Proyecto de laboratorio - Python

Estructura general del proyecto

A. El proyecto debe contener dos carpetas: src y data.
B. La carpeta data debe incluir el fichero CSV del dataset seleccionado.
C. La carpeta src debe incluir el código fuente, organizado en módulos. Habrá al menos un módulo ***modulo1*.py** con las funciones principales y otro módulo ***modulo1*_test.py** con las pruebas. Además, podrá haber otros módulos secundarios ***modulo2*.py**… con funciones auxiliares.
D. El proyecto debe contener un fichero README.md donde se describirán los datos y las funciones implementadas.

Entrega 1 (fecha límite 23/10/2022)

Crear la estructura básica del proyecto.

Rellenar el fichero README.md con los datos personales, la estructura del proyecto, la descripción de las columnas del dataset y las funciones implementadas.

Definir una tupla con nombre para almacenar los datos.

Escribir en el módulo principal una función que lea los datos del fichero y los almacene en una lista de tuplas. Escribir en el módulo de pruebas un test de esta función, visualizando en pantalla el número total de registros leídos, los 3 primeros registros leídos y los 3 últimos registros leídos.

Nota: el tipo deberá tener como mínimo 6 propiedades de tipos variados, incluyendo al menos lo siguiente:

- Una propiedad de tipo entero.
- Una propiedad de tipo real.
- Una propiedad de tipo cadena.
- Una propiedad de tipo lógico.
- Una propiedad de tipo fecha, hora o ambas cosas.

Entrega 2 (fecha límite 20/11/2022)

Bloque I: Implementar, documentar y probar DOS funciones que trabajen sobre el dataset y respondan a preguntas interesantes. La primera se debe escoger entre los tipos (1) y (2) y la segunda debe ser de tipo (3):

1) Función que filtre y/o seleccione una serie de filas y/o columnas del dataset. Ejemplos: funciones `filtra_por_pais` o `filtra_por_paises_y_años` del proyecto Población.
2) Función que cuente el número de valores distintos de un atributo, o que devuelva un conjunto con los valores distintos. Ejemplos: función `numero_nacionalidades_distintas` del proyecto Extranjería (examen de junio de 2019) y función `calcula_paises` del proyecto Población.
3) Función que calcule la suma, el total o la media de una propiedad numérica. Ejemplos: funciones `calcular_total_camas_centros_accesibles` y `calcular_media_coordenadas` del proyecto Centros Sanitarios.

Bloque II: Implementar, documentar y probar TRES funciones que trabajen sobre el dataset y respondan a preguntas interesantes. La primera se debe escoger entre los tipos (4) y (5), la segunda debe ser de tipo (6) y la tercera de tipo (7):

4) Función que obtenga una tupla (o una parte de ella) con el valor máximo o mínimo de una propiedad de entre las tuplas que cumplan una condición.
5) Función que obtenga una lista con las tuplas cuyo valor de una propiedad concreta es igual al máximo o mínimo valor de esa propiedad.
6) Función que obtenga una lista con n tuplas ordenadas de mayor a menor (o de menor a mayor) por una propiedad determinada de entre las que cumplan una condición.
7) Función que devuelva un diccionario que permita agrupar por una propiedad, en el que los valores sean una lista o un conjunto con las tuplas que tienen el mismo valor de esa propiedad. Ejemplos: funciones `agrupar_por_barrio` y `agrupar_por_distrito` del proyecto Extranjería.

Entrega 3 (fecha límite 11/12/2022)

Bloque III: Implementar, documentar y probar CUATRO funciones que trabajen sobre el dataset y respondan a preguntas interesantes. La primera se debe escoger entre los tipos (8) y (9). La segunda se debe escoger entre los tipos (10) y (11), teniendo en cuenta que si se ha escogido el (8), la segunda debe ser del tipo (11) y si se ha escogido la (9), la segunda debe ser del tipo (10). La tercera debe escogerse entre la (12) y la (13) y la cuarta entre la (14) y la (15).

