**1. Conceitos Básicos de POO**

* **~~O que é POO~~**~~: Entenda o paradigma de programação orientada a objetos.~~
* **~~Classes e Objetos~~**~~: A base de POO. Saiba como criar classes e instanciar objetos.~~
* **~~Atributos e Métodos~~**~~: Como declarar atributos e métodos (funções) dentro das classes.~~
* **Exercício Prático**: Crie uma classe Carro com atributos como marca, modelo, e ano, e métodos para ligar e desligar o carro.

**2. Encapsulamento**

* **~~Atributos Privados e Públicos~~**~~: Entenda a diferença e como controlá-los.~~
* **~~Métodos Getters e Setters~~**~~: Controle o acesso aos atributos de maneira segura.~~
* **Exercício Prático**: Expanda a classe Carro para incluir métodos get e set para acessar e modificar o modelo e o ano do carro de forma controlada.

**3. Herança**

* **~~O que é Herança~~**~~: Entenda como classes podem herdar de outras classes.~~
* **~~Superclasse e Subclasse~~**~~: Como criar classes que compartilham características comuns.~~
* **Exercício Prático**: Crie uma subclasse CarroEletrico que herde de Carro e adicione novos atributos como autonomia e métodos específicos.

**4. Polimorfismo**

* **Sobrescrita de Métodos**: Como modificar o comportamento de métodos em subclasses.
* **Exercício Prático**: Na subclasse CarroEletrico, sobrescreva o método de ligar o carro para incluir uma mensagem adicional sobre o status da bateria.

**5. Abstração**

* **O que é Abstração**: Como ocultar os detalhes complexos e mostrar apenas a funcionalidade essencial.
* **Classes Abstratas**: Introdução ao uso de classes abstratas e métodos abstratos em Python.
* **Exercício Prático**: Crie uma classe abstrata Veiculo com métodos abstratos ligar() e desligar(), e faça Carro e CarroEletrico herdarem dessa classe.

**6. Composição**

* **~~Composição vs. Herança~~**~~: Quando usar um em vez do outro.~~
* **~~Objetos dentro de Objetos~~**~~: Como utilizar composição para criar objetos mais complexos.~~
* **Exercício Prático**: Adicione uma classe Motor que seja um componente da classe Carro, com atributos como potencia e métodos como ligarMotor().

**Proposta de Projeto: Sistema de Gestão de Veículos**

A proposta de projeto será dividida em atividades para aplicar o conhecimento adquirido em cada etapa do mapa de estudos.

**Atividade 1: Criar uma classe base**

* Crie a classe Veiculo com os atributos básicos, como marca, modelo, e ano. Adicione métodos simples como ligar() e desligar().

**Atividade 2: Encapsulamento**

* Modifique a classe Veiculo para tornar seus atributos privados. Crie métodos getters e setters para acessá-los de forma controlada.

**Atividade 3: Herança**

* Crie subclasses Carro, Moto e Caminhao que herdem de Veiculo. Cada uma deve ter atributos e métodos específicos. Exemplo: Carro pode ter número de portas, enquanto Caminhao pode ter capacidade de carga.

**Atividade 4: Polimorfismo**

* Sobrescreva métodos da classe base Veiculo nas subclasses para comportamentos diferentes. Por exemplo, o método ligar() em Carro pode incluir uma verificação de combustível, enquanto Caminhao pode incluir uma verificação de carga.

**Atividade 5: Abstração**

* Transforme a classe Veiculo em uma classe abstrata e faça com que Carro, Moto e Caminhao implementem seus métodos abstratos.

**Atividade 6: Composição**

* Crie uma classe Motor e adicione-a como atributo de Veiculo. Cada tipo de veículo pode ter diferentes tipos de motor (por exemplo, motor elétrico ou a combustão).

**Atividade Final: Sistema Completo**

* Crie um pequeno sistema de cadastro de veículos, onde o usuário possa adicionar veículos diferentes, listar todos os veículos cadastrados e visualizar os detalhes de cada um, incluindo o motor.

Com essas atividades, você irá gradualmente construir um sistema mais completo enquanto aprende cada conceito de POO.