Programação Orientada a Objetos em Python

Instrutor: Tarik Ponciano

Senac Fecomércio Sesc

Links da Disciplina

1. Discord: https://discord.gg/wt5CVZZWJs

2. Drive: https://drive.google.com/drive/folders/1hOl0DaPeAor7gnhKBUIlZ5n8lLRDNvUY?usp=sharing

3. Github: https://github.com/TarikPonciano/Programador-de-Sistema-SENAC

Orientação a Objetos – O que é

A **P**rogramação **O**rientada a **O**bjetos (**POO**) é um paradigma de programação baseado no conceito de **Classes** e **Objetos**.

Classes podem conter dados e código:

- **Dados** na forma de campos (também chamamos de **atributos** ou propriedades); e
- **Código**, na forma de procedimentos (frequentemente conhecido como **métodos**).

Uma importante característica dos objetos é que seus próprios métodos podem acessar e frequentemente modificar seus campos de dados: objetos mantém uma referência para si mesmo, o atributo `self` no Python.

Na POO, os programas são projetados a partir de objetos que interagem uns com os outros.

Esse paradigma se concentra nos objetos que os desenvolvedores desejam manipular, ao invés da lógica necessária para manipulá-los.

Essa abordagem de programação é adequada para programas grandes, complexos e ativamente atualizados ou mantidos.

Classes, Objetos, Métodos e Atributos

Esses conceitos são os pilares da Programação Orientada a Objetos então é muito importante que você os **DOMINE**:

- As **Classes** são tipos de dados definidos pelo desenvolvedor que atuam como um modelo para objetos. ***Pra não esquecer mais: Classes são fôrmas de bolo e bolos são objetos*** 😉
- **Objetos** são instâncias de uma Classe. Objetos podem modelar entidades do mundo real (Carro, Pessoa, Usuário) ou entidades abstratas (Temperatura, Umidade, Medição, Configuração).
- **Métodos** são funções definidas dentro de uma classe que descreve os comportamentos de um objeto. Em Python, o primeiro parâmetro dos métodos é sempre uma referência ao próprio objeto.
- Os **Atributos** são definidos na Classe e representam o estado de um objeto. Os objetos terão dados armazenados nos campos de atributos. Também existe o conceito de atributos de classe, mas veremos isso mais pra frente.

Declarando classes

No paradigma orientado à objetos, uma classe é a representação de algo do mundo real. No Python, o uso de classes é algo constante no desenvolvimento de programas.
Sendo assim, para declarar uma classe no Python é bem simples, como podemos ver abaixo:

```
class Pessoa():
        # Atributos e métodos da classe
```

```
class PessoaFisica():
    # Atributos e métodos da classe
```

Como vimos acima, para declarar uma classe no Python, utilizamos a palavra reservada class seguido do nome desta classe.

No Python, todas as classes devem, por boas práticas, possuir nomes que comecem com letra maiúscula e, caso sejam compostos, a primeira letra de cada palavra deve ser maiúscula, o que chamamos de formato CamelCase:

Criando o Construtor da classe

Uma classe é representada por atributos e métodos. Os atributos de uma classe representam as características que esta classe possui, já os métodos representam o comportamento da classe.

Para declarar um atributo em uma classe no Python é bem simples, basta definir o nome do atributo no método especial chamado __init__, este método define o construtor da classe, ou seja, é onde definimos como uma nova pessoa será criada em nosso programa.

Para definir os atributos de uma classe em seu construtor, basta passá-los como parâmetro, como podemos ver abaixo:

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

Senac Fecomércio Sesc

Usando a classe para instanciar objetos

Como vimos anteriormente, as classes representam a estrutura de um elemento no mundo real, porém ela é apenas o modelo destes elementos.

Sempre que precisamos criar “algo” com base em uma classe, dizemos que estamos “instanciando objetos”. O ato de instanciar um objeto significa que estamos criando a representação de uma classe em nosso programa.

Para instanciar um objeto no Python com base em uma classe previamente declarada, basta indicar a classe que desejamos utilizar como base e, caso possua, informar os valores referentes aos seus atributos, como podemos ver abaixo:

Usando a classe para instanciar objetos

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

if __name__ == "__main__":
    pessoa1 = Pessoa("João", "M", "123456")
    print(pessoa1.nome)
```

Ao executar a linha pessoa1 = Pessoa("João", "M", "123456") estamos criando um objeto do tipo pessoa com nome “João”, sexo “M” e cpf “123456”.

Com isso, agora possuímos uma forma de criar diversas pessoas utilizando a mesma base, a classe Pessoa.

Usando a classe para instanciar objetos

Ao executar o código anterior e imprimir o nome dessa pessoa print(pessoa1.nome), teremos o seguinte retorno:

```python
class Pessoa:
    def __init__(self, nome, sexo, cpf):
        self.nome = nome
        self.sexo = sexo
        self.cpf = cpf

if __name__ == "__main__":
    pessoa1 = Pessoa("João", "M", "123456")
    print(pessoa1.nome)
```

