Python

Programação Orientada a Objetos

Objetivos

- Paradigmas;
- Conceitos de POO;
- Pilares;
- Classes;
- Atributos;
- Métodos;
- Objetos;
- Exemplo prático;
- Exercícios;
- Referências.

Paradigmas

- Os paradigmas são modelagens de escrita de código que podem ser aplicados a várias linguagens;
- Um tipo de estruturação ao qual a linguagem deverá respeitar;
- De qual forma (paradigma) resolver um problema?

Paradigmas

- Exemplos de paradigmas:
  - Paradigma imperativo;
  - Paradigma declarativo;
  - Paradigma funcional;
  - Paradigma lógico;
  - Paradigma orientado a objetos;
  - Paradigma orientado a eventos;
  - Paradigma orientado a aspectos.

Conceitos de POO

- É um paradigma de programação que se baseia na criação e manipulação de objetos. Os objetos representam entidades do mundo real que possuem características (atributos) e comportamentos (métodos). A POO organiza o código em estruturas (classes) mais modulares e reutilizáveis, facilitando a manutenção e o desenvolvimento de software.

Pilares

Classes

- Uma classe é um modelo ou uma definição para criar objetos. Ela encapsula atributos e métodos que definem o comportamento dos objetos. Por exemplo, uma classe Pessoa pode ter atributos como nome e idade, e métodos como andar() e falar()

Atributos

- Um atributo é uma característica de um objeto. Representa dados que descrevem o estado do objeto

Métodos

- Um método é uma função definida dentro de uma classe que representa o comportamento do objeto. Métodos permitem que os objetos realizem ações e interajam entre si.

Objetos

- Um objeto é uma instância de uma classe. É uma entidade real que possui atributos específicos e pode executar métodos. Por exemplo, se Pessoa é uma classe, então uma instância dessa classe poderia ser um objeto chamado pessoa1 com atributos como nome “João" e idade 28.

Exemplo prático - arquivo pessoa

```
class Pessoa:
    pass

    def cadastrarPessoa(self):
        self.nome = input('Digite seu nome: ')
        self.idade = int(input('Digite sua idade: '))
        self.peso = float(input('Digite seu peso:'))

    def exibirPessoa(self):
        print(self.nome, self.idade, self.peso)
```

Exemplo prático – arquivo main

- `from pessoa import Pessoa`

- `p1 = Pessoa()`
- `p1.nome = 'Ana'`
- `p1.idade = 33`
- `p1.peso = 74.7`

- `print(p1.nome, p1.idade,p1.peso)`

- `p2 = Pessoa()`
- `p2.cadastrarPessoa()`
- `p2.exibirPessoa()`

Exercícios

- Criar uma aplicação utilizando os conceitos de POO para uma calculadora com as operações básicas da matemática

Construtor

- Em Python, um construtor é um método especial dentro de uma classe que é chamado automaticamente quando você cria uma nova instância dessa classe. O construtor é usado para inicializar os atributos da instância. O método especial utilizado como construtor em Python é __init__().

Construtores

```
class Pessoa:

    def __init__(self, nome, idade,peso):
        self.nome = nome
        self.idade = idade
        self.peso = peso

    def cadastrarPessoa(self):
        self.nome = input('Digite seu nome: ')
        self.idade = int(input('Digite sua idade: '))
        self.peso = float(input('Digite seu peso:'))

    def exibirPessoa(self):
        print(self.nome, self.idade, self.peso)
```

Construtores

- `from pessoa import Pessoa`

- `p3 = Pessoa('Pedro', 12,55)`
- `p3.exibirPessoa()`