UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Desafio Stark

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

STARK INDUSTRIES

Recientemente ha decidido ampliar su departamento de IT y para acceder a las entrevistas es necesario completar el siguiente desafío, el cual estará dividido en etapas. Cada semana recibiremos un nuevo pedido de parte de la empresa.

La empresa compartió con todos los participantes cierta información confidencial de un grupo de superhéroes. Y semana a semana enviará una lista con los nuevos requerimientos. Quien supere todas las etapas accedera a una entrevista con el director para de la compañía.

Set de datos

La información a ser analizada se encuentra en el archivo data_stark.json.

Desafío #05: (crear biblioteca stark_desafio_5.py)

Basándose en el desafío #01 deberás hacer la siguiente ejercitación, utilizando un menú que permita acceder a cada uno de los puntos por separado.

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!. Capturar las Excepciones donde crea necesario y mostrar por consola un mensaje en caso de error, para que el programa no explote y continúe su ejecución.***

A. Recorrer la lista imprimiendo por consola el nombre de cada superhéroe de género M
B. Recorrer la lista imprimiendo por consola el nombre de cada superhéroe de género F
C. Recorrer la lista y determinar cuál es el superhéroe más alto de género M
D. Recorrer la lista y determinar cuál es el superhéroe más alto de género F
E. Recorrer la lista y determinar cuál es el superhéroe más bajo de género M
F. Recorrer la lista y determinar cuál es el superhéroe más bajo de género F
G. Recorrer la lista y determinar la altura promedio de los superhéroes de género M
H. Recorrer la lista y determinar la altura promedio de los superhéroes de género F
I. Informar cual es el Nombre del superhéroe asociado a cada uno de los indicadores anteriores (ítems C a F)
J. Determinar cuántos superhéroes tienen cada tipo de color de ojos.
K. Determinar cuántos superhéroes tienen cada tipo de color de pelo.
L. Determinar cuántos superhéroes tienen cada tipo de inteligencia (En caso de no tener, Inicializarlo con 'No Tiene').
M. Listar todos los superhéroes agrupados por color de ojos.

N. Listar todos los superhéroes agrupados por color de pelo.
O. Listar todos los superhéroes agrupados por tipo de inteligencia

Para ello tendrás que seguir las siguientes directivas paso a paso.

1. Primera Parte

1.1. Crear la función "imprimir_menu_desafio_5" que imprima el menú de opciones por pantalla (las opciones son las que se van a utilizar para acceder a la funcionalidad de los puntos A hasta el O y Z para salir). Reutilizar la función 'imprimir_dato' realizada en una práctica anterior.

1.2. Crear la funcion 'stark_menu_principal_desafio_5' la cual se encargará de imprimir el menú de opciones y le pedirá al usuario que ingrese la letra de una de las opciones elegidas, deberá validar la letra usando RegEx y en caso de ser correcta tendrá que retornarla, Caso contrario retornará -1. El usuario puede ingresar tanto letras minúsculas como mayúsculas y ambas se deben tomar como válidas
Reutilizar la función 'imprimir_menu_Desafio_5'

1.3. Crear la función 'stark_marvel_app_5' la cual recibirá por parámetro la lista de héroes y se encargará de la ejecución principal de nuestro programa. (usar if/elif o match según prefiera) Reutilizar las funciones con prefijo 'stark_' donde crea correspondiente.

1.4. Crear la función 'leer_archivo' la cual recibirá por parámetro un string que indicará el nombre y extensión del archivo a leer (Ejemplo: archivo.json). Dicho archivo se abrirá en modo lectura únicamente y retornará la lista de héroes como una lista de diccionarios.

