ENTREGA FINAL MÓDULO 3 BOOTCAMP FULL STACK PYTHON

1724882991433

Mini-sistema 'Harry Potter es el único Gran Mago'

Ejercicio para clasificación de Personajes

Author : Jota Cordova (jotacordovaj.io@gmail.com)

Fecha de creación : 16/08/2024

Última modificación: 28/08/2024

Versión para consola : 4.0.0

Versión para web : 1.1.0

Descripción del proyecto

Para la realización del ejercicio, he ofrecido 2 variantes. Una versión por consola, que procesa un archivo externo ".txt" con los personajes y genera un menú con las 6 opciones para obtener consultas y mostrarlas como listados usando la librería 'tabulate', cumpliendo el requerimiento; la otra, es una versión web, con una página que muestra un menú y renderiza la consulta en una tabla que utiliza elementos de bootstrap para que el usuario seleccione su consulta. La versión web también ofrece links para acceder al readme, un video de demostración, acceso a la aplicación montada en WEB, y al repositorio del proyecto en Github.

Nota: La versión WEB fue desplegada en la plataforma "PythonAnywhere", sin embargo, por razones de seguridad esta impide que se lea/escriba en un archivo de texto plano como csv/txt, por lo que se modificó el código e insertó una lista con los personajes, que es convertida a diccionario, siguiendo luego el mismo proceso que las otras versiones.

- **Objetivo:** Clasificar y manipular una lista de personajes.
- **Funcionalidades:**
    1. Lee un archivo externo en formato txt.
    2. Convierte los datos a un diccionario.
    3. Convierte el diccionario en 3 listas, según su valor.
    4. Agrega prefijo "El gran Mago" a personajes de lista "Magos"
    5. Genera lista con todos los personajes antes de ser modificados.
    6. Genera la lista de los "Magos" modificada con el prefijo "El gran mago"
    7. Genera lista con todos los personajes, incluyendo los modificados con su prefijo
    8. (Pendiente para una futura versión) Agregar un nuevo personaje al archivo externo, en formato clave:valor (nombre:clasificación)
- **Dependencias externas:**
    1. Archivo externo: "personajes.txt", permite algún grado de escalabilidad a esta prueba de conceptos, además separa los datos del código, evitando embeberlos.
- **Tecnologías:**
    - Python==3.12.15
    - Tabulate==0.9.0
    - Flask==3.0.3
    - Bootstrap 5
- **Estructura del código:**
    - Cada función incluye docstrings, con indicación de:

      a) Su Propósito o qué hace.

      b) Parámetros que recibe.

      c) Lo qué retorna.

Detalles técnicos

- **Estructura de datos:** Archivo externo para almacenar raw-data, diccionarios y listas para manipular personajes.
- **Algoritmos:** Clasificación ordenamiento y búsqueda de valores utilizando los métodos de la clase diccionario y listas.
- **Librerías:**
    - **Tabulate:** Para formatear tablas de salida.

- **OS:** Para interactuar con el sistema operativo (limpiar pantalla).

Historial de la versión para consola

- **1.0.0 (16/08/2024):** Versión inicial, el listado de personajes se encuentra embebido en el código, se ejecuta y muestra todo sin control, cumple el objetivo básico.
- **2.0.0 (18/08/2024):** Agrega menú e interacciones básicas.
- **2.0.1 (19/08/2024):** Corrige opciones del menú y controla entorno de ejecución.
- **3.0.0 (20/08/2024):** Agrega ingesta de datos desde un archivo externo ".csv".
- **4.0.0 (27/08/2024):** Agrega más validaciones en la ejecución, optimiza limpieza de pantalla, más opciones de menú y acceso desde un ".txt".

