Estructura

El módulo `sys` en Python proporciona acceso a algunas variables y funciones específicas del intérprete de Python. Aquí hay algunas de las funcionalidades principales que proporciona el módulo `sys`:

1. **Variables del intérprete:**
   - `sys.argv`: Una lista que contiene los argumentos de la línea de comandos pasados al script Python.
   - `sys.path`: Una lista que especifica las ubicaciones donde Python buscará los módulos al importar.

2. **Funciones relacionadas con la ejecución:**
   - `sys.exit([status])`: Termina el programa de Python con un estado de salida opcional.
   - `sys.getsizeof(object)`: Devuelve el tamaño en bytes del objeto proporcionado.

3. **Otras funcionalidades:**
   - `sys.version`: Una cadena que especifica la versión del intérprete de Python.
   - `sys.platform`: Una cadena que indica el sistema operativo en el que se está ejecutando Python.

Estas son solo algunas de las funcionalidades proporcionadas por el módulo `sys`. En resumen, `sys` es útil para interactuar con aspectos específicos del entorno de ejecución de Python y para realizar tareas relacionadas con el sistema, como manipular rutas de archivos, obtener información del sistema operativo, etc.

En Python, `str` es una abreviatura de "string", que se refiere a una secuencia inmutable de caracteres Unicode. Los objetos de tipo `str` son utilizados para representar texto en Python. Aquí tienes algunos aspectos clave sobre las cadenas en Python (`str`):

1. **Creación de cadenas:**
   Puedes crear cadenas utilizando comillas simples (`'`) o comillas dobles (`"`).
Por ejemplo:

```
cadena1 = 'Hola Mundo'
cadena2 = "¡Bienvenido a Python!"
```

2. **Operaciones básicas con cadenas:**
   - Concatenación: `+` se utiliza para concatenar dos cadenas.
   - Repetición: `*` se utiliza para repetir una cadena.
   - Indexación y slicing: Puedes acceder a caracteres individuales o a subcadenas utilizando índices y slices, respectivamente.

3. **Inmutabilidad:**
   Las cadenas son inmutables en Python, lo que significa que no puedes modificar los caracteres individuales de una cadena. Sin embargo, puedes crear una nueva cadena con los cambios deseados.

4. **Métodos de cadenas:**
   Python proporciona una variedad de métodos incorporados para trabajar con cadenas, como
Estos son algunos de los métodos integrados más comunes para manipular cadenas en Python:

1. `upper()`: Este método convierte todos los caracteres de una cadena a mayúsculas y devuelve la cadena modificada. Por ejemplo:
   ```python
   cadena = "hola mundo"
   print(cadena.upper())  # Salida: HOLA MUNDO
   ```

2. `lower()`: Este método convierte todos los caracteres de una cadena a minúsculas y devuelve la cadena modificada. Por ejemplo:
   ```python
   cadena = "HOLA MUNDO"
   print(cadena.lower())  # Salida: hola mundo
   ```

3. `split()`: Este método divide una cadena en una lista de subcadenas, utilizando un delimitador como separador. Por defecto, el delimitador es el espacio en blanco. Por ejemplo:
   ```python
   cadena = "Hola,Mundo,Python"
   lista = cadena.split(",")
   print(lista)  # Salida: ['Hola', 'Mundo', 'Python']
   ```

4. `join()`: Este método une los elementos de una lista en una sola cadena, utilizando la cadena de llamada como separador. Por ejemplo:
   ```python
   lista = ['Hola', 'Mundo', 'Python']
   cadena = ','.join(lista)
   print(cadena)  # Salida: Hola,Mundo,Python
   ```

5. `strip()`: Este método elimina los espacios en blanco (u otros caracteres especificados) al principio y al final de una cadena y devuelve la cadena modificada. Por ejemplo:
   ```python
   cadena = "   Hola Mundo   "
   print(cadena.strip())  # Salida: "Hola Mundo"
   ```

6. `replace()`: Este método reemplaza todas las ocurrencias de una subcadena en una cadena con otra subcadena especificada y devuelve la cadena modificada. Por ejemplo:
   ```python
   ```

```
cadena = "Hola Mundo"
nueva_cadena = cadena.replace("Hola", "Adiós")
print(nueva_cadena)  # Salida: "Adiós Mundo"
```

Estos métodos son muy útiles para realizar tareas comunes de manipulación de cadenas en Python, como cambiar la capitalización, dividir y unir cadenas, eliminar espacios en blanco, y reemplazar subcadenas.

5. **Formato de cadenas:**
   Python ofrece varias formas de formatear cadenas, incluyendo el método `format()` y las f-strings (cadenas formateadas). Por ejemplo:
   ```python
   nombre = "Juan"
   edad = 30
   mensaje = "Hola, me llamo {} y tengo {} años.".format(nombre, edad)
   mensaje_fstring = f"Hola, me llamo {nombre} y tengo {edad} años."
   ```

6. **Longitud de cadenas:**
   Puedes obtener la longitud de una cadena utilizando la función `len()`. Por ejemplo:
   ```python
   cadena = "Python"
   longitud = len(cadena)  # longitud será 6
   ```

Las cadenas son fundamentales en Python y se utilizan ampliamente en todo tipo de aplicaciones para manipular texto y datos basados en caracteres.

format

- Es una función de cadenas para crear plantillas
- Utiliza `place-holders` nombrados
- No es posicional como el operador `%`

```
plantilla = 'hola {nombre}'
print(plantilla.format(nombre='Pepe'))

plantilla = 'Hola {nombre}, tu edad es {edad}, adiós {nombre}'
print(plantilla.format(edad=18, nombre='Juanito'))
```

```
hola Pepe
Hola Juanito, tu edad es 18, adiós Juanito
```

cadenas f

- No es exactamente para crear plantillas (dado que no son reutilizables)
- Son una forma conveniente de remplazar un `place-holder` por una variable que ya se tiene
- Se crean poniendo una `f` al inicio de la cadena

```
nombre = 'Pepe'
edad = 18
print(f'Hola {nombre}, tu edad es {edad}')
```

```
Hola Pepe, tu edad es 18
```

CHECAR NOTAS TEMA