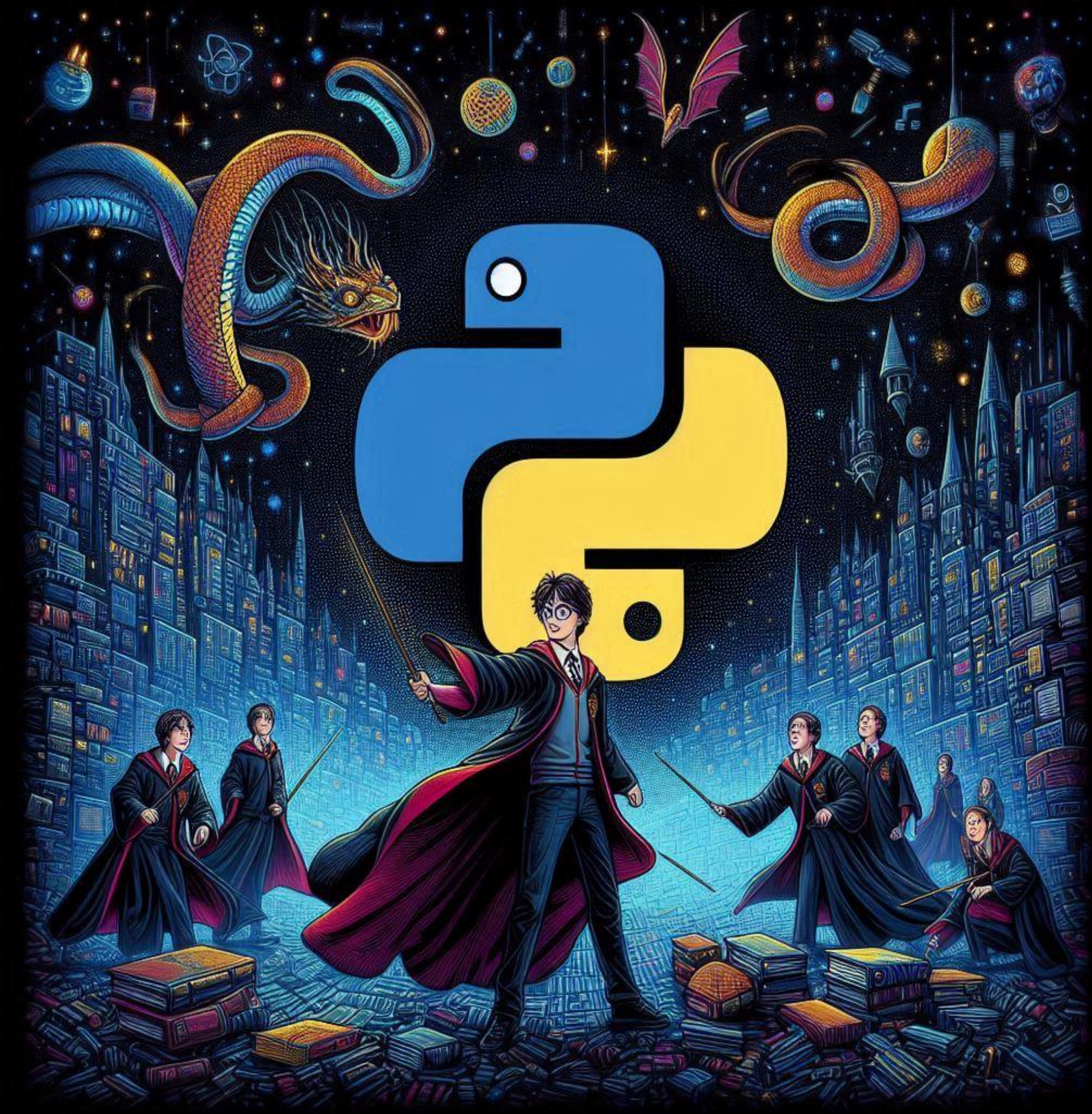
PYTHON

O CÁLICE MÁGICO DA CODIFICAÇÃO



Sthefanie Carvalho

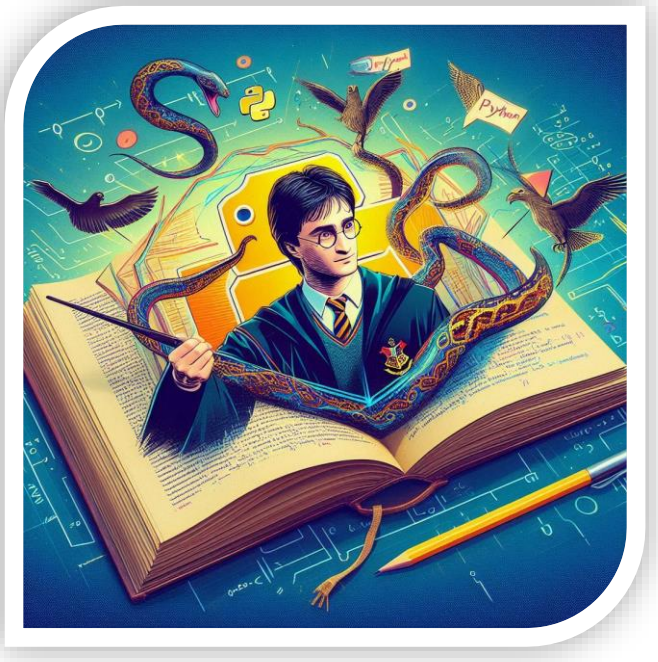
INTRODUÇÃO AO PYTHON

Bem-vindo ao fascinante mundo do Python!

Python é uma linguagem de programação de alto nível, conhecida por sua sintaxe clara e legível. Foi criada por Guido van Rossum e lançada em 1991. Essa linguagem é amplamente utilizada em diversas áreas, desde desenvolvimento web até ciência de dados e inteligência artificial.

Prepare-se para explorar os principais conceitos de Python e ver como essa linguagem pode transformar suas ideias em realidade com eficiência e elegância.