Historial de versiones para web (localhost)

- **1.0.0 (20/08/2024):** Versión inicial, localhost, accede a archivo externo, muestra las 4 opciones básicas del problema. Accede a documentación
- **1.1.0 (28/08/2024):** Versión PythonAnywhere, visible públicamente, se omite función que lee archivo externo por restricciones de seguridad, se lee lista del código y convierte a diccionario, el resto sigue igual. Se actualiza documentación

Diccionario de objetos

Elemento	Tipo	Descripción
Variables		
v_nomArchivo	str	Nombre del archivo de texto a leer.
dict_personajes	dict	Diccionario que almacena los personajes y sus clasificaciones.
lst_magos	list	Lista de nombres de los magos.
lst_cientificos	list	Lista de nombres de los científicos.
lst_otros	list	Lista de nombres de otros personajes.
lst_grandiosos	list	Lista de nombres de magos con el prefijo "El gran".
v_limpiarPantalla	bool	Flag para controlar si se limpia la pantalla.
v_cont	str	Variable para almacenar la respuesta del usuario si desea continuar.
Funciones		
f_limpiarPantalla()/f_limpiar_pantalla()	function	Limpia la pantalla de la consola.
f_leerArchivo()/f_leer_archivo()	function	Lee un archivo de texto y crea un diccionario.
f_clasificarPersonajes()/f_clasificar_personajes()	function	Clasifica los personajes en diferentes categorías.
f_hacerGrandioso()/f_hacer_grandioso()	function	Agrega el prefijo "El gran" a los nombres de los magos.
f_imprimirNombres()/f_imprimir_nombres()	function	Imprime los nombres de una lista.
f_imprimirTodos()/f_imprimir_todos()	function	Imprime todos los personajes con su clasificación.
f_imprimirMagos()/f_imprimir_magos()	function	Imprime solo los nombres de los magos.
f_imprimirGrandiosos()/f_imprimir_grandiosos()	function	Imprime los nombres de los magos con el prefijo.
f_imprimirFinal()/f_imprimir_lista_final()	function	Imprime todos los personajes después de agregar el prefijo.
f_menu()	function	Muestra el menú

Nota: A partir de la versión 3.1, se cambió la notación de las funciones, coherente con "snake_case" Python.

Explicación de las columnas:

- **Elemento:** Nombre de la variable o función.
- **Tipo:** Tipo de dato (str, int, float, list, dict, function, etc.).
- **Descripción:** Qué hace o almacena el objeto.

Consideraciones adicionales:

Guía de estilos y nomenclatura:

1. Para una mejor identificación, se han agregado prefijos a los objetos:

   f_: Para funciones
   lst_ : Para listas
   v_: Para variables
   dict_: Para diccionarios

2. Notación: Además del prefijo, todos los nombres de objetos, inician con la primera palabra en minúscula y la segunda con la primera letra en mayúscula, ejemplo: "unEjemplo"

3. Se trata de mantener coherencia con PEP8

Próximo release:

4.0.0 (30/08/2024): Agregar un nuevo personaje modificando el archivo de texto.

Ejecución

Para las versiones de consola:

- **Bash (Abrir la consola en la carpeta del proyecto, botón derecho, menú contextual)**

```
python magosv4.py
```

- **Windows CMD, moverse hasta la carpeta del proyecto**

```
python magosv4.py
```

- **Windows PS, moverse hasta la carpeta del proyecto**

  ```
  python magosv4.py
  ```

- **Windows PS (alternativo), moverse hasta la carpeta del proyecto**

  ```
  Invoke-Expression "python magosv4.py"
  ```

Para la versión WEB (localhost):

- Instalar Flask

  '''CMD
  pip install Flask'''

- Instalar tabulate

  '''CMD
  pip install tabulate'''

- Correr el programa

  '''CMD
  python efm3_app.py'''

Nota: Desde la versión local (http://127.0.0.1:5000/), se puede acceder a la versión en la nube pública, https://jcordovaj.pythonanywhere.com/.

Si desea, desde la consola de Bash, en el directorio de trabajo, puede ejecutar el programa **"script.sh"**, que instalará las dependencias y correrá el programa para web. Lo mismo si abre una consola de CMD, puede ejecutar el script.bat, que genera el mismo resultado.

Finalmente, basta con hacer clic sobre la dirección del local host o, digitarlo directamente en el navegador. Con Ctrl+C, se interrupe el servidor web local y se sale.

1724879744375

Te gustó?, entonces clona el proyecto y dame una estrella

- Si este repositorio te ha sido útil, por favor, considera clonarlo con el siguiente comando:

  '''bash
  git clone https://github.com/jcordovaj/evalM3.git
  '''

- Si quieres mostrarme tu apoyo, dame una estrella ☆ aquí: Dar estrella