Trabajo Práctico 2: Estructuras Modernas en Python

Ejercicio 1

A partir de una lista de palabras, creá una lista con aquellas que tengan más de 4 letras, usando **comprensión de listas**.

```
palabras = ["sol", "luna", "mar", "montaña", "río", "estrella"]
# Resultado esperado: ["montaña", "estrella"]
```

Ejercicio 2

Generar una matriz de 3x3 con los números del 1 al 9 usando **comprensión de listas** anidadas.

```
# Resultado esperado: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Ejercicio 3

Implementar una función que reciba una cadena que indique el día de la semana y devuelva si es “Laboral”, “Fin de semana” o “No válido”. Usar **match.**

Ejercicio 4

Dado un diccionario que representa un cliente con posibles claves "nombre", "edad" y "profesion". para identificar si:

- Es mayor de edad (edad >= 18)
- Es menor
- No se indica la edad

crear la función *clasificar_cliente(cliente)*

```
# Ejemplo: {"nombre": "Ana", "edad": 17} → "Menor de edad"

# {"nombre": "Carlos", "edad": 21, "profesion": "médico"} → "Mayor de edad"

# {"nombre": "Lucía"} → "Edad no especificada"
```

Ejercicio 5

Dada una lista de palabras, usá **enumerate** para obtener una lista con los índices de las palabras que tienen más de 5 letras.

```
palabras = ["hola", "murciélago", "sol", "universo"]
# Resultado esperado: [1, 3]
```

Ejercicio 6

Dadas dos listas del mismo tamaño, usá **zip** para obtener una lista con la suma de los elementos correspondientes.

```
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]
# Resultado esperado: [5, 7, 9]
```

Ejercicio 7

Dadas dos listas, una con nombres y otra con apellidos, usá **zip** para generar una lista con los nombres completos.

```
nombres = ["Ana", "Luis", "Carla"]
apellidos = ["Pérez", "Gómez", "Ruiz"]
# Resultado esperado: ["Ana Pérez", "Luis Gómez", "Carla Ruiz"]
```