Vamos começar essa jornada emocionante!



CAPÍTULO

FUNDAMENTOS DA LINGUAGEM

SINTAXE BÁSICA

A sintaxe do Python é simples e direta, facilitando a leitura e a escrita do código.

```
print("Hello, World!")
```

Variáveis e Tipos de Dados

Python é dinamicamente tipado, o que significa que você não precisa declarar o tipo da variável explicitamente. No entanto, é importante conhecer os tipos de dados disponíveis em Python..

- **Inteiros (int):** Números inteiros positivos ou negativos, sem casa decimal.

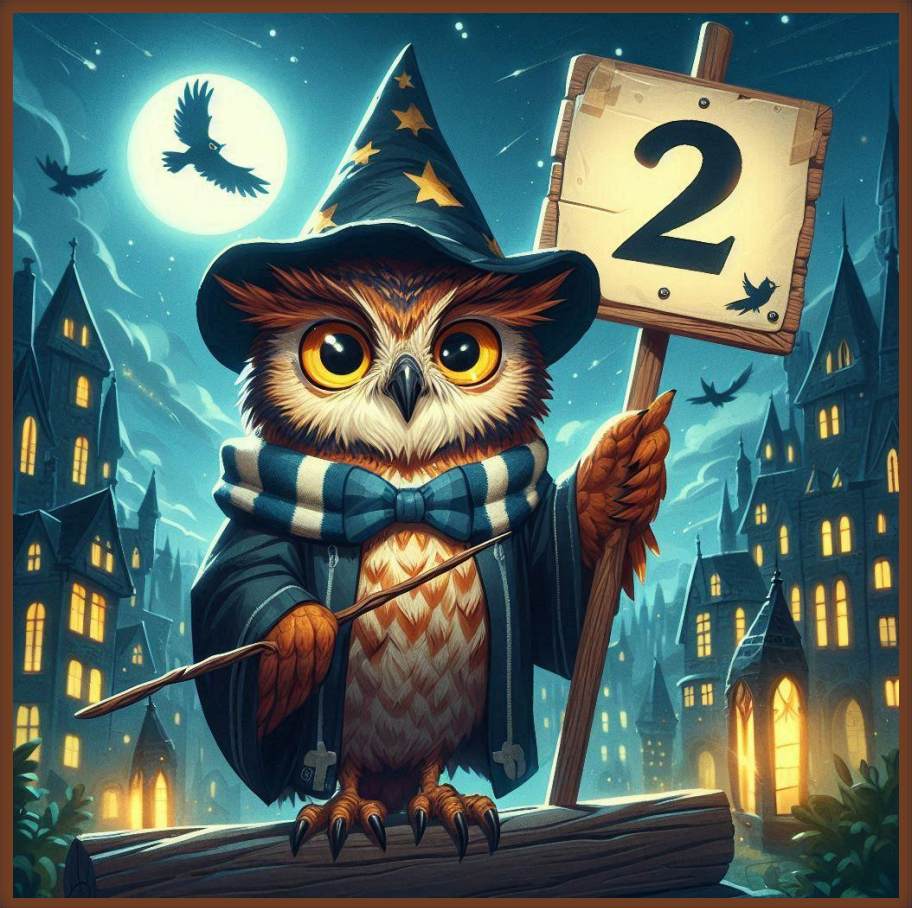
- **Pontos Flutuantes (float):** Números reais, ou seja, com casas decimais.

