Prueba Técnica para Desarrollador Python

Instrucciones Generales:

- Tienes un tiempo estimado de 4 horas para completar la prueba.
- Envía tu solución en un archivo comprimido que contenga el código, la documentación y cualquier otro archivo necesario para ejecutar la prueba.
- Incluye un archivo README con instrucciones sobre cómo ejecutar tu código.

Parte 1: Normalización y Trabajo con Datos

Descripción: Te proporcionamos un archivo CSV llamado `clientes.csv` con información de clientes. Las columnas son: `Nombre`, `Apellido`, `Email`, `Teléfono`, `FechaRegistro`.

Tareas:

1. Crea un script en Python que realice las siguientes acciones:
    - Lee el archivo CSV.
    - Normaliza los datos (por ejemplo, convierte los nombres y apellidos a formato capitalizado, valida que los correos electrónicos tengan un formato correcto y normaliza el número de teléfono a un formato estándar).
    - Guarda los datos normalizados en un nuevo archivo CSV llamado `clientes_normalizados.csv`.

Entradas:

- Archivo: `clientes.csv` (proporcionado).

Salidas:

- Archivo: `clientes_normalizados.csv` con las columnas normalizadas.

Parte 2: Acceso y Manipulación de Datos en SQL

Descripción: Usando una base de datos SQL (puede ser SQLite o cualquier otra de tu elección), crea una tabla llamada `clientes` con las columnas mencionadas en la parte anterior.

Tareas:

1. Escribe un script en Python que:
    - Se conecte a la base de datos.
    - Inserte los datos normalizados del archivo `clientes_normalizados.csv` en la tabla `clientes`.

- Realice una consulta para obtener la cantidad de clientes registrados por año y la imprima en la consola.

Requerimientos:

- Utiliza la librería `sqlite3` o equivalente para manejar la conexión a la base de datos.
- Presenta los resultados de manera clara.

Parte 3: Diseño de API

Descripción: Diseña una API sencilla utilizando Flask (o FastAPI) que permita acceder a los datos de clientes.

Tareas:

1. Crea una API con las siguientes rutas:
   - `GET /clientes`: Devuelve todos los registros de la tabla `clientes`.
   - `GET /clientes/<email>`: Devuelve la información de un cliente específico por su correo electrónico.
   - `POST /clientes`: Permite agregar un nuevo cliente a la base de datos (recibiendo datos en formato JSON).

Documentación:

- Incluye un archivo `API_DOCUMENTATION.md` que explique cómo usar la API, con ejemplos de solicitudes y respuestas.

Parte 4: Automatizaciones y Bots de Acceso a Web

Descripción: Crea un bot sencillo que acceda a una página web (puedes usar un sitio de prueba como http://httpbin.org) y recolecte datos de interés, como el encabezado de respuesta o el contenido de una sección específica.

Tareas:

1. Implementa un script en Python que:
   - Realice una solicitud GET a la página web.
   - Imprima el código de estado de la respuesta y el contenido HTML de la página.
   - Extraiga un dato específico utilizando `BeautifulSoup` (puedes elegir un elemento simple como el título de la página).

Requerimientos:

- Asegúrate de manejar posibles errores (por ejemplo, si la página no está disponible).

Parte 5: Código Limpio y Pruebas

1. Asegúrate de seguir principios de código limpio a lo largo de todos los scripts (nombres descriptivos, comentarios adecuados, funciones bien definidas).
2. Implementa pruebas unitarias para al menos una de las funcionalidades (por ejemplo, la normalización de datos) utilizando `unittest` o `pytest`.

Criterios de Evaluación:

- **Correctitud**: La solución cumple con los requisitos especificados.
- **Calidad del Código**: El código es limpio, bien estructurado y comentado.
- **Documentación**: La documentación es clara y completa.
- **Manejo de Errores**: Se han considerado y manejado adecuadamente los errores potenciales.
- **Pruebas**: Se incluyen pruebas que validan la funcionalidad del código.

¡Buena suerte!

—