Herança e Polimorfismo

1. Herança

Na orientação a objetos, ocorre quando uma classe herda atributos e métodos de uma ou mais classes.Tendo a classe Pai(superclasse) e a classe Filha(subclasse). Podendo ser dividida em dois tipos:

Herança simples: classe herda atributos e métodos de apenas uma classe.

Herança múltipla: classe herda atributos e métodos de duas ou mais classes.

benefícios:

→ Ao herdar de uma classe existente, podemos aproveitar os atributos e métodos já implementados, evitando a necessidade de reescrever o mesmo código várias vezes

→ A herança permite que criemos classes especializadas a partir de classes base, adicionando funcionalidades específicas. Isso nos permite entender o comportamento das classes existentes sem modificar seu código original

1. Polimorfismo

Consiste em um mesmo método apresentar comportamentos diferentes, dependendo da classe em que seja chamado. Pode se apresentar em duas classificações:

Polimorfismo estático: ocorre em momento de compilação. O mesmo método é implementado várias vezes na mesma classe, com parâmetros diferentes. A escolha do método a ser chamado vai variar de acordo com o parâmetro passado.

Polimorfismo dinâmico: ocorre em momento de execução. O mesmo método é implementado várias vezes nas subclasses derivadas, com os mesmos parâmetros. A escolha do método depende do objeto que o chama(e, consequentemente, da classe que o implementa)

1. Exemplos

Herança:

class Animal():

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

self.\_\_nome = nome

self.\_\_cor = cor

def comer(self):

print(f"O {self.\_\_nome} está comendo")

No código acima definimos a classe pai que irá possuir todos os atributos e métodos comuns às classes filhas (Gato, Cachorro e Coelho). Nela, criamos apenas o construtor que irá receber o nome e a cor do animal, além do método comer que vai exibir a mensagem com o nome do animal que está comendo.

Polimorfismo:

class Super:

def hello(self):

print("Esta é a superclasse")

class Sub (Super):

def hello(self):

print("Esta é a subclasse")

class Subsub(Sub):

def helo(self):

print("Esta é a subsubclasse")



Aqui a subclasse herda atributos e métodos da superclasse, porém se ela for chamada ela irá mostrar o método da própria classe,ou seja, ele possui o método da superclasse e o próprio método fazendo uma sobreposição, o mesmo para o sub sub classe, que herda da subclasse que herda da superclasse.

1. Diferenças e semelhanças

| Diferenças | Semelhanças |
| --- | --- |
| A herança é utilizada para criar hierarquias de classes, enquanto o polimorfismo serve para que diferentes objetos sejam tratados de forma igual já que compartilham de uma interface em comum | Ambos são partes fundamentais para a orientação a objetos |
| O foco da herança é na reutilização do código, já o foco do polimorfismo é na flexibilidade e na extensibilidade do código | Ambos ajudam na eficiência do código, ajudam a lidar com a complexibilidade de um código deixando ele mais robusto e fácil de manter |

1. Vantagens

A utilização da herança promove a reutilização do código, permite a criação de estruturas e classes hierárquicas e oferece suporte ao polimorfismo, deixando o código mais flexível e fácil manutenção.

A utilização do polimorfismo permite usar uma única interface para diferentes tipos de dados, permitindo adicionar novas classes com suas implementações sem alterar o código existente

1. Referências Bibliográficas

* <https://www.treinaweb.com.br/blog/utilizando-heranca-no-python#google_vignette>
* <https://www.pythonprogressivo.net/2018/11/Polimorfismo-O-que-Como-Usar-Como-fazer.html#google_vignette>
* <https://www.datacamp.com/pt/tutorial/python-inheritance>
* <https://www.wscubetech.com/resources/python/polymorphism>