FUNDAMENTOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS (POO)

IFBA – Campus Ilhéus

Curso: Back End com Python

Objetivo da Aula

- Introduzir os conceitos básicos de POO para alunos iniciantes e mostrar sua importância na organização de código.

O que é Programação Orientada a Objetos?

- É um paradigma de programação que organiza o código usando objetos, que são instâncias de classes.
- Os objetos representam coisas do mundo real ou conceitos abstratos.
- Esses objetos possuem características (atributos) e comportamentos (métodos).

Analogia do mundo real

- Pense em um **carro**:
  - Ele tem **atributos** (cor, modelo, ano).
  - Ele pode realizar **ações** (andar, frear, buzinar).
- Em POO, você pode criar um objeto “carro” que reúne essas características e comportamentos.

Comparação entre programação procedural e orientada a objetos

Característica	Programação Procedural	Programação Orientada a Objetos (POO)
Definição	Baseia-se em **funções** com e sem retorno. O foco está em **ações**.	Baseia-se em **objetos** que contêm dados (atributos) e comportamentos (métodos).
Organização	Código estruturado em funções, com variáveis globais ou locais.	Código estruturado em **classes** e **objetos**, promovendo encapsulamento.
Reutilização de código	Menor reutilização. Funções são reaproveitadas, mas difícil modularizar bem.	Alta reutilização. Classes podem ser reutilizadas, estendidas (herança) e compostas.
Abstração	Menor nível de abstração. O programador lida com detalhes da implementação.	Maior nível de abstração. Objetos representam entidades do mundo real.
Manutenção	Torna-se difícil com o crescimento do projeto. Alto acoplamento entre funções e dados.	Mais fácil de manter. Separação entre dados e comportamentos. Baixo acoplamento e alta coesão.
Exemplo típico	Scripts simples, pequenos programas, algoritmos.	Sistemas grandes, jogos, aplicativos com GUI, programas orientados a entidades.

Conceitos-chave da POO

- **Classe:** É o **molde** ou **projeto** para criar objetos. Define atributos e métodos que os objetos daquela classe terão.
- **Objeto:** É uma **instância** da classe — um item criado a partir do molde.
- **Atributo:** É uma **característica** ou dado que o objeto possui.
- **Método:** É uma **função** definida dentro da classe que representa um comportamento ou ação do objeto.

Sintaxe para criação de classe em Python

```
class NomeDaClasse:
    def __init__(self, parametros):
        # inicializa os atributos do objeto
        self.atributo = parametros

    def metodo(self):
        # define um método da classe
        print("Este é um método da classe")
```

Explicando os componentes

Parte	Descrição
class NomeDaClasse:	Define o início da declaração de uma classe. O nome geralmente começa com letra maiúscula.
def __init__(self, ...)	Método especial chamado **construtor**, executado quando um novo objeto é criado.
self	Referência ao próprio objeto. É obrigatório como primeiro parâmetro dos métodos da classe.
Atributos	Variáveis associadas ao objeto, geralmente inicializadas no __init__.
Métodos	Funções definidas dentro da classe.

Exemplo 1 – Classe Pessoa

```
# Definindo a classe Pessoa
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome      # Atributo nome
        self.idade = idade    # Atributo idade

    def apresentar(self):     # Método que mostra dados da pessoa
        print(f"Olá, meu nome é {self.nome} e tenho {self.idade} anos.")
```

Continuação - Exemplo 1 – Classe Pessoa

```
# Criando objetos (instâncias) da classe Pessoa
pessoa1 = Pessoa("Ana", 25)
pessoa2 = Pessoa("João", 30)

# Usando o método apresentar
pessoa1.apresentar()  # Olá, meu nome é Ana e tenho 25 anos.
pessoa2.apresentar()  # Olá, meu nome é João e tenho 30 anos.
```

Exemplo 2 – Classe Retangulo

```
# Definindo a classe Retangulo
class Retangulo:
    def __init__(self, largura, altura):
        self.largura = largura
        self.altura = altura

    def area(self):
        return self.largura * self.altura

    def perimetro(self):
        return 2 * (self.largura + self.altura)
```

Continuação - Exemplo 2 – Classe Retangulo

```
# Criando um retângulo
ret = Retangulo(5, 3)

print("Área:", ret.area())          # Saída: Área: 15
print("Perímetro:", ret.perimetro())  # Saída: Perímetro: 16
```

Resumo

Termo	Definição
Classe	Molde ou modelo para criar objetos
Objeto	Instância da classe, um item concreto
Atributo	Característica/estado do objeto
Método	Função/comportamento do objeto

Exercícios

1. Crie uma classe Livro com os atributos:
   - titulo (str)
   - autor (str)
   - ano (int)
   - E um método descricao() que retorna uma string assim:
   - "Título: NOME_DO_LIVRO, Autor: AUTOR, Ano: ANO“
   - Teste: Crie 2 objetos da classe Livro e imprima suas descrições.

Exercícios

2. Crie uma classe Carro com:
   - Atributos:
     - modelo, ano, velocidade (inicialmente 0)
   - Métodos:
     - acelerar(): aumenta a velocidade em 10
     - frear(): diminui a velocidade em 10 (mínimo 0)
     - mostrar_velocidade(): imprime a velocidade atual
   - Teste: Crie um carro, acelere 2x, freie 1x, e mostre a velocidade.

Exercícios

3. Crie uma classe Aluno com os atributos:
   - Nome
   - Nota1
   - Nota2
   - E um método media(): retorna a média aritmética das duas notas
   - Teste: Crie um aluno, calcule e imprima sua média.

Exercícios

4. Crie uma classe ContaBancaria com:
   - Atributos: titular, saldo (inicialmente 0)
   - Métodos:
     - depositar(valor)
     - sacar(valor)
     - mostrar_saldo()
   - Regra: Não permitir saque se o valor for maior que o saldo.
   - Teste: Crie uma conta, faça depósitos e saques, e mostre o saldo após cada operação.

Exercícios

5. Crie uma classe ConversorTempo com:
   - Atributo: total_segundos
   - Método: converter() que usa divmod() para retornar:
     - Quantas horas e quantos segundos restam
   - Teste: Peça ao usuário os segundos e mostre o resultado assim:

     Resultado: X horas e Y segundos