

Cybertron Coding Academy

Códigos de Cybertron: Guia Básico de Python



01

Introdução

Dominando Python: Do Básico ao Avançado

Introdução ao Python

Python é uma linguagem de programação versátil e poderosa, amplamente utilizada em diversas áreas como desenvolvimento web, ciência de dados, automação e inteligência artificial. Este ebook é projetado para guiá-lo do básico ao avançado, com explicações claras e exemplos de código práticos e reais.



Dominando Python: Do Básico ao Avançado

Seu Primeiro Programa

Vamos começar com um simples "Hello, World!".

```
HelloWord.py

1 print("Hello world")
2
```

Este código imprime a frase "Hello, World!" na tela. Simples, não?



02

Fundamentos do Python

Fundamentos do Python

Variáveis e Tipos de Dados

Em Python, você não precisa declarar o tipo da variável explicitamente. Veja alguns exemplos:

```
TipoDado.py

1 # Inteiro
2 idade = 30
3
4 # Float
5 altura = 1.75
6
7 # String
8 nome = "João"
9
10 # Booleano
11 eh_programador = True
12
13
```



Fundamentos do Python

Operadores Básicos

Python suporta operadores aritméticos, de comparação e lógicos.

```
Operadores.py

1 # Aritméticos
2 soma = 10 + 5
3 subtracao = 10 - 5
4 multiplicacao = 10 * 5
5 divisao = 10 / 5
6
7 # Comparação
8 maior = 10 > 5
9 igual = 10 == 10
10
11 # Lógicos
12 e = True and False
13 ou = True or False
14
```



03

Estruturas de Controle

Estruturas de Controle

Condicionais

As estruturas condicionais permitem executar blocos de código com base em condições.



EstruturasDecisão.py

```
1  idade = 18
2
3  if idade >= 18:
4      print("Você é maior de idade.")
5  else:
6      print("Você é menor de idade.")
7
```



Estruturas de Controle

Loops

Loops permitem repetir um bloco de código várias vezes.

```
Loops.py

1 # Loop for
2 for i in range(5):
3     print(i)
4
5 # Loop while
6 contador = 0
7 while contador < 5:
8     print(contador)
9     contador += 1
10
```



04

Funções

Funções

Definindo Funções

Funções são blocos de código reutilizáveis.



Funções.py

```
1 def saudacao(nome):  
2     print(f"Olá, {nome}!")  
3  
4 saudacao("Maria")  
5
```



Funções

Funções com Retorno

Funções podem retornar valores.

```
Funções.py

1 def soma(a, b):
2     return a + b
3
4 resultado = soma(10, 20)
5 print(resultado)
6
```



05

Trabalhando com Listas

Trabalhando com Listas

Listas

Listas são coleções ordenadas de itens.



Listas.py

```
1  numeros = [1, 2, 3, 4, 5]
2
3  # Acessando elementos
4  print(numeros[0]) # 1
5
6  # Adicionando elementos
7  numeros.append(6)
8
9  # Removendo elementos
10 numeros.remove(3)
11
```



Trabalhando com Listas

List Comprehensions

List comprehensions são uma maneira concisa e eficiente de criar listas. Elas permitem gerar listas a partir de outras listas ou qualquer objeto iterável em uma única linha de código, com a possibilidade de adicionar condições.

A estrutura básica de uma list comprehension é:

```
1 nova_lista = [expressao for item in iteravel]
2
3
```

- Expressao: A expressão a ser avaliada para cada item (pode incluir operações ou funções).
- Item: O item atual do iterável.
- Iteravel: Qualquer objeto que possa ser iterado (lista, tupla, string, etc.).



06

Dicionários

Dicionários

Criando Dicionários

Dicionários são coleções de pares chave-valor. Cada valor no dicionário é associado a uma chave única, que pode ser usada para acessar esse valor.

