**5Pregunta 0: Aplicar los principios básicos de la programación orientada a objetos (POO) en Python para resolver un problema matemático (el factorial) utilizando clases, atributos, métodos y objetos.**

**Instrucciones:**

1. Crea una clase llamada **Factorial**.
2. Define un **atributo privado** \_\_numero que almacene el número del cual se calculará el factorial.
3. Crea un **método constructor (\_\_init\_\_)** que reciba el número y lo asigne al atributo.
4. Implementa un **método público** llamado calcular() que:
   * Calcule el factorial del número almacenado.
   * Devuelva el resultado.
5. Crea otro método llamado mostrar\_resultado() que imprima el número y su factorial.
6. Finalmente, **crea un objeto** de la clase y muestra el resultado.

Resultado esperado:



**Pregunta 1: Atributos privados y métodos getters/setters**

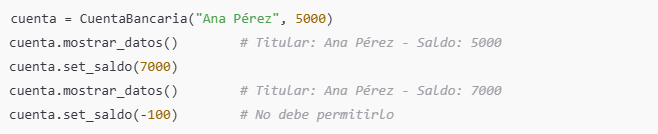
**Enunciado:**

Crea una clase llamada **CuentaBancaria** que tenga los siguientes atributos privados:

* \_\_titular (nombre del titular)
* \_\_saldo (cantidad de dinero en la cuenta)

La clase debe cumplir con lo siguiente:

1. Tener un **constructor** que reciba el nombre del titular y el saldo inicial.
2. Implementar métodos **getters y setters** para acceder y modificar el saldo, asegurando que:
   * No se permita asignar un saldo negativo.
3. Incluir un método público mostrar\_datos() que muestre el nombre del titular y el saldo actual.

Resultado esperado:

**Pregunta 2: Uso de @property para encapsulamiento moderno**

**Enunciado:**

Crea una clase llamada **Producto** con los atributos privados:

* \_\_nombre
* \_\_precio

La clase debe:

1. Definir propiedades con @property y @setter para acceder y modificar el atributo precio.
2. Evitar que se establezca un precio menor o igual a 0.
3. Tener un método mostrar\_producto() que muestre el nombre y el precio.

Resultado esperado:



**Pregunta 3. Implementar la solución al problema clásico de las Torres de Hanoi utilizando programación orientada a objetos en Python y mostrar gráficamente el proceso de resolución usando la biblioteca Tkinter.**

**Enunciado:**

1. Crea una clase llamada **TorresDeHanoi** que tenga los siguientes atributos:
   * num\_discos: número de discos.
   * torres: estructura para representar las tres torres (por ejemplo, listas que contienen discos).
2. La clase debe incluir los siguientes métodos:
   * \_\_init\_\_(self, num\_discos): constructor que inicialice las torres con todos los discos en la torre 1 (de mayor a menor).
   * mover\_disco(self, origen, destino): método para mover el disco superior de una torre a otra, actualizando la estructura interna.
   * resolver(self, n, origen, destino, auxiliar): método recursivo que resuelva el problema para n discos, moviendo discos entre torres.
   * mostrar\_estado(self): método para mostrar el estado actual de las torres (puede imprimir en consola o actualizar la interfaz gráfica).
   * Métodos necesarios para integrar con Tkinter y actualizar la vista gráfica tras cada movimiento.
3. Crea una interfaz gráfica con **Tkinter** que visualice:
   * Las tres torres.
   * Los discos apilados en las torres.
   * Animación o actualización visual en cada movimiento del algoritmo.
4. Finalmente, crea un objeto de la clase TorresDeHanoi con un número de discos definido (por ejemplo, 4), y ejecuta el método para resolver el problema mostrando la animación gráfica paso a paso.