Deusto Facultad de Ingeniería Ingeniaritza Fakultatea	EXAMEN DE PROGRAMACIÓN I	25-NOVIEMBRE-2022

Nombre: _____

DNI: _____ *Aula:*_____ *Fila:*_____ *Columna* _____

Instrucciones

- El ejercicio a pie de máquina se entregará al terminar el examen. Se recogerán en pendrive por parte del profesor/a.
- NO está permitida la conexión a internet en el examen. Tampoco la utilización de tu propio portátil ni la consulta de apuntes o de otros códigos.
- Guarda el código de todos los ejercicios en un fichero que se llame “Examen2-NombreApellidos.py”, por ejemplo: Examen2-PabloGaraizar.py.

Ejercicio 1: Gestión de libros (7 puntos)

Crea un fichero ‘libros.csv’ con la información de 8 libros (puedes inventarte los datos) almacenando el ISBN (int) en la primera columna, el título (str) en la segunda columna y la autoría (str) en la tercera columna, por ejemplo:

```
9788427050297;Si tu quieres, te bajas la Luna; Luna Javierre;
```

Programa una función cargar_libros( ) que reciba como argumento una lista de libros vacía y la llene con los datos de los libros almacenados en el fichero ‘libros.csv’ (1.5 puntos).

Puedes probar tu código así:

```
libros = []
cargar_libros(libros)
print(libros)
```

Programa una función crear_prestamos( ) que reciba como argumento una lista de libros llena y una lista de usuarios llena y devuelva un diccionario en el que las claves sean los nombres de usuario y los valores sean las listas de libros que han pedido prestados esos usuarios. Simula dentro de esta función 1000 préstamos eligiendo un usuario aleatorio de la lista de usuarios y un libro aleatorio de la lista de libros y añadiendo ese libro a la lista correspondiente al usuario dentro del diccionario que tiene que devolver esta función (2.5 puntos).

Puedes probar tu código así:

```
usuarios = ['garaizar', 'bosanz', 'mlguenaga', 'ablago', 'jfajardo']
cargar_libros(libros)
prestamos = crear_prestamos(libros, usuarios)
print(prestamos)

# debería mostrar un diccionario con:
# clave ‘garaizar’ y valor la lista de todos los libros que ha pedido,
# clave ‘bosanz’ y valor la lista de todos los libros que ha pedido,
# clave ‘mlguenaga’ y valor la lista de todos los libros que ha pedido, etc.
```

Si no sabes cómo programar esta función, puedes hacer que devuelva este resultado (no contabilizará, pero te permitirá hacer los siguientes apartados):

```
resultado = {
            'garaizar': [],
            'bosanz': [],
            'mlguenaga': [],
            'ablago': [],
            'jfajardo': [
                  {'isbn': 1, 'titulo': 'Oasis', 'autoría': 'anonimo'},
                  {'isbn': 2, 'titulo': 'Estrellas', 'autoría': 'anonimo'},
            ]
      }
```

Programa una función `usuario_mas_prestamos()` que reciba un diccionario de prestamos (clave: usuario, valor: lista de libros) y devuelva el nombre del usuario que más préstamos ha realizado (1.5 puntos)

Programa una función `guardar_prestamos()` que reciba un diccionario de prestamos (clave: usuario, valor: lista de libros) y guarde en el fichero 'prestamos.csv' el nombre de cada usuario y el total de libros que ha pedido prestados (1.5 puntos), por ejemplo:

```
araizar;199;
bosanz;178;
mlguenaga;204;
ablago;205;
jfajardo;214;
```

Ejercicio 2: Listas multidimensionales (3 puntos)

Programa una función `crear_tabla()` que no reciba nada como argumento y devuelva una lista bidimensional de 10 filas y 10 columnas llena de números aleatorios del 1 al 6. Recuerda usar `random.randint()` para generar esos números (1 punto)

Programa una función `limpiar_tabla()` que reciba una lista de números bidimensional y un número n como argumento y devuelva el resultado de eliminar el número n en toda la lista de números recibida. Si una de las filas de la lista de números recibida contuviera únicamente el número n varias veces, esa fila no debería incluirse en el resultado (2 puntos)

Comprueba tu código así:

```
tabla = crear_tabla()
print(tabla)
print(limpiar_tabla(tabla, 3)) # debería mostrar la tabla sin los 3s

tabla2 = [[3,3,3], [1,3,3]]
print(limpiar_tabla(tabla2, 3)) # debería mostrar [[1]]
```