1.5. Crear la función 'guardar_archivo' la cual recibirá por parámetro un string que indicará el nombre con el cual se guardará el archivo junto con su extensión (ejemplo: 'archivo.csv') y como segundo parámetro tendrá un string el cual será el contenido a guardar en dicho archivo. Debe abrirse el archivo en modo escritura, ya que en caso de no existir, lo creara y en caso de existir, lo va a sobreescribir La función debera retornar True si no hubo errores, caso contrario False, además en caso de éxito, deberá imprimir un mensaje respetando el formato:

.Se creó el archivo: nombre_archivo.csv
En caso de retornar False, el mensaje deberá decir: ‘Error al crear el archivo: nombre_archivo’
Donde nombre_archivo será el nombre que recibirá el archivo a ser creado, conjuntamente con su extensión. (Manejar posibles Excepciones)

1.6. Crear la función 'capitalizar_palabras' la cual recibirá por parámetro un string que puede contener una o muchas palabras. La función deberá retornar dicho string en el cual todas y cada una de las palabras que contenga, deberán estar capitalizadas (Primera letra en mayúscula).

1.7. Crear la función 'obtener_nombre_capitalizado' la cual recibirá por parámetro un diccionario el cual representará a un héroe y devolverá un string el cual contenga su nombre formateado de la siguiente manera:
Nombre: Venom
Reutilizar 'capitalizar_palabras'

1.8. Crear la función 'obtener_nombre_y_dato' la cual recibirá por parámetro un diccionario el cual representará a un héroe y una key (string) la cual representará la key del héroe a imprimir. La función devolverá 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 al estilo: (suponiendo que la key es fuerza)
Nombre: Venom | Fuerza: 500
Reutilizar 'obtener_nombre_capitalizado'

2. Segunda Parte

2.1. Crear la función 'es_genero' la cual recibirá por parámetro un diccionario que representará un héroe y un string el cual será usado para evaluar si el héroe coincide con el género buscado (El string puede ser M, F o NB). retornará True en caso de que cumpla, False caso contrario.

2.2. Crear la función 'stark_guardar_heroe_genero' la cual recibirá por parámetro la lista de héroes y un string el cual representará el género a evaluar (puede ser M o F). Deberá imprimir solamente los héroes o heroínas que coincidan con el género pasado por parámetro y además, deberá guardar todos los datos de los héroes en un archivo con extensión csv (cada dato del heroe/heroina deberá ir separado por una coma)
Reutilizar las funciones 'es_genero', 'obtener_nombre_capitalizado', 'imprimir_dato' y 'guardar_archivo'.
En el caso de 'guardar_archivo', cuando se llame a esta función el nombre de archivo a guardar deberá respetar el formato: heroes_M.csv, heroes_F.csv o heroes_NB.csv según corresponda.
La función retornará True si pudo guardar el archivo, False caso contrario.

3. Tercera Parte

3.1. Basandote en la función 'calcular_min', crear la función 'calcular_min_genero' la cual recibirá como parámetro extra un string que representa el género de la heroína/héroe a buscar. modificar un poco la lógica para que dentro no se traiga por defecto al primer héroe de la lista sino que mediante un flag, se traiga el primer héroe que COINCIDA con el género pasado por parámetro. A partir de allí, podrá empezar a comparar entre héroes o heroínas que coincidan con el género pasado por parámetro. La función retornará el héroe o heroína que cumpla la condición de tener el mínimo (peso, altura u otro dato)

3.2. Basandote en la función 'calcular_max', crear la función 'calcular_max_genero' la cual recibirá como parámetro extra un string que representará el género de la heroína/héroe a buscar. modificar un poco la lógica para que dentro no se traiga por defecto al primer héroe de la lista sino que mediante un flag, se traiga el primer héroe que COINCIDA con el género pasado por parámetro. A partir de allí, podrá empezar a comparar entre héroes o heroínas que coincidan con el género pasado por parámetro. La función retornará el héroe o heroína que cumpla la condición de tener el máximo (peso, altura u otro dato)