SINTAXE BÁSICA

- **Strings (str):** Cadeias de caracteres, usadas para representar texto.
- **Booleanos (bool):** Valores lógicos, podendo ser True ou False.
- **Listas (list):** Coleções ordenadas de elementos, que podem ser de diferentes tipos.
- **Tuplas (tuple):** Coleções ordenadas e imutáveis de elementos.
- **Conjuntos (set):** Coleções desordenadas de elementos únicos.
- **Dicionários (dict):** Coleções de pares chave-valor.

```python
idade = 30
salario = 4500.75
nome = "João"
is_empregado = True
frutas = ["maçã", "banana", "laranja"]
coordenadas = (10, 20)
numeros_unicos = {1, 2, 3, 4, 5}
pessoa = {"nome": "João", "idade": 30, "altura": 1.70}
```

CAPÍTULO



FUNÇÕES E MÓDULOS

DEFININDO FUNÇÕES

Funções permitem que você reutilize código, organizando-o de forma modular. Em Python, você define uma função usando a palavra-chave def, seguida pelo nome da função e parênteses.

Dentro dos parênteses, você pode incluir parâmetros, que são variáveis que a função pode receber como entrada.

```python
def saudacao(nome):
    return f"Olá, {nome}!"

print(saudacao("João"))
```

Parâmetros e Argumentos

Parâmetros são variáveis definidas na declaração da função, enquanto argumentos são os valores reais passados para a função quando ela é chamada.

```python
def soma(a, b):
    return a + b

print(soma(5, 3))  # Output: 8
```

DEFININDO FUNÇÕES

Argumentos Padrão

Você pode definir valores padrão para os parâmetros de uma função. Se um argumento não for fornecido, o valor padrão será usado.

```python
def saudacao(nome, mensagem="Olá"):
    return f"{mensagem}, {nome}!"

print(saudacao("João"))          # Output: Olá, João!
print(saudacao("João", "Oi"))    # Output: Oi, João!
```

Argumentos Nomeados

Ao chamar uma função, você pode especificar os argumentos usando o nome do parâmetro, tornando o código mais legível.

```python
def apresentar(nome, idade):
    return f"Nome: {nome}, Idade: {idade}"

print(apresentar(nome="João", idade=30))

# Output: Nome: João, Idade: 30
```

DEFININDO FUNÇÕES

Funções Lambda

Funções lambda são funções anônimas, definidas usando a palavra-chave lambda. Elas são úteis para funções curtas e rápidas.

```python
soma = lambda a, b: a + b
print(soma(5, 3))  # Output: 8
```

Escopo de Variáveis

O escopo de uma variável determina onde ela pode ser acessada. Em Python, existem dois tipos principais de escopo:

- Escopo Local: Variáveis definidas dentro de uma função só podem ser acessadas dentro dessa função.
- Escopo Global: Variáveis definidas fora de todas as funções podem ser acessadas em qualquer lugar do código.

```python
x = 10  # Escopo global

def funcao():
    x = 5  # Escopo local
    print(f"Escopo local: {x}")

funcao()  # Output: Escopo local: 5
print(f"Escopo global: {x}")  # Output: Escopo global: 10
```

