**HERANÇA E POLIMORFISMO**

1. Conceito teórico de Herança e Polimorfismo

Herança:

A herança é muito utilizada quando utilizamos os mesmos atributos e métodos de uma classe em outra, assim, a classe “filha” ou “subclasse” herda os atributos e métodos da classe “mãe” ou “superclasse”.

Exemplo:

Temos a classe “mãe” “Animal”, e temos classes “filha” como “Cachorro” e “Peixe”.

A classe “Animal” tem os seguintes atributos: “tamanho”, “cor”.

Com isso, as classes filha “cachorro” e “peixe” também terão estes atributos, mas elas podem ter seus atributos específicos como:

A classe “Cachorro” tem como atributo específico: “quantidade\_patas”.

A classe “Peixe” tem como atributo específico: “tamanho\_nadadeira”.

Ao fim disso, as classes terãos seguintes atributos:

Animal: tamanho e cor.

Cachorro: tamanho, cor e quantidade\_patas.

Peixe: tamanho, cor e tamanho\_nadadeira.

Esse exemplo também se aplica a métodos, e não só aos atributos, fazendo com que a classe “filha” também tenha os métodos da classe “mãe”, mas ela pode ter os seus métodos específicos.

Polimorfismo:

O polimorfismo é utilizado quando precisamos realizar um tratamento em um objeto de maneiras distintas, dependendo do contexto em que ele está inserido. Um exemplo de quando usamos polimorfismo, é quando usamos métodos que têm o mesmo nome, mas em diferentes classes, e com jeitos diferentes de se utilizar em diferentes classes, com isso, gerando diferentes resultados. Ou seja, um método pode ser acessado de diferentes classes, e dependendo da classe ele será utilizado de um jeito ou de outro.

Exemplo:

Temos a classe “Meio\_transporte” que tem o método “locomover” e temos outras duas classes chamadas “Carro” e outra chamada “Avião”. Essas duas classes utilizam o método “locomover”, mas de modos diferentes. Quando o método “locomover” é chamado com um objeto da classe “carro”, o objeto dessa classe se locomove 2 posições. Já, quando o método “locomover” é chamado com um objeto da classe “avião”, o objeto dessa classe se locomove 40 posições. Com isso, vemos que através do polimorfismo, podemos utilizar o mesmo método em classes diferentes, e gerar resultados diferentes.

1. Exemplos com código:

Herança:

class Veiculo:

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

self.marca = marca

self.modelo = modelo

def exibir\_info(self):

print(f"Marca: {self.marca}, Modelo: {self.modelo}")

class Carro(Veiculo):

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

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

self.cavalos = cavalos

def exibir\_info(self):

super().exibir\_info()

print(f"Cavalos: {self.cavalos}")

class Moto(Veiculo):

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

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

self.cilindradas = cilindradas

def exibir\_info(self):

super().exibir\_info()

print(f"Cilindradas: {self.cilindradas}")

meu\_carro = Carro("Toyota", "Corolla", 200)

minha\_moto = Moto("Honda", "CBR", 600)

meu\_carro.exibir\_info()

minha\_moto.exibir\_info()

Classe mãe: Veículo.

Classes filhas que herdam atributos e métodos: Carro e Moto.

Polimorfismo:

class Animal:

def fazer\_som(self):

print("Animal faz som.")

class Cachorro(Animal):

def fazer\_som(self):

print("O cachorro late: Au Au!")

class Gato(Animal):

def fazer\_som(self):

print("O gato mia: Miau!")

def reproduzir\_som(animal):

animal.fazer\_som()

animal = Animal()

cachorro = Cachorro()

gato = Gato()

reproduzir\_som(animal)

reproduzir\_som(cachorro)

reproduzir\_som(gato)

Classe mãe: Animal.

Classes filhas que herdam métodos: Cachorro e Gato.

Todas têm um método chamado “fazer som”, mas cada classe tem seu comportamento.

1. Diferenças e relações entre herança e polimorfismo:

Esses dois conceitos são um pouco parecidos, pois os dois utilizam uma ou mais classe “filha” para herdar métodos e atributos da classe “mãe”. Na herança, a classe “filha” herda os atributos e métodos da classe “mãe”, mas ela não sobrescreve o método. Já com o polimorfismo, a classe “filha” herda e sobrescreve o apenas o método da classe “mãe”.

1. Vantagens:

Com a aplicação desses conceitos em um código, é possível fazer a reutilização de código de forma muito mais organizada, fácil e rápida. Além de que o código fica mais organizado, legível e tem uma manutenção muito mais fácil.

**REFERÊNCIAS**

MARCELA. Aprenda sobre Herança e Polimorfismo Em Python: Conceitos Essenciais para Programadores Iniciantes e Avançados. Disponível em: <<https://awari.com.br/aprenda-sobre-heranca-e-polimorfismo-em-python-conceitos-essenciais-para-programadores-iniciantes-e-avancados/>>.

Acesso em: 20 maio. 2025.

Herança e Polimorfismo | Alura Cursos Online. Disponível em: <<https://www.alura.com.br/apostila-python-orientacao-a-objetos/heranca-e-classes-abstratas?srsltid=AfmBOoqGmHx-huOvt7i4y-0TRzjVMBlfZQ1TMB1sVdueqc_hYdawIo1_>>.

Acesso em: 20 maio. 2025.

SILVA, S. Herança e Polimorfismo em Python: Aprenda a Estruturar suas Hierarquias de Classes. Disponível em: <<https://www.dio.me/articles/heranca-e-polimorfismo-em-python-aprenda-a-estruturar-suas-hierarquias-de-classes>>.

Acesso em: 20 maio. 2025.

‌