Desafío - Estructuras de datos y funciones (I)

En este desafío validaremos nuestros conocimientos utilizando las distintas propiedades que las estructuras de datos tienen para resolver problemáticas. Para lograrlo, necesitarás utilizar el archivo ***Desafío evaluado - Estructuras de datos y funciones (I) (Apoyo).zip***

Lee todo el documento antes de comenzar el desarrollo **individual**, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Requerimientos

Crea los siguientes programas:

1. Crear un archivo `conversiones.py` y una estructura de datos apropiada que permita ingresar tasas de conversión. Las distintas tasas de conversión se deben ingresar mediante `sys.argv` en el siguiente orden: Sol, Peso Argentino, Dólar Americano. **(4 Puntos)**

   Para ello considere las siguientes tasas de conversión de Peso Chileno:

- a Sol peruano: 0.0046
- a Peso Argentino: 0.093
- a Dólar Americano: 0.00013

Además ingrese un 4to argumento que sea el valor en peso chileno a convertir. El programa debe devolver el valor en peso chileno convertido en las 3 divisas ingresadas.
Al ejecutar el programa se espera el siguiente output:

```
python conversiones.py 0.0046 0.093 0.0013 10000
```

Respuesta esperada:

```
Los 10000 pesos equivalen a:
46.0 Soles
930.0 Pesos Argentinos
13.0 Dólares
```

2. El texto **lorem ipsum** es un texto de prueba que se utiliza para demostrar distintas tipografías además de usarse para rellenar espacios que requieran largos textos. ¿Cuántas palabras componen este texto?
(3 Puntos)

   Genere un archivo llamado `word_count.py` que importe un texto a Python y realice las siguientes tareas:

   - Utilizando una estructura de datos apropiada, cuente la cantidad de caracteres distintos que componen un texto.
   - Cuente el número de palabras distintas que componen el texto ingresado. Para separar un texto por espacios puede utilizar el `método .split("")`.

TIP: Para importar texto en python puede adaptar el siguiente código:

```
with open("texto.txt", "r") as file:
      texto=file.read()
```

donde "**texto.txt**" corresponderá a la ruta del archivo a importar.

Para comprobar el correcto funcionamiento del código se provee el archivo **lorem_ipsum.txt**. Al ejecutar el programa se espera el siguiente output.

```
python word_count.py lorem_ipsum.txt
```

Respuesta esperada:

```
El número de caracteres distintos es: 40
El número de palabras distintas es: 243
```

3. Se provee del archivo `recordatorios.py` que incluye una serie de eventos que quieren ser recordados por usted. El formato de estos recordatorios son una fecha (año-mes-día), una hora y una descripción del evento.
   (3 Puntos)

   Aplicando métodos apropiados para la estructura de datos entregada edite la lista de recordatorios de la siguiente manera:

    **NOTA**: Los eventos deben siempre editarse de tal manera que mantengan su orden en el tiempo. Y el código debe ejecutarse en el orden entregado en las instrucciones dadas a continuación:

   1. Agregue un evento el 2 de Febrero de 2021 a las 6 de la mañana para "Empezar el Año".
   2. Al revisar los eventos, nota un error, ya que el 15 de Julio no es feriado. Editar de tal manera que sea el 16 de Julio.
   3. Lamentablemente le tocará trabajar el día del trabajo. Elimine el evento del día del trabajo.
   4. Agregue una cena de Navidad y de Año Nuevo cuando corresponda. Ambas a las 22 hrs.

Al ejecutar el programa se espera el siguiente output:

```
python recordatorios.py
```

```
[['2021-01-01', '11:00', 'Levantarse y ejercitar'],
['2021-01-02', '06:00', 'Empezar el año'],
['2021-07-16', '13:00', 'No hacer nada es feriado'],
['2021-09-18', '16:00', 'Ramadas'],
['2021-12-24', '22:00', 'Cena de Navidad'],
['2021-12-25', '00:00', 'Navidad'],
['2021-12-31', '22:00', 'Cena de Año Nuevo']]
```

 ¡Mucho éxito!

Consideraciones y recomendaciones

- Comprime en un .zip los 3 programas desarrollados y luego sube en el LMS