8) Función que devuelva un diccionario que hace corresponder a cada clave el número de tuplas que contienen dicha clave. Ejemplo: función `contar_nombres_por_año` del proyecto Nombres.
9) Función que devuelva un diccionario que hace corresponder a cada clave la suma de los valores de una cierta propiedad de las tuplas que contienen dicha clave. Ejemplo: funciones `total_extranjeros_por_pais` del proyecto Extranjería y `calcular_frecuencias_por_nombre` del proyecto Nombres.
10) Función que devuelva un máximo o mínimo a partir de un diccionario que hace corresponder a cada clave el número de tuplas que contienen dicha clave. Ejemplo: función `genero_mas_presente` del proyecto Videojuegos.
11) Función que devuelva un máximo o mínimo a partir de un diccionario que hace corresponder a cada clave la suma de los valores de una propiedad de las tuplas que contienen dicha clave. Ejemplo: funciones `barrio_con_mas_extranjeros` del proyecto Extranjería y `genero_mas_ventas` del proyecto Videojuegos.
12) Función que devuelva un diccionario que hace corresponder a cada clave el máximo o mínimo de alguna propiedad de las tuplas que contienen dicha clave.
13) Función que devuelva un diccionario que hace corresponder a cada clave el porcentaje de alguna propiedad de las tuplas que contienen dicha clave respecto al total de tuplas. Ejemplo: función `dicc_porcentaje_ventasJP_por_año` del proyecto Videojuegos.
14) Función que devuelva un diccionario que hace corresponder a cada clave una lista ordenada con los n mayores o menores elementos que contienen dicha clave. Ejemplo: función `dicc_top_n_juegos_por_genero` del proyecto Videojuegos.
15) Función que devuelva una lista de incrementos de una propiedad agrupada por otra. Ejemplo: función `incremento_ventas` del proyecto Videojuegos.

Bloque IV: Implementar, documentar y probar UNA función que trabaje sobre el dataset y genere una gráfica interesante, haciendo uso de la biblioteca matplotlib.

Notas:

Sobre las condiciones de los filtros:

- Utilizar filtros variados y no triviales.
- Combinar filtros.
- A modo de ejemplo, se facilitan las siguientes ideas:
    - Que una propiedad o varias tomen unos valores concretos.
    - Que una propiedad sea mayor, menor que un valor concreto o comprendido entre otros dos valores.
    - Que una determinada propiedad esté contenida en un conjunto o lista de valores.
    - Que una determinada propiedad de tipo cadena tenga un formato dado, como por ejemplo comienzar por alguna letra o secuencia de letras.
    - Que el día, mes o año de una fecha tenga un valor dado o esté comprendido entre dos valores dados.

Sobre la escritura del código:

- Organizar el código dividiéndolo en bloques de funciones mediante comentarios.
- Respetar las convenciones de Python para nombrar los identificadores.
- Usar comentarios de documentación en todas las funciones, indicando qué recibe la función y qué devuelve.
- No utilizar caracteres no internacionales (tildes o eñes) en los identificadores.
- Documentar las funciones con el formato utilizado en el proyecto que se proporciona como modelo.

Sobre los elementos del lenguaje:

- Usar siempre que sea posible la tupla con nombre.
- Utilizar en algunas funciones parámetros con valores por defecto, y en ese caso hacer al menos un test para una llamada con parámetro y otra sin parámetro.
- No utilizar módulos que no se hayan usado en clase.
- No debe leerse ningún dato del teclado.

Sobre las pruebas:

- Probar las funciones en el mismo orden en que están definidas.
- En cada prueba, indicar qué función se está probando, qué representa la información que se muestra en pantalla, y mostrar solo algunos elementos de las listas/conjuntos/diccionarios que tengan muchos elementos.
- El test debe limitarse a mostrar los valores devueltos por la función (todos o una parte de ellos), nunca a hacer tratamientos con ellos (del tipo media o suma, por ejemplo); los tratamientos se realizan siempre en las funciones.
- Comprobar que los resultados de los tests son correctos en la medida de lo posible, o al menos que son coherentes. Diseñar siempre que sea posible los test de forma que podamos conocer de antemano el resultado esperado.
- Las funciones no deben imprimir nada, sino limitarse a devolver un resultado. Es en el test donde se imprime.

Sobre los errores:

- Chequear posibles errores de ejecución, como divisiones por cero o acceso a un elemento inexistente de una lista, tupla o diccionario.
- Leer bien los mensajes de error y aprender a interpretarlos. Indican qué error se ha producido y en qué punto del código. Buscar en Google si es preciso. ¡No darse por vencido a la primera!
- Corregir todos los errores antes de realizar una entrega.