador +, los elementos se unen sin ningún espacio adicional, a menos que lo añadas explícitamente.

print("Nunca" + "pares" + "de" + "aprender")

Resultado:

Nuncaparesdeaprender

Para añadir un espacio explícitamente cuando concatenas cadenas, debes incluirlo dentro de las comillas.

print("Nunca" + " " + "pares" + " " + "de" + " " + "aprender")

Resultado:

Nunca pares de aprender

**3. Uso de**sep

El parámetro sep permite especificar cómo separar los elementos al imprimir. En este ejemplo, los elementos “Nunca”, “pares”, “de” y “aprender” se imprimirán con una coma y un espacio entre ellos, resultando en “Nunca, pares, de, aprender”. Puedes cambiar sep por cualquier cadena de caracteres que desees usar como separador.

print("Nunca", "pares", "de", "aprender", sep=", ")

Resultado:

Nunca, pares, de, aprender

**4. Uso de**end

El parámetro end cambia lo que se imprime al final de la llamada a print. En lugar de imprimir cada mensaje en una nueva línea, end="" asegura que “Nunca” y “pares” se impriman en la misma línea, resultando en “Nunca pares”. Por defecto, end es un salto de línea ("\n"), lo que hace que cada llamada a print comience en una nueva línea.

print("Nunca", end=" ")

print("pares de aprender")

Resultado:

Nunca pares de aprender

**5. Impresión de variables**

Puedes usar print para mostrar el valor de las variables. En este ejemplo, imprimirá “Frase: Nunca pares de aprender” y “Autor: Platzi”. Esto es útil para depurar y ver los valores de las variables en diferentes puntos de tu programa.

frase = "Nunca pares de aprender"

author = "Platzi"

print("Frase:", frase, "Autor:", author)

Resultado:

Frase: Nunca pares de aprender Autor: Platzi

**6. Uso de formato con f-strings**

Las f-strings permiten insertar expresiones dentro de cadenas de texto. Al anteponer una f a la cadena de texto, puedes incluir variables directamente dentro de las llaves {}. En este ejemplo, frase y author se insertarán en la cadena, resultando en “Frase: Nunca pares de aprender, Autor: Platzi”. Esto hace que el código sea más legible y fácil de escribir.

frase = "Nunca pares de aprender"

author = "Platzi"

print(f"Frase: {frase}, Autor: {author}")

Resultado:

Frase: Nunca pares de aprender, Autor: Platzi

**7. Uso de formato con**format

El método format es otra forma de insertar valores en cadenas de texto. Usando {} como marcadores de posición, puedes pasar los valores que quieres insertar como argumentos de format. En este ejemplo, se imprimirá “Frase: Nunca pares de aprender, Autor: Platzi”. Es una forma flexible y poderosa de formatear cadenas, aunque las f-strings son más concisas.

frase = "Nunca pares de aprender"

author = "Platzi"

print("Frase: {}, Autor: {}".format(frase, author))

Resultado:

Frase: Nunca pares de aprender, Autor: Platzi

**8. Impresión con formato específico**

Puedes controlar el formato de los números al imprimir. En este ejemplo, :.2f indica que el número debe mostrarse con dos decimales. Así, imprimirá “Valor: 3.14”, redondeando el número a dos decimales. Esto es especialmente útil cuando trabajas con datos numéricos y necesitas un formato específico.

valor = 3.14159

print("Valor: {:.2f}".format(valor))

Resultado:

Valor: 3.14

**9. Saltos de línea y caracteres especiales**

Los saltos de línea en Python se indican con la secuencia de escape \n. Por ejemplo, para imprimir “Hola\nmundo”, que aparecerá en dos líneas:

print("Hola\nmundo")

Resultado:

Hola

mundo

Para imprimir una cadena que contenga comillas simples o dobles dentro de ellas, debes usar secuencias de escape para evitar confusiones con la sintaxis de Python. Por ejemplo, para imprimir la frase “Hola soy ‘Carli’”:

print('Hola soy \'Carli\'')

Resultado:

Hola soy 'Carli'

Si necesitas imprimir una ruta de archivo en Windows, que incluya barras invertidas, también necesitarás usar secuencias de escape para evitar que Python interprete las barras invertidas como parte de secuencias de escape. Por ejemplo:

print("La ruta de archivo es: C:\\Users\\Usuario\\Desktop\\archivo.txt")

Resultado:

La ruta de archivo es: C:\Users\Usuario\Desktop\archivo.txt

En Python, estas secuencias de escape te permiten manejar caracteres especiales y estructurar la salida de texto según sea necesario, asegurando que la salida se formatee correctamente en la consola o en cualquier otro medio donde se imprima.

Con estos ejemplos y explicaciones adicionales, tendrás una comprensión más completa sobre cómo manejar saltos de línea y caracteres especiales en Python al usar la función print.