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"
      }
]
```

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

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!***

Agregar al código elaborado para cumplir el desafío #00 los siguientes puntos, utilizando un menú que permita acceder a cada uno de los puntos por separado.

0.0. Crear la función "*imprimir_menu_2*" que deberá imprimir el menú de opciones por pantalla (las opciones son las mismas del desafío 01). La función no retorna nada. Reutilizar la función *'imprimir_dato'*

0.1. Crear la función *'stark_menu_principal_2'* 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 entero, caso contrario retorna -1. Reutilizar la función *'imprimir_menu_2'*

0.2. Crear la función *'stark_marvel_app_2'* 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 (match solo para los que cuentan con python 3.10+). 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.

1.1. Crear la función '*es género*' la cual recibirá como parámetros:

- Un diccionario que representará un héroe
- Un string el cual será usado para evaluar si el héroe coincide con el género buscado (el string puede ser 'M', 'F' o 'NB').

La función deberá retornar True en caso de que cumpla, False caso contrario.

1.2. Crear la función '*stark_imprimir_heroe_genero*' la cual recibirá como parámetros:

- La lista de héroes
- Un string el cual representará el género a evaluar (el string puede ser 'M', 'F' o 'NB'). Deberá imprimir solamente los héroes que coincidan con el género pasado por parámetro.

Se deberá reutilizar las funciones '*es_genero*', 'obtener_nombre' e '*imprimir_dato*'.

La función no retorna nada.

Con este punto se resuelve el punto A y B del desafío #01

2.1. Crear la función '*calcular_min_genero*' la cual recibirá como parámetros:

- La lista de héroes.
- Un string para representar el dato que deberá ser evaluado a efectos de determinar cuál es el máximo de la lista (por ejemplo: "altura", "peso", etc)
- Un string para representar el género (el string puede ser 'M', 'F' o 'NB')

La función deberá retornar el héroe que cumpla la condición de tener el mínimo del género especificado

2.2. Crear la función '*calcular_max_genero*' la cual recibirá como parámetros:

- La lista de héroes.
- Un string para representar el dato que deberá ser evaluado a efectos de determinar cuál es el máximo de la lista (por ejemplo: “altura”, “peso”, etc)
- Un string para representar el género (el string puede ser ‘M’, ‘F’ o ‘NB’)

La función deberá retornar el héroe que cumpla la condición de tener el máximo del género especificado

2.3. Crear la funcion *'calcular_max_min_dato_genero'* la cual recibiá como parámetros:

- La lista de héroes
- El tipo de cálculo a realizar: es un valor string que puede tomar los valores *‘maximo’* o *‘minimo’*
- Un string que representa la key del dato a calcular, por ejemplo: ‘*altura*’, ‘*peso*’, ‘*edad*’, etc.
- Un string para representar el género (el string puede ser ‘M’, ‘F’ o ‘NB’)

La función deberá retornar el héroe que cumpla con la condición pedida. Reutilizar las funciones hechas en los puntos 2.0 y 2.1

Ejemplo de llamada:
calcular_max_min_dato(lista, "maximo", “F” "edad")

2.4. Crear la función ***'stark_calcular_imprimir_heroe_genero'*** la cual recibirá como parámetros:

- La lista de héroes
- El tipo de cálculo a realizar: es un valor string que puede tomar los valores *‘maximo’* o *‘minimo’*
- Un string para representar el género (el string puede ser ‘M’, ‘F’ o ‘NB’)

- Un string que representa la key del dato a calcular, por ejemplo: '*altura*', '*peso*', '*edad*', etc.

La función deberá obtener el héroe que cumpla dichas condiciones, imprimir su nombre y el valor calculado.

Validar que la lista de héroes no esté vacía para realizar sus acciones, caso contrario no hará nada y retornara -1.

Reutilizar las funciones 'calcular_max_min_dato_genero', 'obtener_nombre_y_dato' y 'imprimir_dato'

Con este punto se resuelve el punto C, D, E y F del desafío #01

3.1. Crear la función '*sumar_dato_heroe_genero*' la cual recibirá como parámetros:

- La lista de héroes
- Un string que representa la key del dato a calcular, por ejemplo: '*altura*', '*peso*', '*edad*', etc.
- Un string para representar el género (el string puede ser 'M', 'F' o 'NB')

La función deberá devolver la suma total del dato recibido como parámetro de todos los héroes del género especificado.

Antes de realizar el cálculo se deberá validar:

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 se podrá realizar la suma.

3.2. Crear la función '*cantidad_heroes_genero*' la cual recibirá como parámetros:

- La lista de héroes
- Un string que representa el género a buscar.

La función deberá iterar y sumar la cantidad de héroes que cumplan con la condición de género pasada por parámetro.

La función deberá retornar la suma de los héroes según el género especificado

3.3. Crear la función '*calcular_promedio_genero*' la cual recibirá como parámetros:

- La lista de héroes
- Un string que representa la key del dato a calcular, por ejemplo: '*altura*', '*peso*', '*edad*', etc.
- Un string que representa el género a buscar.

Se deberá reutilizar las funciones: '*sumar_dato_heroe_genero*', '*cantidad_heroes_genero*' y '*dividir*'

La función deberá retornar el promedio obtenido, según la key y género pasado por parámetro.

3.1. Crear la función '*stark_calcular_imprimir_promedio_altura_genero*' la cual recibirá como parámetros:

- La lista de héroes
- Un string que representa la key del dato a calcular, por ejemplo: '*altura*', '*peso*', '*edad*', etc.
- Un string que representa el género a buscar.

Se deberá validar antes de hacer nada que la lista no esté vacía. En caso de estarlo se deberá imprimir el mensaje: "Error: Lista de héroes vacía"

En caso de no estar vacía, calcular el promedio e imprimir formateado al estilo:
"Altura promedio genero F: 178.45"

Reutilizar las funciones: 'calcular_promedio_genero' e 'imprimir_dato'

<u>Con este punto se resuelve el punto G y H del desafío #01</u>

4.1. Crear la función '*calcular_cantidad_tipo*' la cual recibirá como parámetros:

- La lista de héroes
- Un string que representa el tipo de dato/key a buscar (color_ojos, color_pelo, etc)

Antes de realizar cualquier cálculo se deberá validar que la lista no esté vacía

La función deberá retornar un diccionario con los distintos valores del tipo de dato recibido 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
}
```

