Tuplas

1. Crea una tupla con números, pide un numero por teclado e indica cuantas veces se repite.

2. Crea una tupla con valores ya predefinidos del 1 al 10, pide un índice por teclado y muestra los valores de la tupla.

3. Escribir un programa que permita procesar datos de pasajeros de viaje en una lista de tuplas con la siguiente forma: (nombre, dni, destino). Ejemplo:

```
[("Manuel Juarez", 19823451, "Liverpool"),
 ("Silvana Paredes", 22709128, "Buenos Aires"),
 ("Rosa Ortiz", 15123978, "Glasgow"),
 ("Luciana Hernandez", 38981374, "Lisboa")
]
```

Además, en otra lista de tuplas se almacenan los datos de cada ciudad y el país al que pertenecen. Ejemplo:

```
[("Buenos Aires", "Argentina"),
 ("Glasgow", "Escocia"),
 ("Lisboa", "Portugal"),
 ("Londres", "Inglaterra"),
 ("Madrid", "España")
]
```

Hacer un programa que permita al usuario realizar las siguientes operaciones:

-Agregar pasajeros a la lista de viajeros.
-Agregar ciudades a la lista de ciudades.
-Dado el DNI de un pasajero, emitir a qué ciudad y país viaja.
-Dado un país, mostrar cuántos pasajeros viajan a ese país.
-Salir del programa.

Conjuntos

1. Diseña un programa que reciba dos conjuntos y devuelva los elementos comunes a ambas, sin repetir ninguno. Ejemplo: si recibe los conjuntos [1, 2, 1] y [2, 3, 2, 4], devolverá 2.

2. Diseña un programa que reciba dos conjuntos y devuelva los elementos que pertenecen a una o a otra, pero sin repetir ninguno. Ejemplo: si recibe los conjuntos [1, 2, 1] y [2, 3, 2, 4], devolverá el conjunto [1, 2, 3, 4].

3. Diseña un programa que reciba dos conjuntos y devuelva los elementos que pertenecen al primero pero no al segundo, sin repetir ninguno. Ejemplo: si recibe las listas [1, 2, 1] y [2, 3, 2, 4], devolverá la lista [1].

4. Diseña un programa que facilite el trabajo con conjuntos. Recuerda que un conjunto es una lista en la que no hay elementos repetidos. Debes implementar:
    - lista_a_conjunto(lista): Devuelve un conjunto con los mismos elementos que hay en lista, pero sin repeticiones. (Ejemplo: lista_a_conjunto([1,1,3,2,3]) devolverá la lista [1, 2, 3] (aunque también se acepta como equivalente cualquier permutación de esos mismos elementos, como [3,1,2] o [3,2,1]).
    - union(A, B): devuelve el conjunto resultante de unir los conjuntos A y B.
    - interseccion(A, B): devuelve el conjunto cuyos elementos pertenecen a A y a B.
    - diferencia(A, B): devuelve el conjunto de elementos que pertenecen a A y no a B.
    - iguales(A, B): devuelve cierto si ambos conjuntos tienen los mismos elementos, y falso en caso contrario.

Diccionarios

1. Crea un diccionario donde la clave sea el nombre del usuario y el valor sea el teléfono (no es necesario validar). Tendrás que ir pidiendo contactos hasta el usuario diga que no quiere insertar más. No se podrán ingresar nombres repetidos.

2. Crear un programa donde vamos a declarar un diccionario para guardar los precios de las distintas frutas. El programa pedirá el nombre de la fruta y la cantidad que se ha vendido y nos mostrará el precio final de la fruta a partir de los datos guardados en el diccionario. Si la fruta no existe nos dará un error. Tras cada consulta el programa nos preguntar á si queremos hacer otra consulta.

3. Codifica un programa que permita guardar los nombres de alumnos de una clase y las notas que han obtenido. Cada alumno puede tener distinta cantidad de notas. Guarda la información en un diccionario cuya claves serán los nombres de los alumnos y los valores serán listas con las notas de cada alumno. El programa pedirá el número de alumnos que vamos a introducir, pedirá su nombre e irá pidiendo sus notas hasta que introduzcamos un número negativo. Al final el programa nos mostrará la lista de alumnos y la nota media obtenida por cada uno de ellos. Nota: si se introduce el nombre de un alumno que ya existe el programa nos dará un error.