EJECUTOR DE SCRIPTS PYTHON CON E/S EN ARCHIVOS DE TXT

Introducción:

El **Ejecutor de Scripts Python con E/S en Archivos de TXT** es una herramienta diseñada para leer, ejecutar y almacenar la salida de scripts Python desde archivos de texto. Este ejecutor es ideal para tareas de automatización, procesamiento en lote y generación de registros de ejecución.

¿Qué hace?:

- **Lectura de Código**: Obtiene el contenido de un archivo de texto que contiene un script Python.
- **Ejecución del Script**: Ejecuta el script Python dentro de un entorno controlado.
- **Almacenamiento de Salida**: Redirige la salida estándar del script, almacenando todo lo que normalmente se imprimiría en la consola.
- **Escritura de Salida**: Guarda la salida del script en un archivo de texto especificado, creando un registro persistente.

¿Cómo funciona el ejecutor?:

1 Lectura del Archivo de Entrada:

Utiliza la función `leer_codigo_desde_archivo` para leer el script Python desde un archivo de texto en una ruta especificada.

2 Redirección de la Salida Estándar:

Antes de ejecutar el script, redirige `sys.stdout` a un objeto `StringIO` para almacenar toda la salida de la consola.

3 Ejecución del Script:

Ejecuta el script utilizando `exec` dentro de un espacio de nombres local, que incluye alias personalizados para funciones y módulos. Maneja excepciones durante la ejecución, devolviendo mensajes de error detallados si ocurre algún problema.

4 Restauración de la Salida Estándar:

Restaura `sys.stdout` a su estado original después de la ejecución del script.

5 Escritura del Archivo de Salida:

Utiliza la función `escribir_salida_en_archivo` para guardar la salida capturada en un archivo de texto en una ruta especificada.

Modificación de Módulos, Funciones y Clases:

El ejecutor modifica varios módulos, funciones y clases estándar utilizando alias personalizados. Esta práctica se lleva a cabo para establecer un entorno controlado y, posiblemente, para simplificar el uso de ciertas funciones y clases en el script. A continuación, se detallan las modificaciones y su utilidad:

- `itertools` **como** `loki`: Proporciona herramientas avanzadas para trabajar con iteradores, como combinaciones, permutaciones y productos cartesianos.
- `nltk` **como** `cadena`: Biblioteca para el procesamiento del lenguaje natural. Permite análisis y manipulación de texto.
- `Lark` **como** `dino`: Biblioteca de parsing para convertir texto en estructuras de datos. Útil para el análisis sintáctico.
- `math` **como** `integrales`: Proporciona funciones matemáticas básicas y avanzadas, como trigonometría, logaritmos y cálculos de integrales.
- `print` **como** `paint`: Función estándar para imprimir texto en la consola, renombrada para darle un alias personalizado.
- `len` **como** `metro`: Función estándar para obtener la longitud de una colección, renombrada para darle un alias personalizado.
- `range` **como** `camaleon`: Función estándar para generar una secuencia de números, renombrada para darle un alias personalizado.
- `map` **como** `europa`: Función estándar para aplicar una función a todos los elementos de una iterable, renombrada para darle un alias personalizado.
- `list` **como** `asistencia`: Función estándar para convertir iterables en listas, renombrada para darle un alias personalizado.
- `union`: Función definida en el espacio de nombres local para unir dos elementos. Utiliza el operador de suma (`+`), facilitando la concatenación.

Características Destacadas:

- **Automatización**: Permite la ejecución automatizada de scripts Python sin necesidad de intervención manual, lo que resulta ideal para tareas programadas.
- **Registro de Ejecución**: Guarda toda la salida del script, incluyendo errores, en un archivo de texto para una revisión posterior.
- **Flexibilidad**: Compatible con cualquier script Python, siempre que se proporcionen las dependencias necesarias.
- **Fácil Integración**: Se puede integrar fácilmente en flujos de trabajo existentes que requieren la ejecución de scripts Python y el registro de resultados.

Posibles Usos:

- **Procesamiento en Lote**: Ejecución secuencial de múltiples scripts y almacenamiento de sus salidas para análisis posterior.
- **Automatización de Tareas**: Ejecución automatizada de scripts Python para tareas repetitivas, como la generación de informes o el procesamiento de datos.
- **Depuración y Desarrollo**: Registro detallado de salidas y errores para facilitar la depuración y el desarrollo de scripts.

Ejemplo de Uso:

Supongamos que tienes un script Python en un archivo llamado `Input.txt` con el siguiente contenido:

```
paint("Hola, soy Øwen (;")
```

Este script será leído, ejecutado, y su salida (`Hola, soy Øwen (;`) será guardada en un archivo de texto llamado `Output.txt`. Al final, `Output.txt` contendrá:

```
Hola, soy Øwen (;
```

Flujo de Ejecución del Script:

```
ruta_archivo_entrada = r'D:\VSC Programs\IV Semestre\Lenguaje de Programación II\Python\
Input.txt'
ruta_archivo_salida = r'D:\VSC Programs\IV Semestre\Lenguaje de Programación II\Python\
Output.txt'

codigo = leer_codigo_desde_archivo(ruta_archivo_entrada)

salida = ejecutar_codigo(codigo)

escribir_salida_en_archivo(salida, ruta_archivo_salida)

print("Ejecución completada.")
```

Código Completo del Ejecutor:

```
import sys
import io
import itertools as loki
import nltk as cadena
from lark import Lark as dino
import math as integrales

paint = print
metro = len
camaleon = range
europa = map
asistencia = list

def leer_codigo_desde_archivo(ruta_archivo):
    with open(ruta_archivo, 'r', encoding='utf-8') as archivo:
        codigo = archivo.read()
    return codigo

def escribir_salida_en_archivo(salida, ruta_archivo_salida):
    with open(ruta_archivo_salida, 'w', encoding='utf-8') as archivo:
        archivo.write(salida)

def ejecutar_codigo(codigo):
    old_stdout = sys.stdout
    new_stdout = io.StringIO()
```

```
    sys.stdout = new_stdout

    try:
        nombres local
        local_vars = {
            "loki": loki,
            "cadena": cadena,
            "dino": dino,
            "integrales": integrales,
            "paint": paint,
            "metro": metro,
            "camaleon": camaleon,
            "europa": europa,
            "asistencia": asistencia,
            "union": lambda a, b: a + b
        }
        exec(codigo, {}, local_vars)
    except Exception as e:
        return f"Error al ejecutar el código: {e}"
    finally:
        sys.stdout = old_stdout

    salida = new_stdout.getvalue()
    return salida

ruta_archivo_entrada = r'D:\VSC Programs\IV Semestre\Lenguaje de Programación II\Python\
Input.txt'
ruta_archivo_salida = r'D:\VSC Programs\IV Semestre\Lenguaje de Programación II\Python\
Output.txt'

codigo = leer_codigo_desde_archivo(ruta_archivo_entrada)
salida = ejecutar_codigo(codigo)
escribir_salida_en_archivo(salida, ruta_archivo_salida)

paint("Ejecución completada.")
```

Link del Repositorio y Código QR:

https://github.com/Lucc4z/Programming-Language-II/tree/main/Python