MÓDULOS

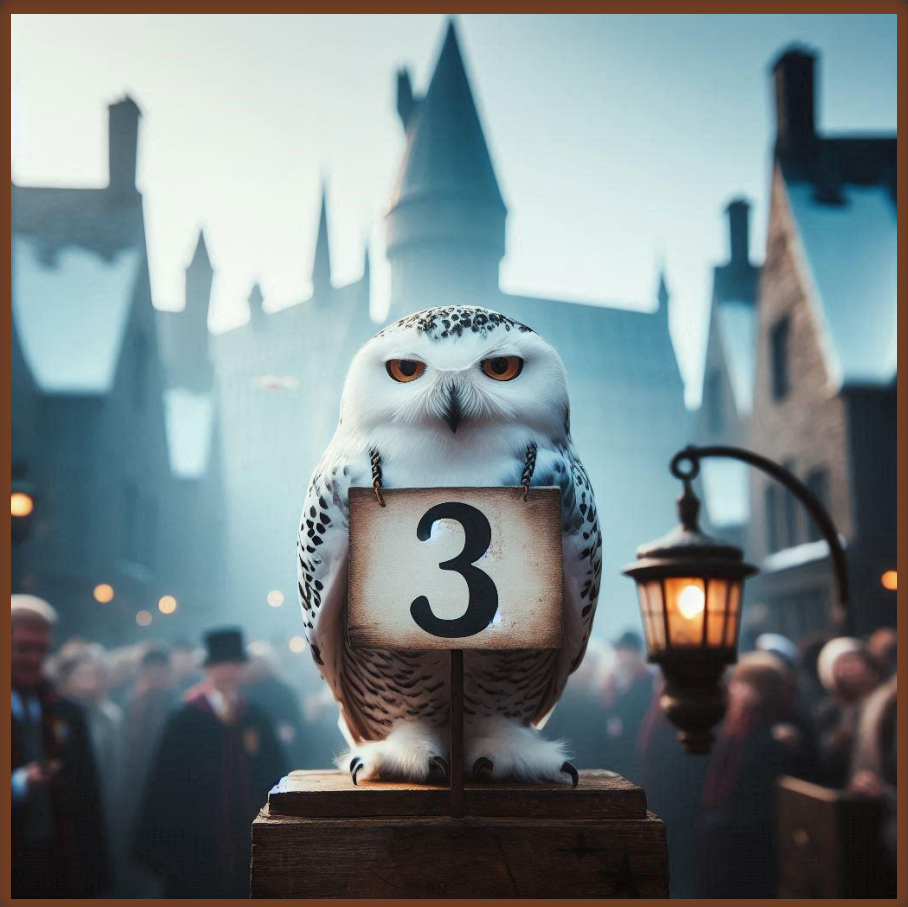
Importando Módulos

Módulos são arquivos Python que contêm definições e instruções. Você pode importar módulos usando a palavra-chave import.

```python
import math

print(math.sqrt(16))
```

CAPÍTULO



PROGRAMAÇÃO ORIENTADA A OBJETOS

CLASSES E OBJETOS

A programação orientada a objetos (POO) é um paradigma de programação que utiliza "objetos" - instâncias de classes - para representar dados e métodos. POO permite criar programas mais estruturados e fáceis de manter.

Classe

Em Python, uma classe é definida usando a palavra-chave class. Dentro de uma classe, você pode definir atributos (variáveis) e métodos (funções).

Objetos

Um objeto é uma instância de uma classe, contendo valores específicos para os atributos definidos na classe e podendo executar os métodos definidos nela.

```python
class Carro:
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def exibir_detalhes(self):
        return f"{self.marca} {self.modelo} {self.ano}"

meu_carro = Carro("Toyota", "Corolla", 2020)
print(meu_carro.exibir_detalhes())  # Output: Toyota Corolla 2020
```

Em Python, uma classe é definida usando a palavra-chave **class**. Dentro de uma classe, você pode definir atributos (variáveis) e métodos (funções).

CLASSES E OBJETOS

Atributos de Instância e de Classe

Atributos de instância são variáveis que pertencem a uma instância específica da classe, enquanto atributos de classe são compartilhados por todas as instâncias da classe.

```python
class Pessoa:
    especie = "Humano"  # Atributo de classe

    def __init__(self, nome, idade):
        self.nome = nome  # Atributo de instância
        self.idade = idade  # Atributo de instância

pessoa1 = Pessoa("João", 30)
pessoa2 = Pessoa("Maria", 25)

print(pessoa1.especie)  # Output: Humano
print(pessoa2.especie)  # Output: Humano
```

CLASSES E OBJETOS

Métodos de Instância e Métodos de Classe

Métodos de instância operam em uma instância específica da classe, enquanto métodos de classe operam na própria classe. Métodos de classe são definidos usando o decorador **@classmethod**.

```python
class Pessoa:
    especie = "Humano"

    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def exibir_detalhes(self):
        return f"{self.nome}, {self.idade} anos"

    @classmethod
    def especie_info(cls):
        return f"Esta classe representa a espécie {cls.especie}"

print(Pessoa.especie_info())  # Output: Esta classe representa a espécie Humano
```

HERANÇA E POLIMORFISMO

Herança

Herança permite que uma classe herde atributos e métodos de outra classe, promovendo reuso e extensão de código.