3.3. Basandote en la funcion 'calcular_max_min_dato', crear una funcion con la misma lógica la cual reciba un parámetro string que representará el género del héroe/heroína a buscar y renombrarla a 'calcular_max_min_dato_genero'. La estructura será similar a la ya antes creada, salvo que dentro de ella deberá llamar a 'calcular_max_genero' y 'calcular_min_genero', pasandoles el nuevo parámetro. Esta función retornará el héroe o heroína que cumpla con las condiciones pasados por parámetro. Por ejemplo, si se le pasa 'F' y 'minimo', retornará la heroína que tenga el mínimo (altura, peso u otro dato)

3.4. Basandote en la función 'stark_calcular_imprimir_heroe' crear la función ‘stark_calcular_imprimir_guardar_heroe_genero’ que además reciba un string el cual representará el género a evaluar. El formato de mensaje a imprimir deberá ser estilo:

Mayor Altura: Nombre: Gamora | Altura: 183.65

Además la función deberá guardar en un archivo csv **todos los datos del heroe obtenido**.

Reutilizar: 'calcular_max_min_dato_genero', 'obtener_nombre_y_dato', 'imprimir_dato' y 'guardar_archivo'.

En el caso de 'guardar_archivo' el nombre del archivo debe respetar el formato:

heroes_calculo_key_genero.csv

Donde:

- cálculo: representará el string máximo o mínimo
- key: representará cual es la key la cual se tiene que hacer el cálculo
- genero: representará el género a calcular.

Ejemplo: para calcular el héroe más alto femenino, el archivo se deberá llamar:

heroes_maximo_altura_F.csv

Esta función retornará True si pudo guardar el archivo, False caso contrario

4. Cuarta Parte

4.1. Basandote en la función 'sumar_dato_heroe', crear la función llamada 'sumar_dato_heroe_genero' la cual deberá tener un parámetro extra del tipo string que representará el género con el que se va a trabajar.

Esta función antes de realizar la suma en su variable sumadora, deberá validar lo siguiente:

A. El tipo de dato del héroe debe ser diccionario.
B. El héroe actual de la iteración no debe estar vacío (ser diccionario vacío)
C. El género del héroe debe coincidir con el pasado por parámetro.

Una vez que cumpla con las condiciones, podrá realizar la suma. La función deberá retornar la suma del valor de la key de los héroes o heroínas que cumplan las condiciones o 0 en caso de que no se cumplan las validaciones

4.2. Crear la función 'cantidad_heroes_genero' la cual recibirá por parámetro la lista de héroes y un string que representará el género a buscar. La función deberá sumar la cantidad de héroes o heroínas que cumplan con la condición de género pasada por parámetro, retornará dicha suma, **usar las funciones filter y lambda**.

4.3. Basandote en la función 'calcular_promedio', crear la función 'calcular_promedio_genero' la cual tendrá como parámetro extra un string que representará el género a buscar. la lógica es similar a la función anteriormente mencionada en el enunciado. Reutilizar las funciones: 'sumar_dato_heroe_genero', 'cantidad_heroes_genero' y

'dividir'.
retornará el promedio obtenido, según la key y género pasado por parámetro.

4.4. Basandote en la función ‘stark_calcular_imprimir_promedio_altura', desarrollar la función 'stark_calcular_imprimir_guardar_ promedio_altura_genero' la cual tendrá como parámetro extra un string que representará el género de los héroes a buscar.

La función antes de hacer nada, deberá validar que la lista no esté vacía. En caso de no estar vacía: calculará el promedio y lo imprimirá formateado al estilo:

Altura promedio género F: 178.45

En caso de estar vacía, imprimirá como mensaje:

Error: Lista de héroes vacía.

Luego de imprimir la función deberá guardar en un archivo los mismos datos. El nombre del archivo debe tener el siguiente formato:

heroes_promedio_altura_genero.csv

Donde:

A. genero: será el género de los héroes a calcular, siendo M y F únicas opciones posibles.

   Ejemplos:
   heroes_promedio_altura_F.txt
   heroes_promedio_altura_M.txt

Reutilizar las funciones: 'calcular_promedio_genero', 'imprimir_dato' y 'guardar_archivo'.

