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.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_heroes =
[
      {
        "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": "average"
      },
      {
        "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"
      }
]
```

Desafío #03:

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!***

1.1. Crear la función '***extraer_iniciales***' que recibirá como parámetro:
nombre_heroe: un string con el nombre del personaje

La función deberá devolver a partir del parámetro recibido un nuevo string con las iniciales del nombre del personaje seguidos por un punto (.)

En el caso que el nombre del personaje contenga el artículo '*the*' se deberá omitir de las iniciales
Se deberá verificar si el nombre contiene un guión '-' y sólo en el caso que lo tenga se deberá reemplazar por un espacio en blanco

La función deberá validar:

Que el string recibido no se encuentre vacío

Devolver '*N/A*' en caso de no cumplirse la validación

Ejemplo de la salida de la función para Howard the Duck:
H.D.

ATENCIÓN: Usar regex

1.2. Crear la función **obtener_dato_formato'** la cual recibirá como parámetro:

dato: un string con un dato especifico

La función deberá convertir el dato pasado a minúsculas y con formato snake_case

por ejemplo: **Howard the Duck** -> howard_the_duck

La función deberá validar:

Que el dato recibido sea del tipo str

En caso de encontrar algún error retornar False, caso contrario el nombre con el formato especificado

ATENCIÓN: Usar regex

1.3. Crear la función '***stark_imprimir_nombre_con_iniciales***' la cual recibirá como parámetro:

nombre_heroe: un string con el nombre del personaje

Se deberá validar:

Que el dato recibido sea del tipo diccionario
Que el diccionario contengan la clave *'nombre'*

La función deberá imprimir el dato en cuestión con el siguiente formato

Delante de cada nombre se deberá agregar un asterisco '*' (de forma de viñeta) seguido de un espacio.

Si el superhéroe es Howard the Duck se deberá mostrar

*** howard_the_duck (H.W.)**

La función deberá devolver True en caso de haber finalizado con éxito o False en caso de que haya ocurrido un error

1.4 Crear la función '***stark_imprimir_nombres_con_iniciales***' la cual recibirá como parámetro:

lista_heroes: la lista de personajes

La función deberá utilizar la función anterior

Luego deberá imprimir la lista completa de los nombres de los personajes con el mismo formato de la anterior

Se deberá validar:

Que *lista_heroes* sea del tipo lista
Que la lista contenga al menos un elemento

La función retornara **True** si salió todo bien y **False** si ocurrió algún error

2.1. Crear la función '***generar_codigo_heroe***' la cual recibirá como parámetro:

diccionario de un héroe
id_heroe (int)

La función deberá generar un string con el siguiente formato:

GENERO-X00…000ID

Es decir, el género recibido por parámetro seguido de un '-' (guión) y por último el identificador recibido. Todos los códigos generados deben tener como máximo 10 caracteres (contando todos los caracteres, inclusive el guión). Se deberá completar con ceros para que todos queden del mismo largo.

Dependiendo del género el primer número del código variará

En caso de que sea un superhéroe de género M -> el código comenzará en 1

En caso de que sea un superhéroe de género F-> el código comenzará en 2

En caso de que sea un superhéroe de género NB-> el código comenzará en 0

Algunos ejemplos:

```
F-20000001 (10 caracteres)
M-10000002 (10 caracteres)
NB-0000010 (10 caracteres)
```

La función deberá validar:

El género no se encuentre vacío y este dentro de los valores esperados ('M', 'F' o 'NB')

En caso de no pasar las validaciones retornar ***'N/A'***. En caso de verificarse correctamente retornar el código generado.

2.2. Crear la función '***stark_generar_codigos_heroes***' la cual recibirá como parametro:

lista_heroes: la lista de personajes:

La función deberá iterar la lista de personajes y generar cada uno de los códigos

El código del héroe (id_heroe) surge de la posición del mismo dentro de la lista_heroes (comenzando por el 1).