Polimorfismo

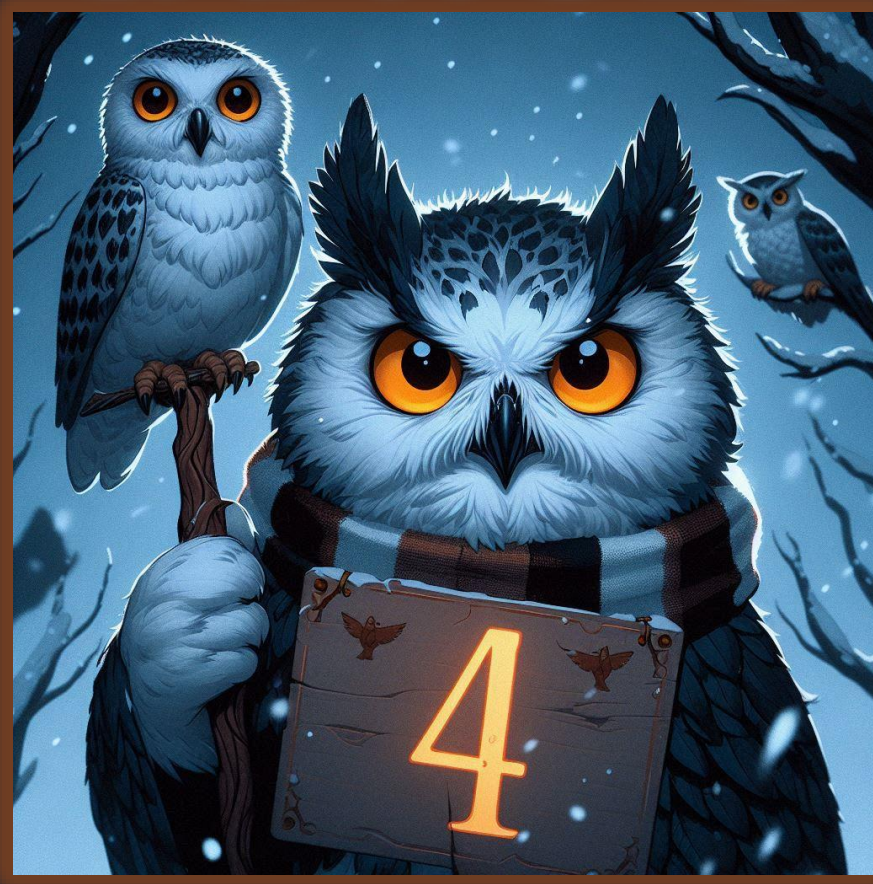
Polimorfismo permite que métodos em diferentes classes tenham o mesmo nome, mas comportamentos diferentes, facilitando a intercambialidade de objetos.

```python
class Animal:
    def som(self):
        return "O animal faz um som"

class Cachorro(Animal):
    def som(self):
        return "O cachorro late"

meu_animal = Cachorro()
print(meu_animal.som())  # Output: O cachorro late
```

CAPÍTULO



TRABALHANDO COM BIBLIOTECAS

BIBLIOTECAS

Bibliotecas Padrão

Python possui uma vasta biblioteca padrão, incluindo módulos para manipulação de strings, coleções, entrada/saída e muito mais.

```python
import os

print(os.listdir("."))
```

Bibliotecas Externas

Python possui um rico ecossistema de bibliotecas externas, como NumPy para computação científica, Pandas para análise de dados e Django para desenvolvimento web.

```python
import numpy as np

a = np.array([1, 2, 3])
print(a * 2)
```

CAPÍTULO

MANIPULAÇÃO DE EXCEÇÕES

TRATAMENTO DE EXCEÇÕES

Try - Except

Python utiliza try-except para tratar exceções, garantindo que erros sejam gerenciados de maneira controlada.

```
try:
    resultado = 10 / 0
except ZeroDivisionError:
    print("Erro: divisão por zero.")
finally:
    print("Bloco finally executado.")
```

OBRIGADO POR LER ATE AQUI !

Esse Ebook foi gerado por Inteligência Artificial
e diagramado por um humano.

https://github.com/stneraniecarvaiho/prompts-ebook