Exercícios de Orientação a Objetos em Python

Nível 1: Introdução (Fácil)

Exercício 1: Herança Simples de Atributos e Métodos

Crie uma classe base chamada **Animal** com um construtor que aceita o atributo nome. Ela deve ter um método chamado emitir_som() que apenas imprime "Som genérico". Em seguida, crie uma classe **Cachorro** que herda de Animal. No Cachorro, sobrescreva (override) o método emitir_som() para imprimir "Latido!". Crie instâncias de ambas as classes e chame o método.

Nível 2: Classes Abstratas e Herança (Intermediário)

Exercício 2: Classe Abstrata Básica

Crie uma classe abstrata chamada **FormaGeometrica** usando o módulo abc. Esta classe deve ter um método abstrato chamado **calcular_area()**. Crie uma classe concreta **Retangulo** que herda de FormaGeometrica. A classe Retangulo deve ter atributos para largura e altura e implementar o método calcular_area() para retornar a área correta.

Exercício 3: Herança com Construtores

Crie uma classe base **Funcionario** com um construtor que inicializa nome e salario. Crie uma classe filha **Gerente** que herda de Funcionario. O construtor de Gerente deve aceitar um atributo adicional departamento e usar super().__init__ para passar nome e salario para a classe base.

Nível 3: Abstração e Múltiplas Implementações (Avançando)

Exercício 4: Abstração com Propriedades

Crie uma classe abstrata **Veiculo**. Ela deve ter um método abstrato acelerar() e uma *propriedade abstrata* (usando @abstractmethod e @property) chamada **rodas** que as classes filhas devem implementar. Crie duas classes filhas: **Carro** (rodas = 4) e **Moto** (rodas = 2). Ambas devem implementar acelerar() com uma mensagem relevante.

Exercício 5: Herança em Múltiplos Níveis (Hierarquia)

Crie uma hierarquia de classes de três níveis:

1. **DispositivoEletronico** (Classe Base com ligar() e desligar()).
2. **Computador** (Herda de DispositivoEletronico, adiciona instalar_software()).
3. Notebook (Herda de Computador, adiciona verificar_bateria()).
   Crie uma instância de Notebook e chame um método de cada nível de herança.

Nível 4: Desafios com Abstração e Lista de Objetos (Mais Difícil)

Exercício 6: Classe Abstrata com Método Concreto

Crie uma classe abstrata **ContaBancaria**. Ela deve ter um método abstrato sacar(valor) e um método *concreto* chamado verificar_saldo() que usa um atributo saldo (inicializado no

construtor) para imprimir o saldo atual. Crie uma classe filha **ContaCorrente** que implementa o método sacar(valor) (simplesmente subtrai o valor do saldo). Crie uma lista de objetos ContaCorrente e itere sobre ela chamando sacar e verificar_saldo.

Exercício 7: Múltiplas Classes Abstratas (Mixins Conceituais)

Crie uma classe base abstrata **Percurso** com um método abstrato tempo_estimado(). Crie uma segunda classe base abstrata **Cobranca** com um método abstrato calcular_tarifa(). Crie uma classe concreta **Taxi** que *herda* (e implementa) ambas as classes abstratas, implementando os métodos de forma coerente.

Nível 5: O Desafio Final (Complexo)

Exercício 8: Abstração e Herança para Sistema de Cadastro

Crie um sistema de cadastro com classes abstratas e herança:

1. Classe Abstrata **Pessoa** com atributos id e nome e um método abstrato detalhes_de_cadastro().
2. Classe Concreta **Cliente** que herda de Pessoa e adiciona o atributo data_cadastro. Implemente detalhes_de_cadastro().
3. Classe Concreta **Fornecedor** que herda de Pessoa e adiciona o atributo cnpj. Implemente detalhes_de_cadastro().
4. Crie uma função que aceita uma lista de objetos que são **Pessoa** (ou seja, instâncias de Cliente ou Fornecedor) e itere sobre ela, chamando o método detalhes_de_cadastro() para cada item.

Modo de envio

-> Criar pasta no github em qualquer repositório chamada (modulo 02) e separadamente criar arquivos para cada questão.