Reutilizar la función *anterior* pasándole como argumentos el id_heroe correspondiente

Guardar en un string cada uno de los heroes con el siguiente formato y al final de toda la iteración debería quedar un mensaje como el siguiente ejemplo

*** howard_the_duck (H.W.) | M-10000001**

*** rocket_raccoon (R.R.) | NB-0000002**

*** wolverine (W.) | M-10000003**

*** black_widow (B.W.) | F-20000004**

……

Se asignaron ## códigos

En donde ## es la cantidad de códigos que se generaron

La función deberá validar::

La lista contenga al menos un elemento
Todos los elementos de la lista sean del tipo diccionario

La función retornara la **cadena generada** si salió todo correctamente y **False** en caso de error

3.1. Crear la función '***sanitizar_entero***' la cual recibirá como parámetro:
numero_str: un string que representa un posible número entero

La función deberá analizar el string recibido y determinar si es un número entero positivo. La función debe devolver distintos valores según el problema encontrado:

Si contiene carácteres no numéricos retornar -1
Si el número es negativo se deberá retornar un -2
Si ocurren otros errores que no permiten convertirlo a entero entonces se deberá retornar -3

También deberá quitar los espacios en blanco de atras y adelante del string en caso que los tuviese

En caso que se verifique que el texto contenido en el string es un número entero positivo, retornarlo convertido en entero

3.2. Crear la función '***sanitizar_flotante***' la cual recibirá como parámetro:
numero_str: un string que representa un posible número decimal

La función deberá analizar el string recibido y determinar si es un número flotante positivo. La función debe devolver distintos valores según el problema encontrado:

Si contiene carácteres no numéricos retornar -1
Si el número es negativo se deberá retornar un -2

Si ocurren otros errores que no permiten convertirlo a entero entonces se deberá retornar -3

También deberá quitar los espacios en blanco de atras y adelante del string en caso que los tuviese

En caso que se verifique que el texto contenido en el string es un número flotante positivo, retornarlo convertido en flotante

3.3. Crear la función '***sanitizar_string***" la cual recibirá como parámetro
valor_str: un string que representa el texto a validar
valor_por_defecto: un string que representa un valor por defecto (parámetro opcional, inicializarlo con '-')

La función deberá analizar el string recibido y determinar si es solo texto (sin números). En caso de encontrarse números retornar "*N/A*"

En el caso que valor_str contenga una barra '/' deberá ser reemplazada por un espacio

El espacio es un caracter válido

En caso que se verifique que el parámetro recibido es solo texto, se deberá retornar el mismo convertido todo a minúsculas

En el caso que el texto a validar se encuentre vacío y que nos hayan pasado un valor por defecto, entonces retornar el valor por defecto convertido a minúsculas

Quitar los espacios en blanco de atras y adelante de ambos parámetros en caso que los tuviese

3.4. Crear la función '***sanitizar_dato***' la cual recibirá como parámetros:

heroe: un diccionario con los datos del personaje

clave: un string que representa el dato a sanitizar (la clave del diccionario). Por ejemplo *altura*

tipo_dato: un string que representa el tipo de dato a sanitizar. Puede tomar los valores: '*string*', '*entero*' y '*flotante*'

La función deberá sanitizar el valor del diccionario correspondiente a la clave y al tipo de dato recibido

Para sanitizar los valores se deberán utilizar las funciones creadas en los puntos 3.1, 3.2, 3.3 y 3.4

Se deberá validar:

Que *tipo_dato* se encuentre entre los valores esperados ('*string, 'entero, 'flotante*)' la validación debe soportar que nos lleguen mayúsculas o minúsculas. En caso de encontrarse un valor no válido informar por pantalla: '*Tipo de dato no reconocido*'

Que *clave* exista como clave dentro del diccionario *heroe*. En caso de no encontrarse, informar por pantalla: 'La clave especificada no existe en el héroe'. (en este caso la validación es sensible a mayúsculas o minúsculas)

Ejemplo de llamada a la función válida:

sanitizar_dato(dict_personaje, "altura", "Flotante")

La función deberá devolver True en caso de haber sanitizado algún dato y False en caso contrario.

