Desafio Stark

Industrias Stark es principalmente una empresa de defensa que desarrolla y fabrica armas avanzadas y tecnologías militares.

La empresa compartió con los alumnos de la UTN cierta información confidencial de un grupo de superhéroes. En base a esta información la empresa ha solicitado llevar a cabo una serie de informes y análisis de datos.

Conjunto de datos

La información a ser analizada se encuentra en el archivo data_stark.py, por tratarse de una lista, bastará con incluir dicho archivo en el proyecto de la siguiente manera:

```
from data_stark import lista_personajes
```

Formato de los datos recibidos

```
lista_personajes =\
[
  {
    "nombre": "Howard the Duck",
    "identidad": "Howard (Last name unrevealed)",
    "empresa": "Marvel Comics",
    "altura": "79.349999999999994",
    "peso": "18.449999999999999",
    "genero": "M",
    "color_ojos": "Brown",
    "color_pelo": "Yellow",
    "fuerza": "2",
    "inteligencia": ""
  },
  {
    "nombre": "Rocket Raccoon",
    "identidad": "Rocket Raccoon",
    "empresa": "Marvel Comics",
    "altura": "122.77",
    "peso": "25.73",
    "genero": "M",
    "color_ojos": "Brown",
    "color_pelo": "Brown",
    "fuerza": "5",
    "inteligencia": "average"
  }
```

IMPORTANTE: ***Para todas y cada una de las funciones creadas, documentarlas escribiendo que es lo que hacen, que son los parámetros que reciben (si es una lista, un string, etc y que contendrá) y que es lo que retorna la función!***

0. Crear la función '***stark_normalizar_datos()***' la cual recibirá por parámetro la lista de héroes. La función deberá:
    - Recorrer la lista y convertir al tipo de dato correcto las keys (solo con las keys que representan datos numéricos) por ejemplo fuerza (int), altura (float), etc
    - Validar primero que el tipo de dato no sea del tipo al cual será casteado. Por ejemplo si una key debería ser entero (ejemplo fuerza) verificar antes que no se encuentre ya en ese tipo de dato.
    - Si al menos un dato fue modificado, la función deberá retornar el valor booleano **True**
    - En caso de que la lista esté vacía o ya se hayan normalizado anteriormente los datos se deberá retornar el valor booleano **False**
    - Crear una opción en el menú que me permita normalizar los datos (No se debería poder acceder a ninguna otra opción del menú hasta que los datos esten normalizados)
    - En caso de que la llamada a la función retorne **True** mostrar un mensaje diciendo “Datos Normalizados” sino mostrar el mensaje *“Hubo un error al normalizar los datos. Verifique que la lista no este vacía o que los datos ya no se hayan normalizado anteriormente”*

1.1. Crear la función *”**obtener_dato()**”* la cual recibirá por parámetro un diccionario el cual representara a un héroe y también recibirá un string que hace referencia a una *“clave”* del mismo
Validar siempre que el diccionario no esté vacío y que el mismo tenga una key llamada *“nombre”*. Caso contrario la función retornara un **False**

1.2 Crear la función *'**obtener_nombre**'* la cual recibirá por parámetro un diccionario el cual representara a un héroe y devolverá un string el cual contenga su nombre formateado de la siguiente manera:

Nombre: Howard the Duck

Validar siempre que el diccionario no este vacío y que la key que se pide exista. Caso contrario la función retornara un **False**

NOTA: Reutilizar la función creada en el punto anterior

2. Crear la función *'obtener_nombre_y_dato'* la misma recibirá por parámetro un diccionario el cual representara a un héroe y una key (string) la cual representará el dato que se desea obtener.

    - La función deberá devolver un string el cual contenga el nombre y dato (key) del héroe a imprimir. El dato puede ser 'altura', 'fuerza', 'peso' O **CUALQUIER OTRO DATO**.

    - El string resultante debe estar formateado de la siguiente manera: (suponiendo que la key es fuerza)
      Nombre: Venom | fuerza: 500
    - Validar siempre que la lista no este vacía. Caso contrario la función retornara un **False**

NOTA: Reutilizar las funciones del punto anterior

3.1 Crear la función ***"obtener_maximo()"*** la cual recibirá como parámetro una lista y una key (string) la cual representará el dato al cual se le debe calcular su cantidad **MÁXIMA.**

- Validar siempre que la lista no esté vacía y que el dato que está buscando sea un int o un float. Caso contrario la función retornara un **False**
- En caso de que el dato que se está buscando en el diccionario es de tipo int o float retornar el mayor que haya encontrado en la búsqueda.

3.2 Crear la función “***obtener_minimo()”*** la cual recibirá como parámetro una lista y una key (string) la cual representará el dato al cual se le debe calcular su cantidad **MÍNIMA.**

- Validar siempre que la lista no esté vacía y que el dato que está buscando sea un int o un float. Caso contrario la función retornara un **False**
- En caso de que el dato que se está buscando en el diccionario es de tipo int o float retornar el menor que haya encontrado en la búsqueda.

