Introdução à Programação Orientada a Objetos com Python

Capítulo 1: Introdução à POO

O que é POO?

A programação orientada a objetos (POO) é um paradigma de programação que utiliza "objetos" para modelar o mundo real dentro do código. Objetos são entidades que possuem características (atributos) e ações (métodos).

Benefícios da POO

- **Modularidade**: Divide o código em partes menores e organizadas.
- **Reutilização**: Classes podem ser usadas para criar vários objetos semelhantes.
- **Manutenção**: O código é mais fácil de entender, modificar e depurar.

Capítulo 2: Os Quatro Pilares da POO

1. Abstração

- **O que é**: A abstração focaliza nas características essenciais de um objeto, ignorando detalhes complexos internos.
- **Exemplo**: Ao modelar um carro, a abstração nos permite pensar nele como um objeto que possui cor, marca e pode acelerar/frear, sem se preocupar com detalhes do motor e transmissão.
- **Quando usar**: Sempre que quiser simplificar a representação de algo complexo em seu código.

2. Encapsulamento

- **O que é**: Reúne dados (atributos) e métodos (comportamento) relacionados dentro de uma classe, controlando o acesso a eles.
- **Exemplo**: Numa classe `ContaBancaria`, o saldo é um atributo encapsulado, podendo ser acessado/modificado apenas pelos métodos da própria classe, como `depositar` e `sacar`.
- **Quando usar**: Para proteger a integridade dos dados e facilitar a manutenção do código.

3. Herança

- **O que é**: Permite criar novas classes (subclasses) que herdam características de classes existentes (superclasses), adicionando ou modificando funcionalidades.
- **Exemplo**: Podemos ter uma classe `Animal` e subclasses `Cachorro` e `Gato` que herdam atributos comuns (nome, idade) e possuem métodos específicos como `latir()` e `miar()`.

- **Quando usar**: Quando há hierarquias ou categorias no domínio do problema que você está modelando.

4. Polimorfismo

- **O que é**: A capacidade de objetos de diferentes classes responderem ao mesmo método de maneiras distintas.
- **Exemplo**: Tanto um `Cachorro` quanto um `Gato` podem responder ao método `emitir_som()`, mas o som produzido será diferente para cada um.
- **Quando usar**: Para criar código flexível que pode lidar com diferentes tipos de objetos de forma homogênea.

Capítulo 3: Quando Utilizar POO?

A POO é especialmente útil quando:

- O problema a ser resolvido envolve entidades do mundo real, como no caso do cachorro ou da conta bancária.
- O projeto é complexo e precisa ser modularizado. A POO permite organizar o código em classes e objetos, facilitando a manutenção.
- Há necessidade de reutilização de código. Classes podem ser herdadas e reutilizadas em diferentes partes do programa.
- Flexibilidade é desejada. Polimorfismo permite lidar com diferentes tipos de objetos de forma homogênea.

Quando não usar POO

- Para programas simples. Em programas pequenos, a POO pode adicionar complexidade desnecessária.
- Para scripts de processamento de dados. A lógica procedural pode ser mais adequada nesse contexto.
- Quando a performance é crítica. A POO pode ter um impacto no desempenho em alguns casos.

Capítulo 4: Exemplos Práticos

Modelando um Cachorro

```
class Cachorro:
    def __init__(self, nome, raça, idade):
        self.nome = nome
        self.raça é raça
        sua idade é idade

    def latir(self):
        print("Woof!")

    def brincar(self):
        print(f"{self.nome} está brincando!")
```

Modelando uma Conta Bancária

```
class ContaBancaria:
    def __init__(self, titular, saldo=0):
        self.__titular é titular
        self.__saldo é saldo

    def depositar(self, valor):
        self.__saldo += valor
        print(f"Depósito de R${valor:.2f} realizado. Novo saldo: R${

    def sacar(self, valor):_
        se sua.__saldo >= valor:
            sua.__saldo -= valor
            print(f"Saque de R${valor:.2f} realizado. Novo saldo: R$
        caso contrário, ele diz "Saldo insuficiente!"
```

Capítulo 5: Exercícios

1. Crie uma classe `Pessoa` com atributos `nome` e `idade`, e um método `apresentar()`.
2. Crie uma classe `Retangulo` com métodos para calcular área e perímetro.
3. Implemente a herança com classes `Funcionário`, `Gerente` e `Programador`.

4. Use herança e um método abstrato `area()` em classes `Figura`, `Circulo` e `Quadrado`.
5. Crie uma classe `Televisao` com métodos para ligar/desligar, trocar canal e ajustar volume. 6. Crie uma classe `Aluno` com métodos para calcular a média e verificar aprovação.
6. Implemente uma classe `Loja` com métodos para gerenciar produtos.
7. Crie um sistema de biblioteca com classes `Livro`, `Usuario` e `Emprestimo`.
8. Crie um jogo de adivinhar o número usando classes.
9. Implemente um sistema de agenda com classes `Contato` e `Compromisso`.

Conclusão

A POO é uma ferramenta poderosa para modelar problemas complexos de forma organizada e eficiente. Utilize os pilares da POO e continue praticando para dominar esta técnica!

Este E-book foi escrito com prompt na IA Generativa Gemini Google AI Studio.