Herança

**Conceito:** Uma classe filha herda atributos e métodos da classe pai.

class Animal:

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

self.nome = nome

def emitir\_som(self):

print("O animal faz um som")

class Cachorro(Animal): # Herança

def emitir\_som(self):

print("O cachorro late")

dog = Cachorro("Rex")

dog.emitir\_som() # Saída: O cachorro late

Polimorfismo

**Conceito:** Permite que métodos com o mesmo nome tenham comportamentos diferentes em classes diferentes.

class Gato(Animal):

def emitir\_som(self):

print("O gato mia")

animais = [Cachorro("Rex"), Gato("Mimi")]

for animal in animais:

animal.emitir\_som()

# Saída:

# O cachorro late

# O gato mia

Encapsulamento

**Conceito:** Protege os dados da classe, controlando o acesso com atributos/métodos privados.

class ContaBancaria:

def \_\_init\_\_(self, saldo):

self.\_\_saldo = saldo # Atributo privado

def depositar(self, valor):

if valor > 0:

self.\_\_saldo += valor

def ver\_saldo(self):

return self.\_\_saldo

conta = ContaBancaria(1000)

conta.depositar(500)

print(conta.ver\_saldo()) # Saída: 1500

# print(conta.\_\_saldo) -> ERRO! Atributo privado

Abstração

**Conceito:** Foca apenas nos detalhes relevantes, escondendo a complexidade com classes base e métodos abstratos (com abc).

from abc import ABC, abstractmethod

class Forma(ABC):

@abstractmethod

def calcular\_area(self):

pass

class Quadrado(Forma):

def \_\_init\_\_(self, lado):

self.lado = lado

def calcular\_area(self):

return self.lado \*\* 2

forma = Quadrado(4)

print(forma.calcular\_area()) # Saída: 16

Explicação mais detalhada

Abstração é o princípio de esconder os detalhes complexos de implementação e expor apenas os comportamentos essenciais e relevantes para o uso de um objeto.

Você não precisa saber como algo funciona internamente, apenas como usá-lo.

Foca no “o que” uma coisa faz, e não no “como” ela faz.

Ajuda a reduzir a complexidade e isolar mudanças.

Quando você usa um controle remoto, você aperta o botão “Ligar”.

* Você não precisa saber como os sinais infravermelhos são enviados ou como o circuito é ativado.
* O que importa é que você pode usar aquele botão para ligar a TV.

from abc import ABC, abstractmethod

class Forma(ABC): # Classe abstrata

@abstractmethod

def calcular\_area(self):

pass # Não implementa nada, só define a interface

class Retangulo(Forma):

def \_\_init\_\_(self, largura, altura):

self.largura = largura

self.altura = altura

def calcular\_area(self): # Implementação concreta

return self.largura \* self.altura

A classe Forma define o que deve ser feito (calcular\_area), mas não como.

Retangulo é obrigado a implementar esse comportamento.

Quem usa Forma sabe que pode chamar calcular\_area() sem se preocupar com os detalhes de cálculo.