3.3 Crear la función ***'obtener_dato_cantidad()***' la cual recibira tres parámetros:

- La lista de héroes
- Un número que me indique el valor a buscar (puede ser la altura máxima, la altura mínima o cualquier otro dato)
- Un string que representa la key del dato a calcular, por ejemplo: ‘*altura*’, ‘*peso*’, ‘*edad*’, etc.

La función deberá retornar una lista con el héroe o los heroes que cumplan con la condición pedida. Reutilizar las funciones hechas en los puntos 3.1 y 3.2

Ejemplo de llamada:

mayor_altura = obtener_maximo(lista_heroes,”altura”)
lista_heroes_max_altura = 'obtener_dato_cantidad(lista_heroes,mayor_altura,”altura”)

El objetivo de estás llamadas es obtener todos los superhéroes que tengan la altura correspondiente a la altura máxima, y la misma función me podría servir tanto como para altura menor, como la mayor o alguna altura que yo le especifique también.

3.4 Crear la función ***'stark_imprimir_heroes'*** la cual recibirá un parámetro:

- La lista de héroes

Validar que la lista de héroes no esté vacía para realizar sus acciones, caso contrario no hará nada y retornara **False**

En caso de que la lista no este vacia imprimir la información completa de todos los heroes de la lista que se le pase

Ejemplo de llamada:

mas_pesado = obtener_maximo(lista_heroes,”peso”)
lista_mas_pesados = 'obtener_dato_cantidad(lista_heroes,mas_pesado ,”peso”)

stark_imprimir_heroes(lista_mas_pesados) -> Imprimo sólo los héroes más pesados

stark_imprimir_heroes(lista_heroes) -> Imprimo todos los héroes

4.1 Crear la función '***sumar_dato_heroe***' la cual recibirá como parámetro una lista de héroes y un string que representara el dato/key de los héroes que se requiere sumar. Validar que cada héroe sea tipo diccionario y que no sea un diccionario vacío antes de hacer la suma. La función deberá retorna la suma de todos los datos según la key pasada por parámetro

4.2 Crear la función '***dividir***' la cual recibirá como parámetro dos números (dividendo y divisor). Se debe verificar si el divisor es 0, en caso de serlo, retornar **False**, caso contrario realizar la división entre los parámetros y retornar el resultado

4.3 Crear la función '***calcular_promedio***' la cual recibirá como parámetro una lista de héroes y un string que representa el dato del héroe que se requiere calcular el promedio. La función debe retornar el promedio del dato pasado por parámetro

IMPORTANTE: hacer uso de las las funciones creadas en los puntos 4.1 y 4.2

4.4 Crear la función '**mostrar_promedio_dato**' la cual recibirá como parámetro una lista de héroes y un string que representa la clave del dato

- Se debe validar que el dato que se encuentra en esa clave sea de tipo int o float. Caso contrario retornaria **False**
- Se debe validar que la lista a manipular no esté vacía , en caso de que esté vacía se retornaria también **False**

5.1 Crear la función "***imprimir_menu***" que imprima el menú de opciones por pantalla, el cual permite utilizar toda la funcionalidad ya programada.

5.2 Crear la función “***validar_entero***” la cual recibirá por parámetro un string de número el cual deberá verificar que sea sea un string conformado únicamente por dígitos. Retornara True en caso de serlo, False caso contrario

5.3 Crear la función *'**stark_menu_principal**'* la cual se encargará de imprimir el menú de opciones y le pedirá al usuario que ingrese el número de una de las opciones elegidas y deberá validarlo. En caso de ser correcto dicho número, lo retornara casteado a int , caso contrario retorna **False**. Reutilizar las funciones del ejercicio 5.1 y 5.2

6.Crear la función *'**stark_marvel_app**'* la cual recibirá por parámetro la lista de héroes y se encargará de la ejecución principal de nuestro programa.
Utilizar if/elif o match según prefiera. Debe informar por consola en caso de seleccionar una opción incorrecta y volver a pedir el dato al usuario. Reutilizar las funciones con prefijo 'stark_' donde crea correspondiente.

7. Una vez realizadas y probadas las funciones resolver en un menú los siguientes puntos del desafio anterior.

- **A.** Normalizar datos (No se debe poder acceder a los otros puntos)
- **B.** Recorrer la lista imprimiendo por consola el nombre de cada superhéroe de género **NB**
- **C.** Recorrer la lista y determinar cuál es el superhéroe más alto de género **F**
- **D.** Recorrer la lista y determinar cuál es el superhéroe más alto de género **M**
- **E.** Recorrer la lista y determinar cuál es el superhéroe más débil de género **M**
- **F.** Recorrer la lista y determinar cuál es el superhéroe más débil de género **NB**
- **G.** Recorrer la lista y determinar la fuerza promedio de los superhéroes de género **NB**
- **H.** Determinar cuántos superhéroes tienen cada tipo de color de ojos.
- **I.** Determinar cuántos superhéroes tienen cada tipo de color de pelo.
- **J.** Listar todos los superhéroes agrupados por color de ojos.
- **K.** Listar todos los superhéroes agrupados por tipo de inteligencia.