Python Assault

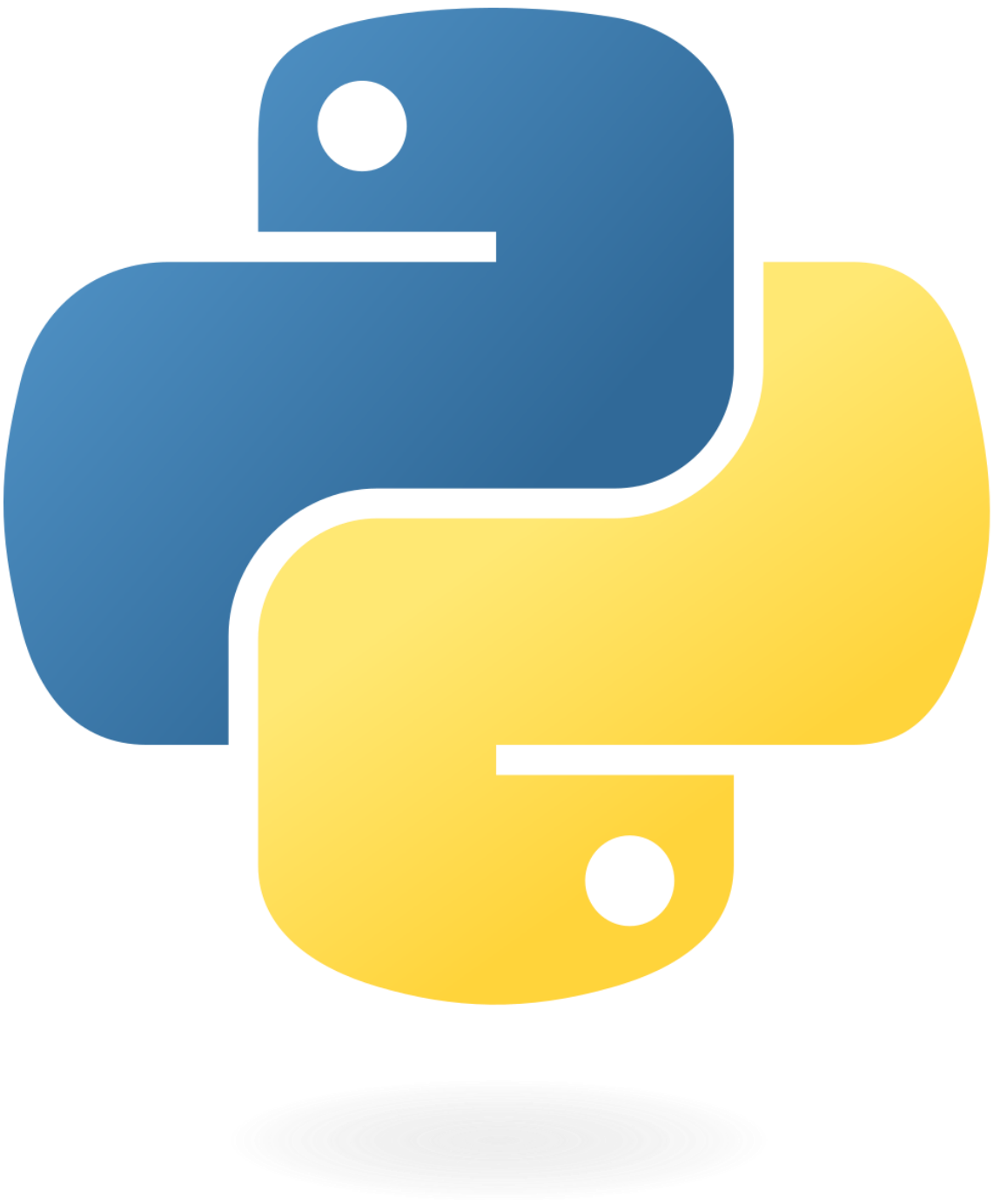
Desbrave os Campos de Batalha da Codificação

GABRIEL ELIAS SILVA

Guia Prático

Principais Regras do Python

Python é uma das linguagens de programação mais populares e acessíveis do mundo. Suas regras são claras e consistentes, o que a torna ideal tanto para iniciantes quanto para desenvolvedores experientes. Neste guia, vamos explorar as principais regras do Python com exemplos práticos de código



01

INDENTAÇÃO

Indentação

A indentação em Python é fundamental. Diferente de outras linguagens que utilizam chaves {} para definir blocos de código, Python usa a indentação. O nível de indentação determina o escopo do código.

```python
if True:
    print("Este bloco está corretamente indentado.")
    if False:
        print("Este não será exibido.")
    print("Este será exibido.")
```

02

DECLARAÇAO DE VARIÁVEIS

Declaração de Variáveis

Em Python, você não precisa declarar o tipo da variável. Python é uma linguagem dinamicamente tipada, o que significa que você pode atribuir valores de diferentes tipos à mesma variável em momentos diferentes.

