**Conceito entre Polimorfismo e Herança**

Polimorfismo em Python

Primeiro conceito que a gente deve entender sobre quando a gente vai usar a orientação de objetos em Python é o Polimorfismo, ele é muito bem utilizado quando a gente cria um objeto para ser tratado em diferentes maneiras, dependendo do contexto em que é utilizado dessa classe.

Muitas vezes, o Polimorfismo ele é alcançado por uso de métodos em diferentes classes, ele promove a flexibilidade e a reutilização de código, isso, significa que ele pode ter comportamentos diferentes em classes diferentes, ou seja, ele permite o mesmo método que seja chamado para objetos em diferentes classes, produzindo resultados diferentes. Além do mais, ele também é útil quando se trabalho da **herança.**

1) Com base nisso, iremos analisar o *Polimorfismo*. Para isso iremos adicionar as classes vilão e atacar, observe abaixo:

Class Personagem

def \_\_init\_\_(self, nome, vida):

self.nome = nome

self.vida = vida

class Heroi(Personagem):

def \_\_init\_\_(self, nome, vida, habilidade):

super().\_\_init\_\_(nome, vida)

self.habilidade = habilidade

class Vilao(Personagem): # Adicionando a classe Vilão

def \_\_init\_\_(self, nome, vida, poder):

super().\_\_init\_\_(nome, vida)

self.poder = poder

def atacar(personagem):

print(f”{personagem.nome} está atacando!”)

heroi1 = Heroi(“Superman”, 100, “Voo”)

vilao1 = Vilao(“Lex Luthor”, 80, “Inteligência”)

atacar(heroi1) # Chamando a função atacar() com um herói

atacar(vilao1) # Chamando a função atacar() com um vilão

Nós Usamos o Poliformismo em ação,

Observe que a função atacar pode ser chamada com qualquer objeto que tenha um atributo nome como Herói ou vilão, mesmo eles sendo diferentes

Podemos tratar ele da mesma maneira nessa função. Esse é o Poliformismo que estávamos procurando.

Herança em Python

Segundo conceito é a Herança: Ele é um conceito fundamental na programação orientada a objetos (POO) e desempenha um papel crucial em Python. A herança permite que uma classe herde atributos e métodos de outra classe, facilitando a reutilização de código e a criação de hierarquias de classes.

# **Exemplo de herança em Python:**

Em Python, para criar uma classe que herda de outra classe, utilizamos a palavra-chave **CLASS** seguida pelo nome da nova classe e entre parênteses o nome da classe da qual queremos herdar. Por exemplo:

class Veículo:

def \_\_init\_\_(self, marca, modelo):

self.marca = marca

self.modelo = modelo

def acelerar(self):

print(“Acelerando o veículo...”)

Class carro(Veículo):

def \_\_init\_\_(self, marca, modelo, cor):

super().\_\_init\_\_(marca, modelo)

self.cor = cor

def ligar\_radio(self):

print(“Ligando o rádio do carro...”)

No exemplo acima, a classe Carro herda da classe Veiculo. Isso significa que a classe Carro terá acesso aos atributos e métodos da classe Veiculo, como marca e modelo, além de poder adicionar seus próprios atributos e métodos, como cor e ligar\_radio.