Orientação a Objetos com Python

Classes e Objetos

WOMAKERS CODE

Programação Orientada a Objetos

- Programação orientada a objetos: é um **paradigma** de programação que modela os dados e comportamentos como se fossem objetos do mundo real.

```
muda_canal_para_cima(televisao, canal)
```

- Programação orientada a objetos: é um **paradigma** de programação que modela os dados e comportamentos como se fossem objetos do mundo real.
- **Objetos** em Python são a representação de um objeto do mundo real.
- **Objetos** possuem em geral dois componentes:

  1. Propriedades
  Ex: ligado/desligado, canal, canal máximo/mínimo, volume, volume máximo/mínimo
  2. Comportamentos
  Ex: mudar canal/volume para cima/baixo, ligar/desligar

```
televisao.mudar_canal_para_cima()
```

Programação Orientada a Objetos

- Na programação orientada a objetos (POO), objetos não são só variáveis (pedaços da memória que guardam valores).
- Eles também são representações de algo no mundo real (sujeito/ator), e reúnem propriedades e comportamentos desse sujeito.
- Objetos são autocontidos e reutilizáveis:

```
televisao.ligar()
```

```
televisao.aumentar_volume()
```

- Essa forma de modelar o mundo real cria códigos claros que mostram quem é o ator (**objeto**) e qual é o comportamento que está sendo invocado (**método**).
- POO não substitui a programação tradicional – ela te dá mais ferramentas para escrever um código limpo, conciso e legível.
- POO traz novos conceitos para a linguagem: **classes**, **herança**, **encapsulamento**, **abstrações** e **polimorfismo**.

Classes

- Como criar meus próprios objetos em Python? Utilizando **classes**.
- Classes estruturas usadas para definir um novo tipo de dados (criado pela programadora).
- Classes não são objetos!
- Classes descrevem o que um objeto vai ser, mas elas não criam o objeto.

Classes são definidas com a palavra-chave **class**

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 3

    def ligar(self):
        self.ligada = True

    def desligar(self):
        self.ligada = False

tv_sala = Televisao()
tv_sala.ligar()
tv_sala.canal = 4
tv_sala.desligar()
```

Classes

- Para criar um objeto daquela classe, nós temos que definir uma **instância**.
- Uma classe pode ter múltiplas instâncias, cada uma delas vai ser um objeto diferente e auto-contido.
- Classes sempre possuem nomes no singular. Se queremos um conjunto de objetos do tipo da classe, podemos criar coleções de objetos com aquele tipo. Por ex: uma lista de objetos **`Televisao`**

Para instanciar um objeto, você adiciona parênteses ao nome da classe.

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 3

    def ligar(self):
        self.ligada = True

    def desligar(self):
        self.ligada = False

tv_sala = Televisao()
tv_sala.ligar()
tv_sala.canal = 4
tv_sala.desligar()
```

Classes

- Toda classe tem um método especial chamado **construtor**.
- Ele inicializa o novo objeto da classe com seus valores padrão.

O método especial **__init__** será chamado sempre que criarmos um novo objeto do tipo da classe

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 3

    def ligar(self):
        self.ligada = True

    def desligar(self):
        self.ligada = False

tv_sala = Televisao()
tv_sala.ligar()
tv_sala.canal = 4
tv_sala.desligar()
```

Classes

- Tudo que aprendemos com funções é também válido para métodos.
- A principal diferença é que um método é associado a uma classe e atua sobre um objeto.
- O primeiro parâmetro do método é chamado **self** e representa a instância sobre a qual o método atuará.

Métodos da classe Televisão

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 3

    def ligar(self):
        self.ligada = True

    def desligar(self):
        self.ligada = False

tv_sala = Televisao()
tv_sala.ligar()
tv_sala.canal = 4
tv_sala.desligar()
```

Classes

- O parâmetro especial **self** indica a instância da classe que está sendo considerada.
- Observe também que escrevemos **self**.canal para criar um atributo, e não uma simples variável local.
- Atributos definidos dentro de **self** podem ser acessados por qualquer método dentro da classe.

self representa o objeto em si

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 3

    def ligar(self):
        self.ligada = True

    def desligar(self):
        self.ligada = False

tv_sala = Televisao()
tv_sala.ligar()
tv_sala.canal = 4
tv_sala.desligar()
```

Classes

- O parâmetro especial **self** é a indica a instância da classe que está sendo considerada.
- Observe também que escrevemos **self**.canal para criar um atributo, e não uma simples variável local.
- Atributos definidos dentro de **self** podem ser acessados por qualquer método dentro da classe.

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 3

    def ligar(self):
        self.ligada = True

    def desligar(self):
        self.ligada = False

tv_sala = Televisao()
tv_sala.ligar()
tv_sala.canal = 4
tv_sala.desligar()
```

Não é necessário passar o parâmetro **self** ao chamar os métodos da classe.
O interpretador faz isso aumaticamente para nós. Entretanto, não se esqueça de declarar self como o primeiro parâmetro de seus métodos.