Você pode criar dicionários usando chaves {} e separando chaves e valores com dois pontos :.

```
Dicionarios.py

1 # Criando um dicionário
2 pessoa = {"nome": "Ana", "idade": 25, "cidade": "São Paulo"}
3
4 # Acessando valores
5 print(pessoa["nome"]) # Saída: Ana
6
7 # Adicionando novos pares
8 pessoa["profissao"] = "Engenheira"
9 print(pessoa) # Saída: {'nome': 'Ana', 'idade': 25, 'cidade': 'São Paulo', 'profissao': 'Engenheira'}
10
```



Dicionários

Métodos de Dicionários

Dicionários têm vários métodos úteis para manipulação. Veja a seguir algumas maneiras:

Dicionarios.py

```
1 # Obtendo todas as chaves
2 print(pessoa.keys()) # Saída: dict_keys(['nome', 'idade', 'cidade', 'profissao'])
3
4 # Obtendo todos os valores
5 print(pessoa.values()) # Saída: dict_values(['Ana', 25, 'São Paulo', 'Engenheira'])
6
7 # Verificando a existência de uma chave
8 print("idade" in pessoa) # Saída: True
9
10 # Removendo um par chave-valor
11 pessoa.pop("cidade")
12 print(pessoa) # Saída: {'nome': 'Ana', 'idade': 25, 'profissao': 'Engenheira'}
13
```



07

Manipulação de Arquivos

Manipulação de Arquivos

Leitura e Escrita de Arquivos

Manipular arquivos é uma tarefa comum em Python, seja para ler dados de um arquivo ou para escrever dados em um arquivo.

Vamos escrever um texto em um arquivo.

ManipularArquivos.py

```
1 with open("exemplo.txt", "w") as arquivo:
2     arquivo.write("Olá, Mundo!")
3
```



Manipulação de Arquivos

Lendo de um Arquivo

Agora, vamos ler o conteúdo do arquivo que acabamos de criar.

ManipularArquivos.py

```
1 with open("exemplo.txt", "r") as arquivo:
2     conteudo = arquivo.read()
3     print(conteudo)
4 # Saída: Olá, Mundo!
5
```



Manipulação de Arquivos

Lendo Arquivo Linha por Linha

Para ler um arquivo linha por linha, você pode usar um loop.

```
ManipularArquivos.py

1 with open("exemplo.txt", "r") as arquivo:
2     for linha in arquivo:
3         print(linha.strip()) # .strip() remove os espaços em branco e novas linhas
4
```

Contagem de Linhas

Vamos contar o número de linhas em um arquivo.

```
ManipularArquivos.py

1 contador_de_linhas = 0
2
3 with open("exemplo.txt", "r") as arquivo:
4     for linha in arquivo:
5         contador_de_linhas += 1
6
7 print(f"O arquivo tem {contador_de_linhas} linhas.")
8 # Saída: O arquivo tem 1 linhas.
9
```



08

Programação Orientada a Objetos

Programação Orientada a Objetos (POO)

Classes e Objetos

A Programação Orientada a Objetos (POO) é um paradigma de programação que utiliza "objetos" e suas interações para projetar aplicativos e programas. Em Python, quase tudo é um objeto, com suas propriedades e métodos.

Uma classe é como um molde para criar objetos. Veja como criar uma classe:

```
POO.py

1 class Pessoa:
2     def __init__(self, nome, idade):
3         self.nome = nome
4         self.idade = idade
5
6     def apresentar(self):
7         print(f"Olá, eu sou {self.nome} e tenho {self.idade} anos.")
8
```



Programação Orientada a Objetos (POO)

Criando Objetos

Agora vamos criar um objeto a partir da classe Pessoa.

```
POO.py

1 pessoa1 = Pessoa("Carlos", 30)
2 pessoa1.apresentar()
3 # Saída: Olá, eu sou Carlos e tenho 30 anos.
4
```



Programação Orientada a Objetos (POO)

Herança

Herança permite criar uma nova classe com base em uma classe existente.

```
POO.py

1 class Funcionario(Pessoa):
2     def __init__(self, nome, idade, cargo):
3         super().__init__(nome, idade)
4         self.cargo = cargo
5
6     def apresentar(self):
7         super().apresentar()
8         print(f"Meu cargo é {self.cargo}.")
9
10 funcionario1 = Funcionario("Ana", 28, "Engenheira")
11 funcionario1.apresentar()
12 # Saída:
13 # Olá, eu sou Ana e tenho 28 anos.
14 # Meu cargo é Engenheira.
15
```



09

Bibliotecas e Módulos

Bibliotecas e Módulos

Importando Módulos

Python possui uma vasta quantidade de bibliotecas e módulos que você pode importar e usar.

Módulo '**math**' Vamos usar o módulo '**math**' para realizar algumas operações matemáticas.

