Taller de Programación Orientada a Objetos (POO) en Python

Objetivo: Reforzar los conceptos de la Programación Orientada a Objetos (POO) en Python a través de preguntas teóricas y ejercicios prácticos.

Parte 1: Preguntas Teóricas (Conceptos Clave)

1. **¿Qué es la Programación Orientada a Objetos (POO) y por qué es útil?**
   - Explica con tus propias palabras qué significa "programar con objetos".
   - Menciona una ventaja de usar POO en lugar de escribir todo el código en una secuencia.
2. **Diferencia entre Clase y Objeto:**
   - Imagina que vas a construir una casa. ¿Qué sería la "clase" y qué sería el "objeto" en este ejemplo?
3. **¿Qué son los Atributos y los Métodos en una clase?**
   - Si tienes una clase Perro, ¿qué atributos podría tener? ¿Qué métodos podría realizar?
4. **¿Cuál es la función del método especial __init__ en Python?**
   - ¿Cuándo se llama este método y para qué se utiliza?
5. **Explica el concepto de Herencia en POO.**
   - Da un ejemplo sencillo de cómo una clase puede "heredar" características de otra.
6. **¿Qué significa Encapsulamiento y cómo se logra en Python?**
   - ¿Por qué es importante "proteger" los datos de un objeto?
   - Menciona cómo se indican los atributos "privados" en Python.
7. **Define Polimorfismo con un ejemplo.**
   - Piensa en diferentes animales que hacen un "sonido". ¿Cómo se relaciona esto con el polimorfismo?
8. **¿Qué es la Abstracción en POO?**
   - ¿Por qué es útil ocultar los detalles complejos y mostrar solo lo esencial?
   - ¿Puedes crear un objeto directamente de una "clase abstracta"? ¿Por qué sí o por qué no?
9. **¿Qué son los Métodos Mágicos (Dunder Methods) en Python?**
   - Menciona un ejemplo de un método mágico y para qué se utiliza.

Parte 2: Ejercicios Prácticos (¡A Codificar!)

10. **Crea una Clase Libro:**
    - Define una clase llamada Libro con los siguientes atributos en su constructor (__init__): titulo, autor y anio_publicacion.
    - Agrega un método llamado mostrar_info que imprima el título, autor y año del

libro.
    - Crea un objeto de la clase Libro y llama a su método mostrar_info.

11. **Herencia: Clase Vehiculo y Coche:**
    - Crea una clase Vehiculo con un constructor que reciba marca y modelo.
    - Crea un método info_vehiculo que imprima la marca y el modelo.
    - Crea una clase Coche que herede de Vehiculo y que además tenga un atributo num_puertas.
    - Sobrescribe el método info_vehiculo en Coche para que también muestre el número de puertas.
    - Crea un objeto Coche y prueba su método info_vehiculo.

12. **Encapsulamiento: Clase Producto con Getter:**
    - Define una clase Producto con un atributo "privado" __precio.
    - Crea un método get_precio que permita obtener el valor del precio.
    - Intenta acceder al precio directamente (producto.__precio) y luego usando el getter. ¿Qué sucede?

13. **Polimorfismo: Función describir_animal:**
    - Crea dos clases: Pajaro y Pez.
    - Ambas clases deben tener un método llamado moverse. Pajaro.moverse debe imprimir "El pájaro vuela." y Pez.moverse debe imprimir "El pez nada.".
    - Crea una función describir_animal que reciba un objeto y llame a su método moverse.
    - Crea instancias de Pajaro y Pez y pásalas a la función describir_animal.

14. **Abstracción: Clase Abstracta Forma:**
    - Importa ABC y abstractmethod de abc.
    - Crea una clase abstracta Forma con un método abstracto calcular_area.
    - Crea una clase Circulo que herede de Forma y que implemente calcular_area (puedes usar 3.14159 para Pi).
    - Intenta crear un objeto de Forma. ¿Qué error obtienes?
    - Crea un objeto de Circulo y calcula su área.

15. **Método Mágico __str__: Clase Estudiante:**
    - Crea una clase Estudiante con atributos nombre y edad.
    - Implementa el método mágico __str__ para que, al imprimir un objeto

Estudiante, muestre un mensaje como: "Estudiante: [Nombre], Edad: [Edad]".

- Crea un objeto Estudiante y imprímelo directamente.