Ejercicios de Preparación para Solemne 2

1. Escriba una función en Python que, dada una lista de tuplas representando personas con su nombre y edad, busque y retorne la tupla de la persona especificada por el usuario. Si no se encuentra ninguna tupla con el nombre especificado, el programa debe indicarlo.

2. Escriba una función en Python que reciba como entrada un string y retorne el carácter que más se repite. Si hay varios caracteres con la misma frecuencia máxima, puedes devolver cualquiera de ellos.

3. Usted ha decido armar una empresa de video juegos que pretende vender en 3 continentes (América, Europa y Asia). Antes de comenzar a desarrollar su primer video juego, ha realizado una encuesta para determinar qué géneros son los preferidos en cada uno de estos continentes. Para automatizar el proceso de conteo de preferencias, considere que tiene una lista de tuplas, cada una de ellas para representar a un género de video juego en particular. Además, tiene una segunda lista de tuplas con el conteo de las preferencias de cada continente. Ejemplos de estas listas se muestran a continuación:

```
generos = [('Aventura', 'A'), ('Rol', 'R'), ('Estrategia', 'S'), ('Deportes', 'D')]
votaciones = [('América', 'RASARRD'), ('Europa', 'SSAARRRD'), ('Asia', 'SARRASD')]
```

   i) Escriba una función en Python que reciba como entradas el nombre del continente y la lista votaciones y retorne el carácter que representa al género más votado. Un ejemplo de llamado a la función sería el siguiente:

```
genero = mas_votado('Europa', votaciones)
print(f'El género más votado es {genero}')
```

   El resultado en consola sería el siguiente:

```
>> R
```

   ii) Escriba una función en Python que reciba como entradas las listas géneros y votaciones y retorne el nombre del género más votado considerando los tres continentes. Un ejemplo de llamado a la función sería el siguiente:

```
nombre_genero = mas_votado_total(generos, votaciones)
print(f'El nombre del género más votado es {nombre_genero}')
```

   El resultado en consola sería el siguiente:

```
>> Rol
```

4. Usted ha comenzado a prepararse para participar en la maratón de Santiago que se realizará dentro de 3 meses. Para monitorear sus entrenamientos, ha decidido registrar los tiempos de 20 entrenamientos, en cada uno de los cuales recorre 21 kilómetros.
   En vista del contexto anterior, escriba en Python:

   a. Una función llamada `registrar_tiempos()` que permita registrar 20 mediciones de tiempo (en minutos) y retorne una lista con los datos.
   b. Una función llamada `obtener_promedio(tiempos)` que reciba una lista con mediciones y retorne el promedio de estos.
   c. Una función llamada `generar_archivo(tiempos)` que reciba una lista con mediciones y genere un archivo txt con todos los datos de la lista.
   d. Una función llamada `detectar_super_entrenamientos(tiempos, promedio)` que reciba una lista con mediciones de entrenamientos, un promedio, y retorne una lista con todas las mediciones de entrenamientos que estén por sobre el promedio.

5. Escriba una función en Python que calcule el factorial de un número dado. La función debe manejar correctamente los casos en que el número dado sea 0 o 1, devolviendo 1 como resultado.

6. Escriba una función en Python que calcule el coseno de un número dado (en radianes) utilizando la serie de Taylor (serie de Maclaurin). La función debe sumar términos de la serie hasta que el valor absoluto del término actual sea menor que una tolerancia especificada. Debe utilizar la función factorial del ejercicio anterior.

   La fórmula de la serie de Taylor para estimar el coseno de x es la siguiente:

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \, x^{2n}}{(2n)!}$$

   Un ejemplo de llamado a la función sería el siguiente con una tolerancia de 0.005:

```
import math

# Convertir 45 grados a radianes
x = math.radians(45)
resultado = coseno(x,0.005)
print(f"El valor aproximado de cos(45°) es: {resultado}")
```

   El resultado en consola sería el siguiente:

```
>> El valor aproximado de cos(45°) es: 0.7071032148228457
```

7. La empresa **PyCornerShop** dispone de una aplicación que le permite a sus usuarios conseguir productos del supermercado desde la comodidad del hogar usando sus teléfonos móviles. La información de los repartidores y usuarios se encuentra almacenada en dos listas llamadas `repartidores` y `usuarios`, respectivamente.
La lista usuarios contiene tuplas de 2 elementos: Código del usuario y ubicación del domicilio (tupla con coordenadas):

```
usuarios = [(1221, (5, 2)), (441, (8, 2)), (587, (10, 1)), ...]
```

La lista `repartidores` contiene tuplas de 4 elementos: El nombre del repartidor, su ubicación actual (tupla con coordenadas), disponibilidad (valor booleano) y lista de visitas a usuarios. Esta lista contiene tuplas con el código del usuario y la cantidad de veces que ha sido visitado por el repartidor:

```
repartidores = [('rayo macuin', (10, 2), True, [(1221, 5), (441, 8), (587, 2)]),
                ('reparti dhor', (9, 3), True, [(1221, 2), (441, 5), (587, 3)]),
                ('eliseo al-azar', (5, 5), False, [(1221, 8), (441, 2)]), ...
]
```

Si un repartidor nunca ha visitado a un usuario, no aparecerá en la lista una tupla con el código de ese usuario.

