**Samuel Miller Soares - DS17**

---

# POO - Programação Orientada a Objeto

Orientação em objeto organiza o código em objetos específicos, cada um deles podendo ter atributos e métodos

## Estruturação

```
class Cachorro:
        def __init__(self, variaveis) #Função
                #Parâmetros
```

__init__: utilizado para criar o objeto, o que é necessário ter para iniciar a classe

self: referência a classe presente, variável local

## Polimorfismo e Herança - Pesquisa

Orientação a objeto se consiste em um paradigma de programação no qual organiza o código em tornos de objetos, possui 4 elementos para sua estruturação:

- Classe
- Objeto
- Método
- Atributo

### *Conceito:*

*Herança:* Como já indicado no nome, a herança em orientação a objeto, tem como objetivo herdar tudo que outra classe tem, ou seja, compartilhar de forma hierárquica as características de uma classe a outra. Muito benéfico quando é necessário utilizar as mesmas funcionalidades várias vezes durante o código.

*Polimorfismo:* Já o polimorfismo se consiste na aplicação de um método para diferentes classes, ou seja, temos um método igual nas duas classes, porém com o resultado do método sendo diferente um do outro, por exemplo:

Animais fazem som, porém um cachorro late e o gato mia, ambos têm a mesma “funcionalidade” porém cada um com um resultado diferente, de forma simples podemos fazer o seguinte esquema para demonstrar essa situação.

---

animal = “Fazer som”
    gato = “Miar”
    cachorro = “Latir

O polimorfismo na POO compartilha o comportamento dessas duas classes (Gato e Cachorro) realizarem a mesma ação, porém com um final diferente, e

---

***Exemplo prático:***

```
class Veiculo:
    def __init__(self, marca, modelo):
        self.marca = marca
        self.modelo = modelo

    def descricao(self):
        desc = "É um veiculo!"
        return desc

class Carro(Veiculo):
    def __init__(self, marca, modelo, num_portas):
        super().__init__(marca, modelo)
        self.num_portas = num_portas

    def descricao(Veiculo):
        desc = "É um carro"
        return desc

class Moto(Veiculo):
    def __init__(self,marca, modelo, tipo_guidon):
        super().__init__(marca, modelo)
        self.tipo_guidon = tipo_guidon

    def descricao(Veiculo):
        desc = "É uma moto"
        return desc

veiculo01 = Veiculo("Scania", "P. 21")
carro01 = Carro("Fiat", "Argo", "4")
moto01 = Moto("Honda", "Elite 125", "Flat")

for i in (veiculo01, carro01, moto01):
    print(f"{i.marca} - {i.modelo} - {i.descricao()}")
```

No exemplo acima é possível perceber as principais características tanto de herança, quanto de polimorfismo. A princípio é criada a classe “Veiculo”, onde apresenta os atributos “marca” e “modelo”, nas classes “Carro” e “Moto” esses atributos são passados através da função “super()”, e abaixo é adicionado outro atributo referente a cada um desses objetos. O polimorfismo está presente no momento da função “Descrição”, onde as subclasses puxam o método do “Veiculo” e os sobrescrevem para um método específico.

**Samuel Miller Soares - DS17**

---

## Diferenças principais

A herança funciona tanto para atributos quanto para métodos, e é possível adicionar ambos em subclasses no código, tem como principal objetivo passar de uma classe para outra as características já definidas anteriormente. Já no polimorfismo, que funciona principalmente em atributos, ocorre uma sobrescrita/adaptação dos métodos anteriores, criando um método específico para a classe local.

## Vantagens

**Herança:**

- Permite a reutilização de código, passando as características das classes “pais” para as “filhas”. (Reutilização)
- É visível a presença de um código mais organizado e menos complexo com o uso desse método, menos variáveis são criadas, permitindo uma maior legibilidade e entendimento do código. (Organização)

**Polimorfismo:** É visível as mesmas vantagens da “Herança”, com algumas adições.

- Caso algum método está com problema ou com um erro específico, o uso do polimorfismo facilita a visualização desse método e garante na maioria das vezes que o problema será somente local. (Manutenção)
- É facilmente expansível, somente a criação de uma nossa classe, com um método puxando o de uma classe “pai” já permite a extensão do programa. (Extensível)

---

## Referências

***https://www.dio.me/articles/heranca-e-polimorfismo-em-python-aprenda-a-estruturar-suas-hierarquias-de-classes***

***https://www.devmedia.com.br/conceitos-e-exemplos-polimorfismo-programacao-orientada-a-objetos/18701***

***https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos?srsltid=AfmBOorYVpZ-i2U0urKrcXe4-5xBMFtl-3Dr_Pmc2dqc9KsLXahh03PD***

***https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP#:~:text=Object%2Doriented%20programming%20(OOP)%20is%20a%20computer%20programming%20model,has%20unique%20attributes%20and%20behavior*.**