Código Infinito

python

para novatos Poderosos

CHARLES LIMA

01

VARIÁVEIR E TIPOS DE DADOS

Variáveis e Tipos de Dados

Entendendo os Blocos de Construção do Python

As variáveis são usadas para armazenar dados, enquanto os tipos de dados determinam o que pode ser feito com esses valores.
Exemplos:

Untitled-1

```python
# Declarando variáveis
nome = "Alice"  # Tipo string
idade = 30      # Tipo inteiro
altura = 1.65   # Tipo float
ativo = True    # Tipo booleano

# Exibindo informações
print(f"Nome: {nome}, Idade: {idade}, Altura: {altura}, Ativo: {ativo}")
```

Explicação:

- Strings são usadas para texto.
- Inteiros são usados para números sem decimais.
- Floats são números com ponto decimal.
- Booleanos representam valores verdadeiros ou falsos.

Variáveis e Tipos de Dados

Operadores com Variáveis

Untitled-1

```
# Operações Matemáticas
soma = 10 + 5          # Resultado: 15
multiplicacao = 2 * 3  # Resultado: 6

divisao = 15 / 3       # Resultado: 5.0
modulo = 10 % 3        # Resultado: 1

# Operações Lógicas
maior = 10 > 5         # Resultado: True
igual = 5 == 5         # Resultado: True
```

Essas operações são a base para muitos cálculos e comparações em seus programas.

02

Listas, Tuplas e Dicionários

Listas, Tuplas e Dicionários

Armazenando e Organizando Dados

Essas estruturas são fundamentais para armazenar coleções de dados de forma organizada.

Listas:

Untitled-1

```python
# Lista de frutas
frutas = ["maçã", "banana", "laranja"]
frutas.append("uva")  # Adiciona um novo elemento
print(frutas)          # Resultado: ['maçã', 'banana', 'laranja', 'uva']

# Remover um item
frutas.remove("banana")
print(frutas)  # Resultado: ['maçã', 'laranja', 'uva']
```

Tuplas:

Untitled-1

```python
# Tupla imutável de coordenadas
coordenadas = (10, 20)
print(coordenadas[0])  # Resultado: 10

# Tentativa de modificar gera erro:
# coordenadas[0] = 15  # Erro: 'tuple' object does not support item assignment
```

Dicionários:

Untitled-1

```python
# Dicionário com dados pessoais
pessoa = {"nome": "Carlos", "idade": 28}
pessoa["altura"] = 1.75  # Adicionando um novo par chave-valor
print(pessoa)  # Resultado: {'nome': 'Carlos', 'idade': 28, 'altura': 1.75}

# Iterar pelas chaves e valores
for chave, valor in pessoa.items():
```

03

Condicionais e Laços de Repetição

Condicionais e Laços de Repetição

Controlando o Fluxo do Programa

Condicionais:

Untitled-1

```python
idade = 18
if idade >= 18:
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")

# Condicional com elif
nota = 85
if nota >= 90:
    print("A")
elif nota >= 80:
    print("B")
else:
    print("C")
```

Condicionais e Laços de Repetição

Controlando o Fluxo do Programa

Laço For:

Untitled-1

```python
for numero in range(1, 6):
    print(numero)  # Resultado: 1, 2, 3, 4, 5

# Iterando sobre uma lista
animais = ["gato", "cachorro", "papagaio"]
for animal in animais:
    print(animal)
```

Laço While:

Untitled-1

```python
contador = 0
while contador < 5:
    print(contador)
    contador += 1

# Loop infinito (cuidado!)
# while True:
#     print("Isso nunca termina!")
```

Dicas para Laços

Evite loops infinitos e utilize a função **break** para sair de um laço antecipadamente:

Untitled-1

```python
for numero in range(10):
    if numero == 5:
        break  # Sai do loop quando número for 5
    print(numero)
```

04

Funções e Módulos

Funções e Módulos

Organizando e Reutilizando Código

Funções:

Untitled-1

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

print(saudacao("Maria"))  # Resultado: Olá, Maria!

# Função com valor padrão
def somar(a, b=10):
    return a + b

print(somar(5))  # Resultado: 15
```

Módulos

Untitled-1

```python
# Importando o módulo math
import math
print(math.sqrt(16))  # Resultado: 4.0

# Criando seu próprio módulo (arquivo meu_modulo.py)
def dobro(x):
    return x * 2

# Usando o módulo
from meu_modulo import dobro
print(dobro(4))  # Resultado: 8
```

Explicação:

- Funções encapsulam lógica reutilizável.
- Módulos permitem reutilizar código em diferentes partes do programa, promovendo organização.

05

Manipulação de Arquivos

Manipulação de Arquivos

Trabalhando com Dados Externos

Leitura de Arquivos:

Untitled-1

```
with open("dados.txt", "r") as arquivo:
    conteudo = arquivo.read()
    print(conteudo)

# Lendo linha por linha
with open("dados.txt", "r") as arquivo:
    for linha in arquivo:
        print(linha.strip())
Escrita em Arquivos:
with open("saida.txt", "w") as arquivo:
    arquivo.write("Este é um exemplo de escrita em arquivo.")

# Acrescentando ao arquivo
with open("saida.txt", "a") as arquivo:
    arquivo.write("\nNova linha adicionada.")
```

Escrita em Arquivos:

Untitled-1

```
with open("saida.txt", "w") as arquivo:
    arquivo.write("Este é um exemplo de escrita em arquivo.")

# Acrescentando ao arquivo
with open("saida.txt", "a") as arquivo:
    arquivo.write("\nNova linha adicionada.")
```

Manipulação de Arquivos

Manipulação Avançada

Para trabalhar com arquivos JSON:

Untitled-1

```python
import json

# Escrevendo em JSON
dados = {"nome": "Alice", "idade": 25}
with open("dados.json", "w") as arquivo:
    json.dump(dados, arquivo)

# Lendo de JSON
with open("dados.json", "r") as arquivo:
    dados = json.load(arquivo)
    print(dados)
```

Conclusão

Python é uma linguagem incrivelmente poderosa e flexível, e este guia cobriu os fundamentos essenciais para você iniciar sua jornada. Agora é sua vez de praticar! Teste os exemplos, crie seus próprios projetos e explore mais sobre Python.

Recursos Adicionais

- Documentação Oficial do Python
- Cursos Gratuitos no Python.org
- Comunidade no Reddit: r/Python