4.2. Crear la funcion '*imprimir_cantidad_heroes_tipo*' la cual solamente recibirá como parámetros:

- Un diccionario que representara las distintas variedades del tipo de dato (distintos colores de ojos, pelo, etc) como clave con sus

respectivas cantidades como valor.

- Un string que representa el tipo de dato (color_pelo, color_ojos, etc): el cual tendrás que usarlo únicamente en el mensaje final formateado.

La función deberá iterar cada key del diccionario y variabilizar dicha key con su valor para luego formatearlos en un mensaje el cual deberá imprimir.

Por ejemplo:
"Característica: color_ojos Blue - Cantidad de héroes: 9"

Reutilizar la función 'imprimir_dato'

4.3. Crear la función '*stark_calcular_cantidad_por_tipo*' la cual recibirá como parámetros:

- La lista de héroes
- Un string que representa el tipo de dato/key a buscar (color_ojos, color_pelo, etc).

Dentro deberas reutilizar las funciones: *'calcular_cantidad_tipo*' e '*imprimir_cantidad_heroes_tipo*'

Esta función no retorna nada.

Con este punto se resuelve el punto J, K y L del desafío #01

5.1. Crear la función '*obtener_lista_de_tipos*' la cual recibirá como parámetros:

- La lista de héroes
- Un string que representa el tipo de dato/key a buscar (color_ojos, color_pelo, etc)

La 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 hardcodearlo 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.

5.2. Crear la función '*normalizar_dato*' la cual recibirá como parámetros:

- Un string que representa un dato del héroe (el valor de una de sus keys, por ejemplo si la key fuese color_ojos y su valor fuese '*Verde*', recibirá este último)

- Un string el cual se usará como valor por defecto (en caso que un valor se encuentre vacío)

La función deberá validar que el dato no esté vacío, en caso de estarlo lo reemplazará con el valor por defecto pasado por parámetro y lo retornara, caso contrario lo retornará sin modificaciones.

5.3. Crear la funcion '*obtener_heroes_por_tipo*' el cual recibira como parametros:

- La lista de héroes.
- La lista de tipos/variedades que existen (colores de ojos, de pelo, etc)
- El tipo de dato a evaluar (la key en cuestion, '*color_ojos*', '*color_pelo*', etc)

TENER EN CUENTA:

La función deberá iterar la lista de tipos/variedades y por cada tipo tendrá que evaluar si ese tipo existe como key en un diccionario el cual deberás armar. Este diccionario contendrá cada variedad como key y una lista de nombres de héroes como valor de cada una de ellas

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

- 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)

- Una vez normalizado el dato, evaluar si dicho dato coincide con el tipo pasado por parámetro. En caso de que coincida, agregarlo a la lista (inicialmente vacía) de la variedad iterada en el primer bucle.

La función deberá retornar un diccionario con cada variedad como key y una lista de nombres como valor. Por ejemplo:

```
{
    "Celestes": [“Capitan America”, “Tony Stark”],
    "Verde": [“Hulk”, “Viuda Negra”]
}
```

Reutilizar la función 'normalizar_dato'

5.4. Crear la funcion '*imprimir_heroes_por_tipo*' la cual recibira como parámetros:

- Un diccionario que representara los distintos tipos como clave y una lista de nombres como valor (lo retorna la función anterior)

- Un un string el cual representará el tipo de dato a evaluar (color_pelo, color_ojos, etc)

La función deberá recorrer cada key y cada valor (lista) que esta contenga para finalmente crear un string el cual será un mensaje que deberá imprimir formateado. Se la siguiente manera:

"color_ojos Green: Black Widow | Hulk"

Esta función no tiene retorno.

Reutilizar la función 'imprimir_dato'

5.5. Crear la funcion '*stark_listar_heroes_por_dato*' la cual recibira como parametros:

- La lista de héroes
- Un string que representa el tipo de dato a evaluar (color_pelo, color_ojos, etc)

Dentro deberás reutilizar las funciones, '*obtener_lista_de_tipos*', '*obtener_heroes_por_tipo*', '*imprimir_heroes_por_tipo*'

Esta función no retorna nada.

Con este punto se resuelve el punto M, N y O del desafío #01