Resumo Completo sobre Programação Orientada a Objetos (POO) em Python.

Conceitos Fundamentais da POO:

- **Paradigma de Programação:** A POO é um paradigma que organiza o código em torno de objetos, que combinam dados (atributos) e comportamentos (métodos).
- **Classe:**
  - É um modelo ou "molde" para criar objetos. Define os atributos e métodos que os objetos da classe terão.
  - Pense em uma classe como um manual de instruções para construir objetos.
- **Objeto:**
  - É uma instância da classe, ou seja, um elemento individual criado a partir do modelo da classe.
  - Cada objeto tem seus próprios valores para os atributos e pode executar os métodos definidos na classe.
- **Atributos:**
  - São variáveis que armazenam dados sobre o objeto.
  - Representam as características do objeto.
- **Métodos:**
  - São funções definidas dentro de uma classe que realizam ações nos objetos.
  - Representam os comportamentos do objeto.
  - `self` é o primeiro argumento de um método e representa o objeto que está chamando o método.
- **Encapsulamento:**
  - É o princípio de agrupar dados (atributos) e métodos relacionados em uma unidade (classe).
  - Visa proteger os dados de acesso externo e controlar como eles são modificados.
  - "Setters" são metodos usados para modificar os atributos de um objeto.
- **Abstração:**
  - Focar nos aspectos relevantes de um objeto, ignorando detalhes complexos.
- **Herança:**
  - Criar novas classes (subclasses) a partir de classes existentes (superclasses), herdando seus atributos e métodos.
- **Polimorfismo:**
  - Permitir que objetos de diferentes classes sejam tratados de forma uniforme através de uma interface comum.

Método Construtor `__init__`:

- É um método especial que é chamado automaticamente quando um novo objeto é criado.
- É usado para inicializar os atributos do objeto.
- `self` é usado dentro do `__init__` para referenciar o objeto que está sendo criado.

Exemplos de Código:

1. Criação de uma Classe Básica:

Python
```
class Carro:
    cor = "Vermelho"
    modelo = "Sedan"
    ano = 2022
```

2. Criação de Objetos e Acesso aos Atributos:

Python
```
carro1 = Carro()
carro2 = Carro()

print(carro1.cor)    # Saída: Vermelho
print(carro2.modelo) # Saída: Sedan
print(carro1.ano)    # Saída: 2022
```

3. Modificação de Atributos:

Python
```
carro1.cor = "Azul"
carro2.ano = 2020

print(carro1.cor)  # Saída: Azul
print(carro2.ano)  # Saída: 2020
```

4. Método Construtor `__init__`:

Python
```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

joao = Pessoa("João", 30)
print(joao.nome, joao.idade)
```

5. Métodos em Classes:

Python
```
class Pessoa:
    def __init__(self, nome):
        self.nome = nome

    def apresentar(self):
        print(f"Olá, meu nome é {self.nome}!")
```

```
maria = Pessoa("Maria")
maria.apresentar()
```

6. Exemplo com `exibir_produto()`:

Python
```
class Produto:
    def __init__(self, nome, preco):
        self.nome = nome
        self.preco = preco

    def exibir_produto(self):
        print(f"Produto: {self.nome}, Preço: R${self.preco:.2f}")

produto1 = Produto("Notebook", 2500.00)
produto1.exibir_produto()
```

Por que usar POO?

- Reutilização de código.
- Organização do código.
- Modelagem do mundo real.
- Facilidade de manutenção.
- Abstração de codigo complexo.

Dicas:

- Pratique a criação de classes e objetos.
- Experimente criar seus próprios projetos usando POO.
- Explore os conceitos de herança e polimorfismo.