```
João
```

Senac Fecomércio Sesc

Sua vez

Crie uma classe chamada Pokemon. Tente imaginar os atributos que um objeto dessa classe teria.

Faça um programa que instância um objeto da classe Pokemon e imprima os atributos desse objeto.

Bônus: Crie 2 objetos Pokemon e tente criar uma função de batalha que recebe os 2 objetos e determine quem sai ganhando.

Declarando métodos

Como vimos anteriormente, uma classe possui atributos (que definem suas características) e métodos (que definem seus comportamentos).

Imagine que possuímos um atributo ativo na classe Pessoa. Toda pessoa criada em nosso sistema é inicializado como ativo, porém, imagine que queremos alterar o valor deste atributo e, assim, “desativar” a pessoa em nosso sistema e, além disso, exibir uma mensagem de que a pessoa foi “desativada com sucesso”.

Para isso, precisamos definir um comportamento para essa pessoa, assim, agora, ela poderá ser “desativada”.

Declarando métodos

Sendo assim, precisamos definir um método chamado “desativar” para criar este comportamento na classe Pessoa, como podemos ver abaixo:

```
class Pessoa:
    def __init__(self, nome, sexo, cpf, ativo):
        self.nome = nome
        self.sexo = sexo
        self.cpf = cpf
        self.ativo = ativo

    def desativar(self):
        self.ativo = False
        print("A pessoa foi desativada com sucesso")

if __name__ == "__main__":
    pessoa1 = Pessoa("João", "M", "123456", True)
    pessoa1.desativar()
```

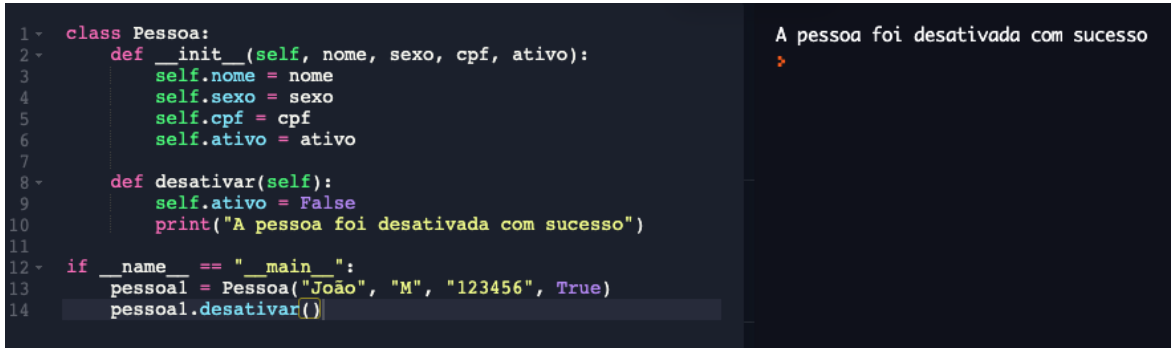
Declarando métodos

```
def desativar(self):
    self.ativo = False
    print("A pessoa foi desativada com sucesso")
```

Para criarmos este “comportamento” na classe Pessoa, utilizamos a palavra reservada def, que indica que estamos criando um método da classe, além do nome do método e seus atributos, caso possuam.

Declarando métodos

Depois disso, é só definir o comportamento que este método irá realizar. Neste caso, o método vai alterar o valor do atributo “ativo” para “False” e imprimir a mensagem “A pessoa foi desativada com sucesso”, como podemos ver abaixo:



Exercícios

1. Classe Triangulo: Crie uma classe que modele um triangulo:

– Atributos: LadoA, LadoB, LadoC
– Métodos: calcular Perímetro, getMaiorLado;

Crie um programa que uFlize esta classe. Ele deve pedir ao usuário que informe as medidas de um triangulo. Depois, deve criar um objeto com as medidas e imprimir sua área e maior lado.

2. Classe Funcionário: Implemente a classe Funcionário. Um funcionário tem um nome e um salário. Escreva um construtor com dois parâmetros (nome e salário) e o método aumentarSalario (porcentualDeAumento) que aumente o salário do funcionário em uma certa porcentagem. Exemplo de uso: harry = f u n c i o n á r i o (" H a r r y " , 2 5 0 0 0) harry.aumentarSalario(10) Faca um programa que teste o método da classe.

Exercícios

2. Classe Funcionário: Implemente a classe Funcionário. Um funcionário tem um nome e um salário. Escreva um construtor com dois parâmetros (nome e salário) e o método aumentarSalario (porcentualDeAumento) que aumente o salário do funcionário em uma certa porcentagem. Exemplo de uso:

```
harry = funcionario( " Harry" , 25000 )
harry.aumentarSalario(10)
```

Faca um programa que teste o método da classe.

Exercícios

3. Crie uma classe Livro que possui os atributos nome, qtdPaginas, autor e preço.

– Crie os métodos getPreco para obter o valor do preco e o método setPreco para setar um novo valor do preco.

Crie um codigo de teste

Obrigado!!

Senac Fecomércio Sesc

Referências

1. https://www.treinaweb.com.br/blog/orientacao-a-objetos-em-python
2. https://www.treinaweb.com.br/blog/os-pilares-da-orientacao-a-objetos/
3. https://pythonacademy.com.br/blog/introducao-a-programacao-orientada-a-objetos-no-python
4. https://pythonacademy.com.br/blog/introducao-a-programacao-orientada-a-objetos-no-python
5. http://200.17.137.109:8081/novobsi/Members/cleviton/disciplinas/introducao-a-programacao-2015-2/aulas/13%20Python%20-%20OO_parte1.pdf