

TREINAMENTO BACKEND SEMANA 5

Programação Orientada a Objetos

criado por: Pedro Neves



Para estudarmos sobre POO,
precisamos saber primeiro o
que é paradigma de
programação.



PARADIGMAS DE PROGRAMAÇÃO

paradigma de programação é uma forma de se abordar a resolução dos problemas na programação

Programação Imperativa

Programação Reativa

Programação Concorrente

Programação Funcional

Programação Orientada a Eventos

Programação Baseada em Componentes

EXEMPLO:

"Faça um programa que some todos os elementos de uma lista"



Paradigma Procedural

O paradigma que estamos acostumados



```
# Somando todos os numeros de uma lista  
(PARADIGMA PROCEDURAL)  
lista_numeros = [1, 2, 3, 4, 5]  
soma_total = 0  
for numero in lista_numeros:  
    soma_total += numero  
print(soma_total)
```

Paradigma Funcional



```
# Somando todos os numeros de uma lista  
(PARADIGMA FUNCIONAL)
```

```
from functools import reduce
```

```
lista_numeros = [1, 2, 3, 4, 5]  
resultado = reduce(lambda x, y: x + y,  
lista_numeros)  
print(resultado)
```


Paradigma Orientado a Objetos



```
# Somando todos os numeros de uma lista  
(PARADIGMA ORIENTADO A OBJETOS)
```

```
class Somador:  
    def __init__(self, lista):  
        self.lista = lista  
  
    def soma(self):  
        return sum(self.lista)
```

```
s = Somador([1, 2, 3, 4, 5])  
resultado = s.soma()  
print(resultado)
```



ORIENTAÇÃO A OBJETOS

É um paradigma de desenvolvimento que tem por objetivo melhorar a produtividade dos(as) programadores(as) no desenvolvimento de sistemas, possibilitando a construção de sistemas robustos com menos linhas de código,

o que torna a POO tão forte?

Pilares da POO

ABSTRAÇÃO

separar as características essenciais de um objeto ou sistema daquelas que são irrelevantes ou secundárias para a sua utilização

POLIMORFISMO

é a reescrita de um método

ENCAPSULAMENTO

capacidade de esconder detalhes da implementação do objeto, expondo só o que deve ser acessado publicamente.

HERANÇA

um objeto ser idealizado baseado em outro objeto (heranca permite reaproveitar código já escrito)

+modular +intuitivo +flexível +seguro
ideal para microserviços

Para programar utilizando a orientação a objetos, é necessário criar classes(modelos) que, são **abstrações do mundo real no mundo virtual**.



```
class Tennis:
    def __init__(self, marca, cor, material):
        self.marca = marca
        self.cor = cor
        self.material = material
```

Para que se possa representar dados a partir das classes, é necessária a **criação de objetos**.

```
class Tennis:
    def __init__(self, marca, cor, material):
        self.marca = marca
        self.cor = cor
        self.material = material

meu_tenis = Tennis('Nike', 'preto', 'couro')
```



Instância da classe = objeto

a criação de um objeto, é necessária para que se possa ter a **representação de dados** e a **manipulação das informações** e **executar operações contidas na classe**.



```
class Tennis:
    def __init__(self, marca, cor, material):
        self.marca = marca
        self.cor = cor
        self.material = material

    def mostrar_tenis(self):
        print(f'este é um tenis da {self.marca}, da cor {self.cor}')
```

```
meu_tenis = Tennis('Nike', 'preto', 'couro')
meu_tenis.marca = 'Adidas'
meu_tenis.mostrar_tenis()
# OUTPUT -> este é um tenis da Adidas, da cor preto
```

<- Representação de dados

<- manipulação das informações

<- executar operações contidas na classe


"Glossário" da POO

classe -> modelo de uma abstração.

atributo -> características
("variáveis") da classe.

método -> são funções que
pertencem a uma classe.

objeto -> uma instância de uma
classe que permite manipulações de
dados.



```
class Tennis:
    def __init__(self, marca, cor, material):
        self.marca = marca
        self.cor = cor
        self.material = material
```

An orange arrow points from the definition of 'classe' to the 'class Tennis:' line. A yellow arrow points from the definition of 'atributo' to the initialization of 'self.marca', 'self.cor', and 'self.material'. A light blue arrow points from the definition of 'método' to the 'def mostrar_tenis(self):' line.

```
def mostrar_tenis(self):
    print(f'este é um tenis da {self.marca}')
```

```
meu_tenis = Tennis('Nike', 'preto', 'couro')
meu_tenis.marca = 'Adidas'
meu_tenis.mostrar_tenis()
# OUTPUT -> este é um tenis da Adidas, da cor preto, feito de couro
```

Só Praticando para
entender tudo isso ...

Slides em:

<https://beacons.ai/pdrtuche>

Terminei de assistir a vídeo
aula de programação

Vou praticar



Errei tudo