Esta función retornará True si pudo la lista tiene algún elemento y pudo guardar el archivo, False en caso de que esté vacía o no haya podido

guardar el archivo.

5. Quinta Parte

5.1. Crear la función 'calcular_cantidad_tipo' la cual recibirá por parámetro la lista de héroes y un string que representará el tipo de dato/key a buscar (color_ojos, color_pelo, etc)

Antes de hacer nada, deberá validar que la lista no esté vacía. En caso de estarlo devolver un diccionario con la siguiente estructura:

```
{
    "Error": "La lista se encuentra vacía"
}
```

La función deberá retornar un diccionario con los distintos valores del tipo de dato pasada por parámetro y la cantidad de cada uno (crear un diccionario clave valor).

Por ejemplo, si el tipo de dato fuese color_ojos, devolverá un diccionario de la siguiente manera:

```
{
    "Celeste": 4,
    "Verde": 8,
    "Marron": 6
}
```

Reutilizar la función 'capitalizar_palabras' para capitalizar los nombres de las keys.

5.2. Crear la función 'guardar_cantidad_heroes_tipo' la cual recibirá como parámetro un diccionario que representará las distintas variedades del

tipo de dato (distintos colores de ojos, pelo, etc) como clave con sus respectivas cantidades como valor. Como segundo parámetro recibirá el dato (color_pelo, color_ojos, etc) el cual tendrás que usarlo únicamente en el mensaje final formateado. Esta función deberá iterar cada key del diccionario y variabilizar dicha key con su valor para luego formatearlos en un mensaje el cual deberá guardar en archivo.
Por ejemplo:
"Caracteristica: color_ojos Blue - Cantidad de heroes: 9"

Reutilizar la función 'guardar_archivo'. El nombre del archivo final deberá respetar el formato:

heroes_cantidad_tipoDato.txt

Donde:

- tipoDato: representará el nombre de la key la cual se está evaluando la cantidad de héroes.

  Ejemplo:
  heroes_cantidad_color_pelo.txt
  heroes_cantidad_color_ojos.txt

La función retornará True si salió todo bien, False caso contrario.

5.3. Crear la función 'stark_calcular_cantidad_por_tipo' la cual recibirá por parámetro la lista de héroes y un string que representará el tipo de dato/key a buscar (color_ojos, color_pelo, etc). Dentro deberás reutilizar 'calcular_cantidad_tipo' y 'guardar_cantidad_heroes_tipo' con la lógica que cada una de esas funciones manejan.
Esta función retornará True si pudo guardar el archivo, False caso contrario.

6. Sexta Parte

6.1. Crear la función 'obtener_lista_de_tipos' la cual recibirá por parámetro la lista de héroes y un string que representará el tipo de dato/key a buscar (color_ojos, color_pelo, etc).

Esta función deberá iterar la lista de héroes guardando en una lista las variedades del tipo de dato pasado por parámetro (sus valores).
En caso de encontrar una key sin valor (o string vacío), deberás hardcodear con el valor 'N/A' y luego agregarlo a la lista donde irás guardando todos los valores encontrados, si el valor del héroe no tiene errores, guardarlo tal cual viene.
Finalmente deberás eliminar los duplicados de esa lista y retornarla como un set.
Reutilizar 'capitalizar_palabras' para guardar cada uno de los datos con la primera letra mayúscula.

6.2. Crear la función 'normalizar_dato' la cual recibirá por parámetro un dato de héroe (el nombre de una de sus keys, por ejemplo si la key fuese color_ojos y su valor fuese Verde, recibira este ultimo) y tambien una variable como string la cual representará el valor por defecto a colocar en caso de que el valor está vacío. Deberá validar que el dato no esté vacío, en caso de estarlo lo reemplazará con el valor default pasado por parámetro y lo retornará, caso contrario lo retornará sin modificaciones.

