Feitiçaria Digital

Domine Python - do Zero ao Jujutsu

Alan Ambrozio

01

COMEÇANDO COM PYTHON

Exibindo Mensagens: ‘print’

O comando ‘print’ é usado para exibir mensagens ou valores na tela.

O comando print é usado para exibir mensagens ou valores na tela. Este é um dos primeiros comandos que se aprende ao iniciar com Python. Ele é muito útil para depuração e para fornecer informações ao usuário.

print.py

```
print("Bem-vindo ao curso de Python!")
```

Neste exemplo, a mensagem "Bem-vindo ao curso de Python!" será exibida na tela.

Comentários no Código

Comentários ajudam a explicar o código e são ignorados pelo Python.

Eles são importantes para tornar o código mais compreensível para outras pessoas (ou para você mesmo no futuro). Use # para adicionar um comentário.

comentario.py

```
# Isso é um comentário
print("Comentários são úteis para documentação.")
```

Neste exemplo, a linha que começa com '#' é um comentário e não será executada pelo Python.

02

Trabalhando com Variáveis e Tipos de Dados

Armazenando Valores: Variáveis

Variáveis são usadas para armazenar valores que podem ser reutilizados.

Você pode pensar nelas como "caixinhas" onde você guarda informações. Em Python, você não precisa declarar o tipo da variável explicitamente, ele é inferido automaticamente.

variaveis.py

```python
nome = "Carlos"
idade = 30
print(f"Nome: {nome}, Idade: {idade}")
```

Neste exemplo, a variável ‘nome’ armazena a string "Carlos" e a variável ‘idade armazena o número 30. O comando ‘print’ exibe esses valores na tela.

Tipos de Dados Básicos

Python suporta vários tipos de dados como strings, inteiros, floats e booleanos.

É importante conhecer esses tipos para usar a variável adequada para cada situação.

tipo_dados.py

```
# String
cidade = "São Paulo"

# Inteiro
ano = 2024

# Float
temperatura = 23.5

# Booleano
chovendo = False
```

Aqui, 'cidade' é uma string, ano é um inteiro, 'temperatura' é um float e 'chovendo' é um booleano. Cada tipo de dado é usado para representar diferentes tipos de informações.

03

Estruturas de Dados

Listas: Coleções de Itens

Listas permitem armazenar múltiplos valores em uma única variável.

Elas são úteis para armazenar coleções de itens, como uma lista de frutas ou números.

listas.py

```
frutas = ["maçã", "banana", "cereja"]
print(f"Frutas disponíveis: {frutas}")
```

Neste exemplo, a lista ‘frutas’ armazena três strings. O comando ‘print’ exibe a lista completa na tela.

Dicionários: Pares Chave-Valor

Dicionários armazenam dados em pares de chave-valor.

Permitindo associar um valor a uma chave específica. Eles são úteis para armazenar informações mais complexas, como os atributos de uma pessoa.

dicionario.py

```
pessoa = {"nome": "Ana", "idade": 22, "cidade": "Rio de Janeiro"}
print(f"Nome: {pessoa['nome']}, Cidade: {pessoa['cidade']}")
```

Aqui, o dicionário pessoa contém três pares chave-valor. O comando 'print' exibe o nome e a cidade armazenados no dicionário.

04

Controle de Fluxo

Condicionais: Tomando Decisões

Condicionais permitem executar código baseado em condições.

Usamos if, elif e else para tomar decisões no código. Isso é útil quando precisamos que nosso programa responda de maneiras diferentes dependendo da entrada do usuário ou de outras condições.

condicoes.py

```
idade = 20
if idade >= 18:
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")
```

Neste exemplo, se a variável 'idade' for maior ou igual a 18, o programa exibirá "Você é maior de idade." Caso contrário, exibirá "Você é menor de idade."

Loops: Repetindo Ações

Loops permitem repetir um bloco de código várias vezes.

O que é útil para automatizar tarefas repetitivas.
Usando 'for'
O loop for é usado para iterar sobre uma sequência (como uma lista, tupla ou string).

repeticoes.py

```python
for fruta in ["maçã", "banana", "cereja"]:
    print(f"Eu gosto de {fruta}.")
```

Neste exemplo, o loop 'for' percorre cada item na lista de frutas e imprime uma mensagem para cada um.
Usando 'while'
O loop 'while' repete um bloco de código enquanto uma condição for verdadeira.

repeticoes.py

```python
contador = 1
while contador <= 3:
    print(f"Contando: {contador}")
    contador += 1
```

Neste exemplo, o loop 'while' imprime o valor do 'contador' enquanto ele for menor ou igual a 3. A variável 'contador' é incrementada a cada iteração.

05

Funções

Criando Funções: Reutilizando Código

Funções agrupam comandos em blocos reutilizáveis.

Permitindo que você execute o mesmo código em diferentes partes do programa sem repeti-lo.

funcoes.py

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

saudacao("Mariana")
```

Neste exemplo, a função ‘saudacao’ recebe um parâmetro ‘nome’ e imprime uma mensagem de saudação. A função é chamada com o argumento "Mariana".

06

Módulos e Bibliotecas

Importando Módulos

Módulos são bibliotecas de código que adicionam funcionalidades ao seu programa.

O Python possui uma vasta coleção de módulos integrados, além de permitir a instalação de módulos externos.

biblioteca.py

```python
import math

print(f"A raiz quadrada de 16 é {math.sqrt(16)}")
```

Aqui, o módulo ‘math’ é importado para usar a função ‘sqrt’, que calcula a raiz quadrada de um número.

07

Tratamento de Exceções

Lidando com Erros

O tratamento de exceções lida com erros que ocorrem durante a execução do programa.

Permitindo que você controle o que acontece quando ocorre um erro.

excecoes.py

```
try:
    numero = int(input("Digite um número: "))
    print(f"Você digitou: {numero}")
except ValueError:
    print("Por favor, digite um número válido.")
```

Neste exemplo, o bloco 'try' tenta converter a entrada do usuário para um inteiro. Se ocorrer um 'ValueError' (por exemplo, se o usuário digitar texto em vez de um número), o bloco 'except' será executado, exibindo uma mensagem de erro.

END

AGRADECIMENTOS

Obrigado por ler até aqui!

Este guia apresentou os principais comandos do Python de forma simples e prática. Com esses conhecimentos, você está pronto para explorar mais da linguagem e criar seus próprios programas. Continue praticando e experimentando com Python!

Ebook criado utilizando o ChatGPT e revisado por Alan César Ambrozio.
Imagem da capa criada utilizando OpenArt.

https://github.com/alanambrozio/ebook-DIO-AI