Nome: Júlia Alves Pereira
Turma: 2DSM-SL
Matéria: BackEnd
Professor: Dorival

Polimorfismo e Herança

Herança

Você já ouviu falar sobre herança? É como quando você herda coisas de seus pais, como olhos azuis ou cabelos cacheados. Em Python, é parecido.

Imagine que você tem uma classe chamada "Animal" que tem algumas características, como "comer" e "dormir". Agora, se você quiser fazer uma classe específica para um "Cachorro", você pode dizer que ela é uma espécie de "Animal" e, portanto, herda as características básicas de um "Animal". Isso é herança!

Ex:

```
class Personagem: #Definimos a classe Personagem
    def __init__(self, nome, vida): #método construtor __init__ e atribuímos os parâmetros nome e vida
        self.nome = nome
        self.vida = vida

class Heroi(Personagem):  # Herói herda de Personagem
    def __init__(self, nome, vida, habilidade):
        super().__init__(nome, vida)  # Chamando o construtor da classe mãe
        self.habilidade = habilidade

heroi1 = Heroi("Superman", 100, "Voo")  # Criando um herói
print(heroi1.nome)  # Saída: Superman
print(heroi1.vida)  # Saída: 100
print(heroi1.habilidade)  # Saída: Voo
```

Polimorfismo em Python

Polimorfismo é como uma palavra grande para algo bem simples. É quando diferentes coisas podem ser tratadas da mesma forma.

Por exemplo, se você tem uma função que espera um "Animal", você pode passar um "Cachorro" ou um "Gato" para ela. Porque, afinal, ambos são tipos de "Animal". Isso é polimorfismo em ação!

Ex:

```
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):  # Função para atacar, pode ser chamada por heróis ou vilões
    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
```

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.

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