6.3. Crear la función 'normalizar_heroe' la cual recibirá dos parámetros. el primero será un diccionario que representará un solo héroe, el segundo parámetro será el nombre de la key de dicho diccionario la cual debe ser normalizada.
La función deberá capitalizar las palabras que tenga dicha key como valor, luego deberá normalizar el dato (ya que si viene vacío, habrá que setearlo con N/A).
Finalmente deberá capitalizar todas las palabras del nombre del héroe y deberá retornar al Hero con cada palabra de su nombre capitalizados, cada palabra del valor de la key capitalizadas y normalizadas (con N/A en caso de que estuviesen vacías por defecto).
Reutilizar: 'capitalizar_palabras' y 'normalizar_dato'. Ademas Deberá parsear los valores numericos de las keys que posean strings representando un número (Usar RegEx para realizar dicha acción)

6.4. Crear la funcion 'obtener_heroes_por_tipo' el cual recibira por parámetro:

A. La lista de héroes
B. Un set de tipos/variedades (colores de ojos, de pelo, etc)
C. El tipo de dato a evaluar (la key en cuestion, color_ojos, color_pelo, etc)

PRESTAR ATENCIÓN:

A. Deberá iterar el set de tipos/variedades y por cada tipo tendrá evaluar si ese tipo existe como key en un diccionario el cual deberás armar. (contendrá cada variedad como key y una lista de nombres de héroes como valor de cada una de ellas).

B. En caso de no existir dicha key en el diccionario, agregarla con una lista vacía como valor.

C. Dentro de la iteración de variedades, iterar la lista de héroes (for anidado) 'normalizando' el posible valor que tenga la key evaluada, ya que podría venir vacía (qué función usarias aca? guiño guiño)

D. Una vez normalizado el dato, evaluar si dicho dato coincide con el tipo pasado por parámetro.

E. En caso de que coincida, agregarlo a la lista (inicialmente vacía) de la variedad iterada en el primer bucle.
Esta función retornará un diccionario con cada variedad como key y una lista de nombres como valor.

Por ejemplo:

```
{
   "Celestes": ["Capitan America", "Tony Stark"],
   "Verdes": ["Hulk", "Viuda Negra"]
   ....
}
```

6.5. Crear la funcion 'guardar_heroes_por_tipo' la cual recibira por parámetro un diccionario que representará los distintos tipos como clave y una lista de nombres como valor (Lo retorna la función anterior) y además como segundo parámetro tendrá un string el cual representará el tipo de dato a evaluar (color_pelo, color_ojos, etc). Deberá recorrer cada key y cada valor (lista) que esta contenga para finalmente crear un string el cual será un mensaje que deberás imprimir formateado.
Por ejemplo:
"color_ojos Green: Black Widow | Hulk"

Reutilizar la función 'guardar_archivo'. El archivo final deberá respetar el formato:
heroes_segun_TipoDato.txt

Donde:

- TipoDato: es la key la cual indicará qué cosas se deben guardar en el archivo.

Ejemplo:

heroes_segun_color_pelo.csv (Agrupados por color de pelo)
heroes_segun_color_ojos.csv (Agrupados por color de ojos)

Esta función retorna True si salió todo bien, False caso contrario.

6.6. Crear la función 'stark_listar_heroes_por_dato' la cual recibirá por parámetro la lista de héroes y un string que representará el tipo de dato a evaluar (color_pelo, color_ojos, etc). Dentro deberás reutilizar

las funciones:

A. 'obtener_lista_de_tipos'
B. 'obtener_heroes_por_tipo'
C. 'guardar_heroes_por_tipo'

Pasando por parámetro lo que corresponda según la lógica de las funciones usadas.
Esta función retornará True si pudo guardar el archivo, False caso contrario.