3.5. Actualizar la función '***stark_normalizar_datos***' *usando las funciones de sanitizar.*

La función deberá recorrer la lista de héroes y sanitizar los valores solo de las siguientes claves: *'altura'*, '*peso*', '*color_ojos*', '*color_pelo*', '*fuerza*' e '*inteligencia*'

Validar que la lista de héroes no esté vacía para realizar sus acciones, caso contrario imprimirá el mensaje: "Error: Lista de héroes vacía"
Reutilizar la función *sanitizar_dato*

NOTA:El comportamiento de la función deberá ser el mismo del anterior stark

Para las claves color_ojos, color_pelo e inteligencia verificar que la cadena por ejemplo no este vacia y en caso de estar vacia reemplazar lo vacio por "No tiene" (Usar regex para ello)

4.1. Crear la función '***stark_imprimir_indice_nombre***' la cual recibirá como parámetro:

lista_heroes: la lista de personajes

La función deberá mostrar por pantalla cada una de las palabras **(en minúscula)** de cada uno de los nombres que existan en el data_stark separado por un guión entre cada una de las palabras ignorando las palabras que contengan ***"the" -> (Usar regex para separar esto)***

Por ejemplo:

howard-duck-rocket-raccoon-wolverine…

5.1 Crear la función '***generar_separador***' la cual recibirá como parámetro

patron: un carácter que se utilizará como patrón para generar el separador

largo: un número que representa la cantidad de caracteres que va ocupar el separador.

imprimir: un parámetro opcional del tipo booleano (por default definir en True)

La función deberá generar un string que contenga el patrón especificado repitiendo tantas veces como la cantidad recibida como parámetro (uno junto al otro, sin salto de línea)

Si el parámetro booleano recibido se encuentra en False se deberá solo retornar el separador generado. Si se encuentra en True antes de retornarlo, imprimirlo por pantalla

La función deberá validar:

Que el parámetro *patrón* tenga al menos un carácter y como máximo dos

Que el parámetro *largo* sea un entero entre 1 y 235 inclusive

En caso de no verificarse las validaciones devolver '*N/A*'

Ejemplo de llamada:

generar_separador('*', 10)

Ejemplo de salida:

****

5.2 Crear la función '***generar_encabezado***' la cual recibirá como parámetro

titulo: un string que representa el título de una sección de la ficha

La función deberá devolver un string que contenga el título envuelto entre dos separadores (estimar el largo requerido para tu pantalla).

Ejemplo de salida:

****
PRINCIPAL
****

La función deberá convertir el título recibido en todas letras mayúsculas

5.3 Crear la función '***imprimir_ficha_heroe***' la cual recibirá como parámetro:

heroe: un diccionario con los datos del héroe

La función deberá a partir los datos del héroe generar un string con el siguiente formato e imprimirlo por pantalla::

****
PRINCIPAL
****

NOMBRE DEL HÉROE:	spider_man (S.M.)
IDENTIDAD SECRETA:	peter_parker
CONSULTORA:	marvel_comics
CÓDIGO DE HÉROE :	M-10000001

****
FISICO

****

ALTURA: 178 cm.

PESO: 74,25 kg.

FUERZA: 55 N

****

SEÑAS PARTICULARES

****

COLOR DE OJOS: Hazel

COLOR DE PELO: Brown

5.4 Crear la función '***stark_navegar_fichas***' la cual recibirá como parámetros:

lista_heroes: la listas personajes

La función deberá comenzar imprimiendo la ficha del primer personaje de la lista y luego solicitar al usuario que ingrese alguna de las siguientes opciones:

[1] Ir a la izquierda [2] Ir a la derecha [3] Salir

Si el usuario ingresa '1': se debe mostrar el héroe que se encuentra en la posición **anterior** en la lista (en caso de estar en el primero, ir al último)

Si el usuario ingresa '2': se debe mostrar el héroe que se encuentra en la posición **siguiente** en la lista (en caso de estar en el último, ir al primero)

Si ingresa 3: volver al menú principal

Si ingresa cualquier otro valor, volver a mostrar las opciones hasta que ingrese un valor válido

Recordatorio: Se pueden usar índices negativos si desean

6. Crear funciones para el manejo del menú principal (usar de guía el anterior desafío)

El menú principal debe tener las siguientes opciones

1 - Imprimir la lista de nombres junto con sus iniciales

2 - Imprimir la lista de nombres y el código del mismo

3 - Normalizar datos

4 - Imprimir índice de nombres

5 - Navegar fichas

6 - Salir