📘 Guía Paso a Paso: Inmutabilidad de Cadenas en Python

✨ Introducción

En esta guía exploraremos el concepto de **inmutabilidad de las cadenas en Python**. Aprenderemos por qué una vez que se crea una cadena sus caracteres no pueden modificarse directamente y cómo podemos asignar nuevos valores a una variable de cadena mediante la creación de nuevos objetos en memoria. A través de un ejemplo práctico, veremos cómo funciona esta propiedad y cómo las referencias de variables apuntan a los objetos en memoria.

🔥 Paso 1: Crear el archivo de trabajo

Dentro de nuestro proyecto de Cadenas creamos lo siguiente:

📄 **Nombre del archivo:** `inmutabilidad_cadenas.py`
📂 **Ruta del archivo:** `Cadenas/inmutabilidad_cadenas.py`

📝 Descripción del paso:

Vamos a crear un archivo en PyCharm llamado `inmutabilidad_cadenas.py`. En este archivo trabajaremos con una cadena inicial `"Hola mundo"` para explorar la inmutabilidad de las cadenas.

👩‍💻 Código inicial: definición de la cadena

```
# Definimos la cadena inicial
cadena1 = 'Hola mundo'
print(f'Cadena original: {cadena1}')
```

👉 **Explicación:**
Aquí declaramos una variable `cadena1` con el valor `"Hola mundo"` y mostramos su contenido en consola. Este será nuestro punto de partida.

🔥 Paso 2: Intentar modificar un carácter de la cadena

📝 Descripción del paso:

Intentaremos **modificar el primer carácter** de la cadena, cambiando la **'H'** por **'h'** en minúscula, accediendo al índice `0`.

👩‍💻 Código para intentar modificar el carácter:

```
# Intentamos modificar el primer carácter
# Esto provocará un error porque las cadenas son inmutables
# cadena1[0] = 'h'
```

👉 **Explicación:**
El intento de asignar un nuevo valor directamente a un índice de la cadena no es permitido en Python y **provoca un error de tipo**. Por eso, el código está comentado para evitar que detenga el programa, pero muestra que esta operación no es válida.

🔥 Paso 3: Asignar un nuevo valor a la cadena

📝 Descripción del paso:

Aunque no podemos modificar directamente un carácter, **sí podemos asignar una nueva cadena** a la variable `cadena1`, lo que hace que la variable apunte a otro objeto en memoria.

👨‍💻 Código para asignar una nueva cadena:

```
# Asignamos una nueva cadena
cadena1 = 'Adios'
print(f'Nueva cadena: {cadena1}')
```

👉 **Explicación:**
Aquí estamos creando un nuevo objeto de tipo cadena con el valor `"Adios"` y asignándolo a `cadena1`. La variable ya no apunta a `"Hola mundo"`, sino al nuevo valor.

🔥 Paso 4: Mantener referencia al objeto original

📝 Descripción del paso:

Para conservar el valor original antes de asignar un nuevo valor, podemos crear una **nueva variable** que apunte al objeto original.

🦹 Código para mantener referencia al objeto original:

```
# Creamos una nueva referencia al objeto original
cadena1 = 'Hola mundo'
cadena2 = cadena1

# Asignamos una nueva cadena a cadena1
cadena1 = 'Adios'

# Imprimimos ambas variables
print(f'Cadena1: {cadena1}')
print(f'Cadena2: {cadena2}')
```

👉 **Explicación:**
Primero, `cadena2` apunta al mismo objeto que `cadena1` (`"Hola mundo"`). Luego, `cadena1` apunta a `"Adios"`, pero **`cadena2` sigue apuntando al objeto original** `"Hola mundo"`. De esta manera, el objeto original no es eliminado porque todavía existe una referencia a él.

📂 Archivo Final

✅ **Archivo completo:**

```
# Inmutabilidad en cadenas
cadena1 = 'Hola Mundo'
# cadena1[0] = 'h'  # no podemos modificar los caracteres
cadena2 = cadena1
cadena1 = 'Adios'
print(cadena1)
print(cadena2)
```

🏁 Conclusión

En esta guía aprendimos que **las cadenas en Python son inmutables**, lo que significa que **no podemos modificar directamente sus caracteres**. Sin embargo, sí podemos **asignar una nueva cadena** a la variable, creando un nuevo objeto en memoria. También vimos cómo **referencias adicionales permiten mantener acceso al objeto original**, evitando que sea eliminado inmediatamente por el recolector de basura.

🚀 Este concepto es clave para entender el manejo de memoria y la gestión de objetos en Python. ¡Sigue practicando para dominar el tema!

Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 👋

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de GlobalMentoring.com.mx