Para que la **PyCornerShop** pueda funcionar de manera eficiente, le solicita a Ud. que implemente una función llamada `buscar_repartidor(usuarios, repartidores, codigo)` que reciba como parámetros las listas antes mencionadas y un código de usuario. La función debe retornar una lista de repartidores disponibles y cercanos, ordenados de manera descendente, según el número de visitas que hayan realizado al usuario indicado. Un repartidor se considera cercano si está ubicado a menos de 4 km del usuario y para esto considere que las tuplas de ubicación tienen valores enteros medidos en km. Finalmente, si no hay repartidores cercanos, la función debe retornar una lista vacía.

Ejemplos:

```
>>> buscar_repartidor(usuarios,repartidores,587)
['reparti dhor', 'rayo macuin']
>>> buscar_repartidor(usuarios,repartidores,441)
['rayo macuin', 'reparti dhor']
>>> buscar_repartidor(usuarios,repartidores,1221)
[]
```

Nota: Para calcular la distancia d entre un usuario ubicado en $(x1, y1)$ y un repartidor ubicado en $(x2, y2)$ se debe utilizar la fórmula: $d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$.

8. El prestamista Vito Corleone, aburrido de perseguir a sus clientes, ha solicitado la ayuda de algún astro de la programación para resolver sus problemas logísticos.
Vito cuenta con la lista clientes, la que tiene tuplas con el nombre de cada cliente, el monto adeudado y la fecha del último pago (también dentro de una tupla).

```
clientes = [('Don Ramon', 3500, (9, 4, 2014)),
            ('Miguel', 2785, (30,10, 2014)),
            ('Cesar', 100, (28, 5, 2015)), ...]
```

Nota: Esto sólo es un ejemplo, considere que la lista puede tener muchos clientes. Sin embargo, asuma que cada nombre de cliente aparece sólo una vez.

   a. Implemente la función `deuda_total(clientes)`, que reciba como entrada la lista de tuplas `clientes`, y retorne la suma de las deudas de todos los deudores.

   b. Implemente la función `mayor_deudor(clientes, ultimo)`, que retorne el nombre del cliente que tiene la mayor deuda y que además su último pago haya sido realizado en el año `ultimo`. Si no hay cliente con pagos en el año último, retorne un string vacío.

9. Cineton, una nueva cadena de cines, está ingresando al mercado cinematográfico. Por esto, necesita de su ayuda para implementar ciertas funciones en Python y con ellas manejar la cartelera. Para ello, se cuenta con la información de la cartelera de cine en una lista de tuplas como cartelera. El formato de cada tupla en la lista es la siguiente:

```
(mes, país, nombre_película, año_filmación, [sala1, sala2, …])
```

Un ejemplo se muestra a continuación:

```
cartelera = [
('febrero','FRANCIA','El muelle', 1962, ['sala1','sala3']),
('febrero','FRANCIA','La dama de honor', 2004, ['sala1','sala4']),
('abril','RUSIA','Padre del soldado', 1964, ['sala3','sala2','sala4']),
('enero','CHILE','Gloria', 2013, ['sala1','sala2']),
('mayo','MEXICO','Cumbres', 2013, ['sala3','sala2']),
('julio','FRANCIA','Melo', 1986, ['sala3','sala1']),
('junio','BELGICA','Rondo', 2012, ['sala4','sala2']),
('marzo','ALEMANIA','Tiempo de Canibales', 2014, ['sala1','sala2']),
('marzo','ALEMANIA','Soul Kitchen', 2009, ['sala3','sala4']),
...]
```

Para entender mejor, en cartelera la película `'Gloria'`, creada en `2013`, se exibirá el mes de `'enero'` en las salas `'sala1'` y `'sala2'`.

a. Desarrolle la función `pelicula_por_pais(cartelera, pais)` que recibe la lista de la cartelera y el nombre de un país, y que retorne la lista con las películas realizadas en dicho país. Cada elemento de esta lista resultante es una tupla con el nombre de la película y el año de filmación.

```
>>> pelicula_por_pais(cartelera, 'FRANCIA')
[('El muelle', 1962), ('La dama de honor', 2004), ('Melo', 1986)]
```

b. Desarrolle la función `peliculas_por_sala(cartelera, sala)` que reciba la lista de la cartelera y la sala donde se exhibirán las distintas películas. Esta función debe retornar una lista de tuplas, donde cada tupla tiene la forma `(mes, listapelis)` con `mes` un string de un mes y `listapelis` una lista con todas las películas que se exhibirán en sala. Solo deberá incluir tuplas en donde `listapelis` contenga al menos una película.

```
>>> pelicula_por_sala(cartelera, 'sala1')
[('enero', ['Gloria']), ('febrero', ['El muelle', 'La dama de
honor']),('marzo', ['Tiempo de Canibales']), ('julio', ['Melo'])]
```

c. Desarrolle la función `mas_antigua(cartelera)` que retorne el nombre de la película y el país donde fue filmada la película más antigua. Si dos o más películas son las mas antiguas, seleccione cualquiera.

```
>>> mas_antigua(cartelera)
('El muelle', 'FRANCIA')
```