Python

```python
idade = 25              # Inteiro
nome = "Alice"          # String
altura = 1.75           # Float
print(idade, nome, altura)
```

03

TIPOS DE DADOS

Tipos de Dados

Python oferece vários tipos de dados integrados, como inteiros, floats, strings, listas, tuplas, dicionários e conjuntos.

- Listas são mutáveis (podem ser alteradas), enquanto tuplas são imutáveis.
- Dicionários armazenam pares de chave-valor.
- Conjuntos são coleções de elementos únicos.

Python

```python
inteiro = 10
flutuante = 20.5
texto = "Olá, Python!"
lista = [1, 2, 3, 4, 5]
tupla = (1, 2, 3)
dicionario = {"nome": "Alice", "idade": 25}
conjunto = {1, 2, 3, 4, 5}
print(inteiro, flutuante, texto, lista, tupla, dicionario, conjunto)
```

04

ESTRUTURAS DE CONTROLE

Estruturas de Controle

Estruturas de controle como if, elif e else são usadas para tomar decisões no código.

- As condições são avaliadas sequencialmente.
- O bloco elif (else if) é opcional e pode haver vários em uma estrutura.
- O bloco else é executado se nenhuma das condições anteriores for verdadeira.

Python

```python
x = 10
if x > 0:
    print("x é positivo")
elif x == 0:
    print("x é zero")
else:
    print("x é negativo")
```

05

LOOPS

Loops

Python suporta loops for e while para iterar sobre sequências de dados ou executar blocos de código repetidamente.

- O for é geralmente usado para iterar sobre uma sequência (lista, tupla, dicionário, conjunto ou string).
- O while é usado quando a quantidade de iterações não é conhecida previamente e depende de uma condição.

Python

```python
# Usando for
for i in range(5):
    print(f"For loop: {i}")

# Usando while
i = 0
while i < 5:
    print(f"While loop: {i}")
    i += 1
```

06

FUNÇOES

Funções

Funções permitem encapsular blocos de código que podem ser reutilizados. Definimos funções usando def e retornamos valores com return.

- Funções podem ter parâmetros opcionais com valores padrão.
- Parâmetros podem ser passados por posição ou por nome.
- Funções podem retornar múltiplos valores utilizando tuplas

Python

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

print(saudacao("Mundo"))
```

07

MANIPULAÇAO DE STRINGS

Manipulação de Strings

Python oferece várias operações e métodos para manipulação de strings, como alterar maiúsculas e minúsculas, substituir substrings, fatiar e formatar.

- - Strings são imutáveis, o que significa que qualquer operação que modifique a string retorna uma nova string.
- - Fatiamento de strings permite extrair substrings utilizando índices.

Python

```python
texto = "Python é incrível!"
print(texto.upper())
print(texto.lower())
print(texto.replace("incrível", "fantástico"))
print(texto[0:6])  # Fatiamento: obtém "Python"
```

08

LISTAS

Listas

Listas são coleções ordenadas que podem conter elementos de diferentes tipos e são mutáveis.

- - Listas podem ser aninhadas, permitindo a criação de listas de listas.
- - Métodos comuns incluem `append()`, `remove()`, `sort()`, `reverse()`, `extend()`, `insert()`, entre outros

Python

```python
frutas = ["maçã", "banana", "cereja"]
frutas.append("laranja")
print(frutas)
frutas.remove("banana")
print(frutas)
frutas.sort()
print(frutas)
```

09

DICIONARIOS

Dicionários

Dicionários armazenam pares de chave-valor e são mutáveis. As chaves devem ser imutáveis e únicas.

- - Dicionários são ideais para mapeamentos e armazenamento de dados associados.
- - Métodos comuns incluem `keys()`, `values()`, `items()`, `get()`, `pop()`, `update()`, entre outros.

Python

```python
aluno = {"nome": "Carlos", "idade": 23, "curso": "Engenharia"}
print(aluno["nome"])
aluno["idade"] = 24
aluno["cidade"] = "São Paulo"
print(aluno)
del aluno["curso"]
print(aluno)
```

LIST COMPREHENSIONS

List Comprehensions

List comprehensions oferecem uma maneira concisa de criar listas. Elas podem incluir condições para filtrar elementos.

- - List comprehensions podem substituir loops `for` e funções `map`/`filter` de maneira mais legível.
- - Podem ser aninhadas para criar listas multidimensionais.

Python

```python
quadrados = [x**2 for x in range(10)]
print(quadrados)

pares = [x for x in range(20) if x % 2 == 0]
print(pares)
```

AGRADECIMENTOS

Obrigado por ler até aqui

Esse Ebook foi gerado por IA, e diagramado por humano.
O passo a passo se encontra no meu Github

.

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.

https://github.com/gabbsmano