Bibliotecas.py

```
1 import math
2
3 # Raiz quadrada
4 print(math.sqrt(16)) # Saída: 4.0
5
6 # Fatorial
7 print(math.factorial(5)) # Saída: 120
8
```



Bibliotecas e Módulos

Bibliotecas Populares

- NumPy;
- Matplotlib;
- Pandas;

NumPy é uma biblioteca para computação científica em Python.

```
Bibliotecas.py

1 import numpy as np
2
3 # Criando um array
4 array = np.array([1, 2, 3, 4, 5])
5 print(array)
6 # Saída: [1 2 3 4 5]
7
8 # Operações com arrays
9 print(array * 2)
10 # Saída: [ 2  4  6  8 10]
11
```



Bibliotecas e Módulos

Bibliotecas Populares

- NumPy;
- Matplotlib;
- Pandas;

Pandas é uma biblioteca para análise de dados.

```
Bibliotecas.py

1 import pandas as pd
2
3 # Criando um DataFrame
4 dados = {
5     "Nome": ["Ana", "Carlos", "João"],
6     "Idade": [25, 30, 22]
7 }
8 df = pd.DataFrame(dados)
9 print(df)
10 # Saída:
11 #      Nome  Idade
12 # 0     Ana    25
13 # 1  Carlos    30
14 # 2   João    22
15
```



Bibliotecas e Módulos

Bibliotecas Populares

- NumPy;
- Matplotlib;
- Pandas;

Matplotlib é uma biblioteca para visualização de dados.

```
Bibliotecas.py

1 import matplotlib.pyplot as plt
2
3 # Dados
4 x = [1, 2, 3, 4, 5]
5 y = [2, 3, 5, 7, 11]
6
7 # Criando um gráfico
8 plt.plot(x, y)
9 plt.xlabel('X')
10 plt.ylabel('Y')
11 plt.title('Gráfico Simples')
12 plt.show()
13
```



10

Tópicos Avançados

Tópicos Avançados

Expressões Lambda

Funções lambda são funções anônimas e rápidas, definidas usando a palavra-chave lambda.

Bibliotecas.py

```
1 dobro = lambda x: x * 2
2 print(dobro(5))
3 # Saída: 10
4
```



Tópicos Avançados

List Comprehensions Avançadas

List comprehensions podem ser usadas para operações complexas em uma linha de código.

Exemplo: Filtrando e Transformando

Bibliotecas.py

```
1 numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 pares_ao_quadrado = [x**2 for x in numeros if x % 2 == 0]
3 print(pares_ao_quadrado)
4 # Saída: [4, 16, 36, 64, 100]
5
```



Tópicos Avançados

Manipulação de Erros

Tratamento de exceções permite que seu programa lide com erros de maneira controlada.

Bibliotecas.py

```
1 try:
2     numero = int(input("Digite um número: "))
3 except ValueError:
4     print("Valor inválido. Por favor, insira um número inteiro.")
5 else:
6     print(f"Você digitou o número {numero}.")
7 finally:
8     print("Execução do bloco try-finally concluída.")
9
```



Agradecimentos

Obrigado por ler até aqui

```
Agradecimento.py

1 import time
2 import os
3
4 def clear_console():
5     # Limpa a tela do console
6     os.system('cls' if os.name == 'nt' else 'clear')
7
8 def typing_effect(text):
9     # Função para criar efeito de digitação
10    for char in text:
11        print(char, end='', flush=True)
12        time.sleep(0.1)
13    print()
14
15 def main():
16     clear_console()
17
18     thank_you_message = """
19     *****
20     *                                     *
21     *           Obrigado por ler nosso ebook!           *
22     *                                     *
23     *   Esperamos que tenha gostado e aprendido!   *
24     *                                     *
25     *           - Equipe de Desenvolvimento           *
26     *                                     *
27     *****
28     """
29
30     # Exibe a mensagem de agradecimento com efeito de digitação
31     typing_effect("Gerando sua mensagem de agradecimento")
32     time.sleep(1)
33     clear_console()
34     typing_effect(thank_you_message)
35
36 if __name__ == "__main__":
37     main()
38
```



Obrigado por ler até aqui

Este ebook foi criado por uma IA e diagramado por um humano. As etapas do ebook se encontram no meu GitHub.



<https://github.com/Lucas-Casa-Mausa>

