(TODOS LOS EJERCICIOS DEBEN INCLUIR CÓDIGO DE PRUEBA DE LA SOLUCIÓN PROPUESTA).

1. Ejercicio sobre listas y acceso mediante notación slice. (1’5 puntos).

Crea una lista de 10 números enteros. Luego, implementa tres funciones utilizando notación *slice* para obtener:

1. Los tres últimos elementos de la lista.
2. Los elementos con posiciones pares en la lista.
3. Los elementos con posiciones impares en la lista.

2. Ejercicio sobre diccionarios. (1’5 puntos).

Crea un diccionario con 5 elementos donde las claves sean nombres de estudiantes y los valores sean sus calificaciones. Luego, implementa una función que imprima los estudiantes con calificación mayor a 7.

3. Ejercicio sobre funciones con parámetro(s) por defecto. (1’5 puntos).

Calcular la edad promedio de tres personas mediante una función (3 parámetros y el último por defecto 18). Utiliza una llamada a la función con 2 edades entradas por teclado y otra llamada con 3 entradas por teclado.

4. Ejercicio sobre excepciones. (1’5 puntos).

Crea una **función** que pida un número al usuario y lo divida entre 100, manejando las siguientes excepciones del tipo que correspondan:

- Si el usuario no introduce un número entero.
- Si el usuario introduce 0, evitando la división por cero.

Asegúrate de que, independientemente de si ocurre un error o no, la función **muestre un mensaje al final de la ejecución**.

5. Ejercicio de clase Persona. (2 puntos).

Crea una clase Persona con las siguientes características:

- Debe tener **atributos privados** __nombre y __edad.
- Implementa su **constructor** para inicializar los atributos.
- Implementa **getters y setters utilizando el correspondiente decorador con @,** con las siguientes validaciones:
  - nombre debe ser una **cadena de texto (str)**, de lo contrario, se debe lanzar una excepción del tipo que corresponda.
  - edad debe ser un **número entero positivo (int > 0)**, de lo contrario, se debe lanzar una excepción del tipo que corresponda.

6. Ejercicio de clases y herencia. (2 puntos).

Crea una **jerarquía de clases** donde:

- Animal sea la **clase base**, con:
  - Un **atributo nombre** inicializado en el constructor.
  - Un método hacer_sonido(), que retorne un mensaje genérico.
- Perro y Gato sean **subclases** de Animal, con:
  - Un constructor que llame al constructor de Animal.
  - Un atributo adicional:
    - Perro debe tener raza.
    - Gato debe tener color_pelaje.
  - Sobrescriben el método hacer_sonido() para devolver sonidos específicos:
    - Perro debe devolver "Guau!".
    - Gato debe devolver "Miau!".
- Implementa una prueba creando instancias de Perro y Gato, mostrando sus atributos y llamando al método hacer_sonido().