Nota: Los archivos a los que se hace referencia en los distintos ejercicios están en el archivo comprimido que puedes descargar en la UT.

1. Diseñar una aplicación Python que lea e imprima en pantalla el contenido del archivo de texto ***operaciones.txt***. Controla las excepciones que se puedan producir.

2. Diseñar una aplicación Python que muestre numeradas las líneas del archivo ***operaciones.txt***. El contenido a mostrar debe coincidir con la imagen de la derecha.

```
Unidad_03_Archivos/ejercicios_unidad
1: operacion:codigo_producto:precio
2: update:S10_1678:100.2
3: delete:S11_1111
4: update:s20_2022:1000.2
5: delete:s29_1220
6: delete:S10_2016
7: delete:211_2322
8: delete:S10_1678
```

3. Diseñar una aplicación Python que añada líneas al archivo ***operaciones.txt.*** Las líneas que debe añadir al archivo están contenidas en el archivo ***mas_operaciones.txt.*** Controla las excepciones que se puedan producir.

4. Diseñar una aplicación Python que lea el contenido del archivo ***operaciones.txt*** y guarde cada línea en una lista. Mostrar en pantalla las líneas que incluyan la palabra ***delete***.

5. Diseña una clase llamada ***GestionArchivo***. Esta clase contendrá los siguientes métodos estáticos con los argumentos archivo y cadena:
    a. Buscar cadena en archivo. Retorna True si la cadena existe en el archivo.
    b. Contar repeticiones de cadena en archivo. Retorna el número de veces que aparece la cadena en el archivo.
    c. Mostrar las líneas del archivo en las que aparece una cadena dada. Indicar el número de línea. Retornar las parejas número_linea:contenido_línea.

6. Escribir un programa Python que lea cada línea del archivo ***departamentos.csv*** y las muestre en pantalla excepto las cabeceras. Muestra también el número de filas sin contar las cabeceras y los nombres de las columnas.

7. En este ejercicio partimos del archivo ***summer_olympic_medals.csv*** que contiene las medallas ganadas por cada país en la historia de las olimpiadas (hasta 2020). Con estos datos, debemos diseñar una aplicación como se detalla a continuación:

    Con este ejercicio, se pretende leer una sola vez el archivo, cargar los datos en memoria como objetos, y procesar esos datos sin tener que volver a leer el archivo.

    a. Convertir cada línea del archivo csv en un objeto **ResultadoJuego**. Esta clase tendrá como atributos las columnas del archivo csv. Necesitaremos por tanto, una lista donde guardar estos objetos y una clase llamada **GestionJuegos** que contenga esta lista.

b. Añade métodos a esta clase GestionJuegos para:
   i. Cargar los datos desde el archivo csv y convertirlos en objetos ResultadoJuego. Esto debe hacerse desde el constructor.
   ii. Obtener los datos de la participación de un país concreto cuyo código (columna country_code) se pasará como argumento.
   iii. Retornar el total de medallas de un país en unos juegos. Pasar código de país y año de los juegos.
   iv. Generar un archivo csv con los datos en los juegos de un país del que se conoce su código de país.

8. Partimos en esta ocasión del archivo ***user_behavior_dataset.csv*** que guarda datos relacionados con el uso de aplicaciones móviles. Se pide diseñar una clase con métodos estáticos que permitan:
   a. Obtener un archivo csv con los datos de ese archivo filtrados por tipo de dispositivo, columna “Device Model”. Incluir también las cabeceras.
   b. Obtener un diccionario con las parejas tipo de sistema operativo y número de entradas en el archivo.
   c. Obtener un archivo csv sólo con las columnas ***App Usage Time (min/day) y Gender.*** Añadir cabeceras.

9. Desarrollar una aplicación Python que partiendo del archivo ***summer_olympic_medals.csv,*** que contiene las medallas ganadas por cada país en la historia de las olimpiadas (hasta 2020), sea capaz de obtener un diccionario con los datos de un país concreto en la historia de los juegos. Este diccionario debe estar formado por las parejas **año_juegos:[nº oros, nº platas, nº bronces].** Observa la imagen adjunta con los datos de España.

   Añade la opción de crear un documento JSON a partir del diccionario creado. Llámalo ***country_code.json.***

```
1900: [1, 0, 0]
1920: [0, 2, 0]
1928: [1, 0, 0]
1932: [0, 0, 1]
1948: [0, 1, 0]
1952: [0, 1, 0]
1960: [0, 0, 1]
1972: [0, 0, 1]
1976: [0, 2, 0]
1980: [1, 3, 2]
1984: [1, 2, 2]
1988: [1, 1, 2]
1992: [13, 7, 2]
1996: [5, 6, 6]
2000: [3, 3, 5]
2004: [3, 11, 6]
2008: [5, 11, 3]
2012: [4, 10, 5]
2020: [3, 8, 6]
```

10. A partir de esta lista, que guarda producto, kilos vendidos y precio

```
ventas = [
    ["tomate", 1000, 1.5],
    ["lechuga", 500, 0.5],
    ["cebolla", 300, 0.75],
    ["tomate", 2000, 1.5],
    ["lechuga", 1000, 0.5],
    ["cebolla", 600, 0.75],
```

```
    ["pera", 300, 1.75],
    ["manzana", 500, 1.25],
    ["uva", 1000, 2.5],
    ["uva", 2000, 2.5],
]
```

Debemos diseñar una aplicación Python que genere un archivo JSON en el que se guarde cada producto y el importe total acumulado el mismo.

