Instrucciones:

- *Resuelve cada uno de los problemas* ***en hojas separadas****.*
- *En la resolución* ***no puedes utilizar*** *elementos del lenguaje Python no estudiados en la asignatura, como son las tuplas, los diccionarios, las listas por comprensión, las estructuras* `for-else`*, los métodos* `index` *o* `count`*, etc.*

Problema 1 (5 puntos)

Disponemos de una serie de matrices que almacenan información sobre las reservas de agua en todos los embalses de España. Cada matriz tiene dimensiones $N \times D$, siendo N el número de embalses de los que se tiene información, y D el número de días del año (por simplicidad, consideraremos que es 365 en todos los casos). Cada celda `[i][j]` de la matriz contiene un número real entre 0 y 100 que indica el porcentaje de agua en el embalse `i` el día `j`. Algunas celdas de la matriz contienen el valor `None`, lo que indica que el dispositivo de medida del nivel del agua falló ese día.

Implementa en Python:

a) (1,5 PUNTOS) Una función, `peor_embalse`, que tenga como parámetro una matriz como la descrita con la información de un año completo y devuelva como resultado el número de embalse cuyo promedio anual de agua embalsada sea más bajo. Para calcular el promedio, debes ignorar las celdas que contengan el valor `None`.

b) (1,5 PUNTOS) Una función, `día_mayor_aumento`, que tenga como parámetros dos matrices como las descritas con la información de dos años completos (`matriz1` y `matriz2`) y que devuelva como resultado el día (valor de 0 a 364) para el que el promedio de agua almacenada en todos los embalses sufrió un mayor incremento en `matriz2` con respecto a `matriz1`. Al igual que en la función anterior, para calcular el promedio debes ignorar las celdas que contengan el valor `None`. Si en ningún día del año hubo un incremento del agua embalsada con respecto al año anterior, la función devolverá `None`. Supón (sin comprobarlo) que las dos matrices tienen las mismas dimensiones.

c) (2 PUNTOS) Una función, `sistema_defectuoso`, que tenga como parámetros una matriz como la descrita con la información de un año completo y un valor `T` y que devuelva como resultado una lista con los embalses (identificados por su número de 0 a $N-1$) para los que el sistema de medida falló (registró `None`) durante `T` o más días consecutivos.

Problema 2 (5 puntos)

Tenemos una serie de ficheros de texto con información acerca de los deportistas de alto rendimiento que actualmente cursan estudios en diversas universidades españolas. Cada línea de uno de estos ficheros (correspondiente a una universidad) contiene, separados por el carácter `#`, el DNI del deportista, la modalidad deportiva que practica y la titulación en que está matriculado. A continuación se muestran, a modo de ejemplo, algunas líneas de un fichero como el descrito:

```
12345678A#Atletismo#Ingeniería Informática
24681357B#Rugby#Medicina
33468648C#Ciclismo#Matemática Computacional
56523475D#Voleibol#Derecho
65234523E#Ciclismo#Ingeniería Informática
```

Considera la clase `Deporte`, definida como sigue:

```
class Deporte:
    def __init__(self, nombre):
        self.nombre = nombre   # Modalidad deportiva
        self.estudiantes = []  # DNI de los estudiantes que practican esa modalidad deportiva
```

Implementa en Python:

a) (2 PUNTOS) Una función, `crear_lista_deportes`, que tenga como parámetros el nombre de un fichero de texto como el descrito anteriormente y el nombre de una titulación. La función debe crear y devolver una lista de objetos de la clase `Deporte`, de forma que cada modalidad deportiva practicada por estudiantes de esa titulación aparezca una única vez en la lista. El atributo `estudiantes` de cada objeto debe contener los DNI de todos los deportistas de esa titulación que practican esa modalidad.

Nota: Solo se permite recorrer el fichero una vez.

b) (1 PUNTO) Una función, `modalidad_más_representada`, que tenga como parámetro una lista de objetos como la construida en el apartado anterior y devuelva el nombre de la modalidad deportiva con mayor número de estudiantes de alto rendimiento. En caso de empate entre varias modalidades deportivas, la función puede devolver cualquiera de ellas.

c) (2 PUNTOS) Una función, `hay_representantes`, que tenga como parámetros una lista de objetos como la construida en el primer apartado y otra lista de cadenas con nombres de modalidades deportivas. La función debe devolver `True` cuando la titulación cuente con representantes para todas las modalidades deportivas indicadas en la lista dada y `False` en caso contrario.