11. En este ejercicio leeremos el archivo JSON ***people.json,*** que contiene datos de personas, y mostraremos nombre completo, edad y género de cada una de estas personas como se muestra en la imagen de la derecha.

```
Nombre: James Smith
Edad: 32
Género: male
```

12. Quiz. En este ejercicio simularemos un concurso. Se elegirán 10 preguntas al azar entre las contenidas en el archivo ***questions.json***. El usuario deberá dar una respuesta a cada pregunta y la aplicación deberá indicar cuántas acertó y el total de puntos obtenidos (1 punto por acierto).

    Mejoras:
    - Añadir archivo config.json con número de preguntas y valor de cada pregunta.
    - Añadir un archivo top3.json con nombre y puntos de los 3 jugadores con mayor puntuación.

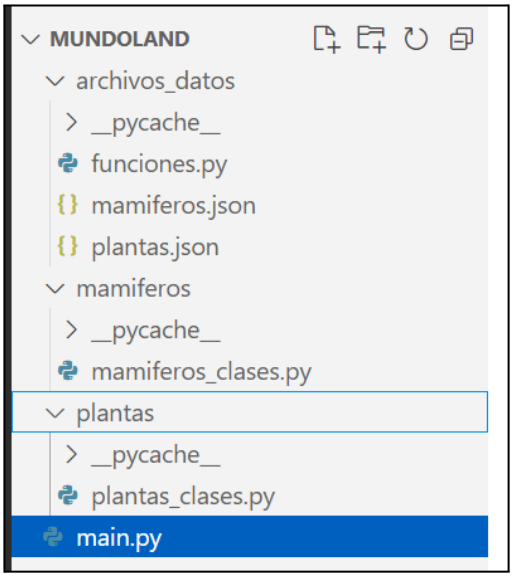
13. Probando Pickle. Diseñar una aplicación Python que trabaje con objetos de la clase ProgrammingLanguage (lenguaje de programación) y RepoLanguage (repositorio de lenguajes) tal y como se indica:
    a. La aplicación mostrará un menú con las opciones:
        i. Añadir lenguaje.
        ii. Eliminar lenguaje.
        iii. Contar lenguajes.
        iv. Mostrar lenguajes.
        v. Salir de la aplicación.
    b. Una vez que finalice la ejecución de una opción se volverá a mostrar el menú, a menos que se elija salir.
    c. La clase ProgrammingLanguage.
        i. Atributos: nombre, extensión y descripción.
        ii. Métodos: __str__, __repr__ e __eq__.
        iii. Dos lenguajes de programación son iguales si tienen el mismo nombre.
    d. La clase RepoLanguage:
        i. Atributos: lista de objetos ProgrammingLanguage

ii. Métodos: nuevo lenguaje, eliminar lenguaje, contar lenguajes y retornar lenguajes.
iii. Añade el método save() para que al finalizar la aplicación, la lista se vuelque en un archivo binario.
iv. Al iniciarla, en el constructor de esta clase, se buscará el archivo binario, si existe se cargarán sus datos en la lista. En la primera ejecución, si el archivo no existe, la lista se creará vacía.
v. Añade los métodos que consideres necesarios para llevar a cabo estas tareas.

14. Desarrollar una aplicación Python que trabaje con el archivo LaLiga2324.csv. Este archivo contiene los resultados de la liga española en su temporada 23/24. El trabajo de la aplicación consistirá en:
    a. Convertir los datos del archivo en objetos Partido.
    b. Crear una colección de partidos y a partir de esa colección obtener.
        i. Un archivo CSV con los partidos de un equipo concreto jugados como local y ordenados ascendentemente por jornada.
        ii. Un archivo JSON con los resultados de un equipo en cada jornada. El diccionario que debemos guardar en el archivo JSON tiene como clave la jornada y como valor un diccionario con los atributos y valores del objeto partido (partido.__dict__). Observa la imagen de la derecha.

        iii. Un archivo CSV con la clasificación de la liga. Recuerda que cada victoria supone 3 puntos y cada empate 1 punto.

15. Paquetes y módulos. Desarrollaremos una aplicación Python que servirá para mostrar una lista de animales y otra de plantas contenidas en sendos archivos json. Echa un vistazo a la estructura del directorio de la aplicación.
    a. El directorio ***archivos_datos*** contiene los dos archivos json y el archivo ***funciones.py*** con dos funciones para leer datos de esos archivos y retornarlos.
    b. El directorio ***mamíferos*** contiene el archivo ***mamiferos_clases.py*** que almacena la clase ***Mamiferos***.
    c. De la misma forma, el directorio ***plantas*** guarda el archivo ***plantas_clases.py*** que describe la clase ***Plantas***.



d. El código de estos dos archivos es el siguiente:

```python
class Mamiferos:
    def __init__(self):
        self.mamiferos = cargar_mamiferos()
```

```python
class Plantas:
    def __init__(self):
        self.plantas = cargar_plantas()
```

Debes descargar el proyecto y completar el código que falta para que se muestren los datos de los dos archivos. El objetivo es realizar las importaciones correctas y completar las funciones.