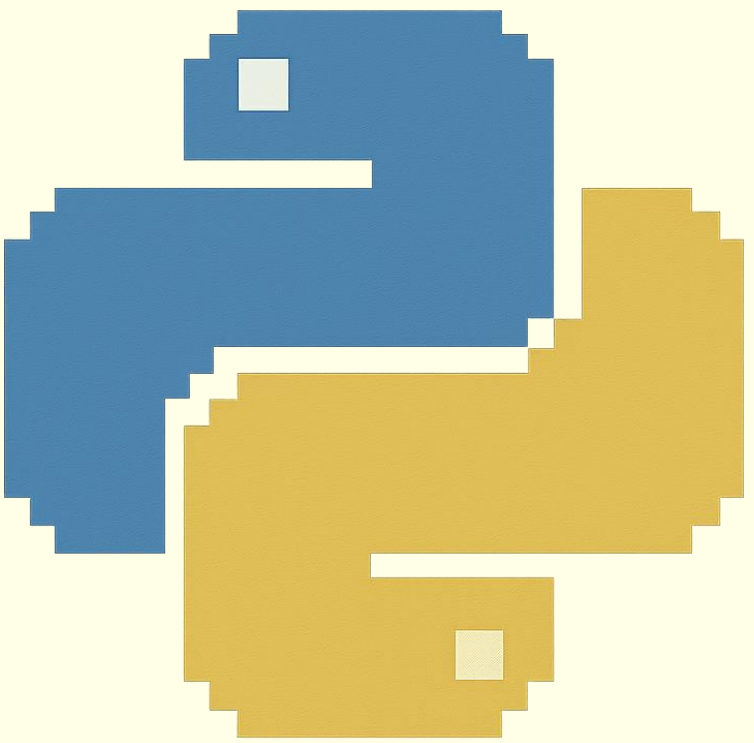
DESVENDANDO PYTHON

A LÓGICA POR TRÁS DA SERPENTE

MATHEUS TERCIOTE

Desvendando Python: Seletores na Prática

Manipular dados com clareza e eficiência é uma das maiores forças do Python. Este capítulo apresenta os seletores mais usados da linguagem — explicados com simplicidade e aplicados a contextos reais de programação.



1.

Seletores na Prática

Acessando dados com índice

Em estruturas sequenciais como listas, strings e tuplas, usamos índices numéricos para acessar elementos. O índice começa do zero.

Desvendando Python.py

```
linguagens = ['Python', 'JavaScript', 'C++']
print(linguagens[0])  # Saída: Python
```

Você também pode acessar valores do final para o início com índices negativos:

Desvendando Python.py

```
print(linguagens[-1])  # Saída: C++
```

Exemplo real: acessar a última linguagem usada por um programador em um sistema de histórico.

Fatiando partes de dados (Slicing)

O fatiamento permite pegar partes de uma sequência com a sintaxe **obj[início:fim:passo]**

Desvendando Python.py

```
mensagem = "Erro 404: Página não encontrada"
print(mensagem[:8])    # Erro 404
print(mensagem[9:15])  # Página
print(mensagem[-11:])  # não encontrada
```

Você também pode inverter uma sequência:

Desvendando Python.py

```
print(mensagem[::-1])
```

Exemplo real: extrair códigos de erro ou formatar textos para logs e relatórios.

Verificando existência com in

O operador **in** verifica se um item está presente dentro de outra estrutura (lista, string, dicionário, etc.).

Desvendando Python.py

```
usuario = "admin"
usuarios_autorizados = ["admin", "root", "dev"]
if usuario in usuarios_autorizados:
    print("Acesso liberado")
```

Também funciona em strings:

Desvendando Python.py

```
comando = "shutdown -r now"
if "shutdown" in comando:
    print("Desligamento solicitado")
```

Exemplo real: validar entrada de usuários, comandos ou palavras-chave.

Filtrando com if (List Comprehension)

A compreensão de listas cria novas listas com base em outra, aplicando filtros e transformações:

Desvendando Python.py

```python
emails = ["joao@gmail.com", "ana@yahoo.com", "paulo@gmail.com"]
gmail = [email for email in emails if "@gmail.com" in email]
print(gmail)
```

Também pode aplicar transformações:

Desvendando Python.py

```python
nomes = ["ana", "pedro", "julia"]
maiusculos = [nome.upper() for nome in nomes]
print(maiusculos)
```

Exemplo real: filtrar e padronizar dados de entrada de usuários.

Acessando dicionários com segurança get()

Dicionários funcionam com chave e valor. A forma padrão de acesso é direta:

Desvendando Python.py

```
aluno = {'nome': 'Carlos', 'curso': 'Python'}
print(aluno['nome'])  # Carlos
```

Mas se você tentar acessar uma chave que não existe, ocorre erro:

Desvendando Python.py

```
print(aluno['email'])  # KeyError
```

Para evitar isso, use **.get()** com valor padrão:

Desvendando Python.py

```
print(aluno.get('email', 'Não informado'))  # Não informado
```

Exemplo real: recuperar campos opcionais em formulários ou APIs.

Percorrendo com enumerate()

enumerate() permite iterar por uma lista e obter o índice de cada item automaticamente:

Desvendando Python.py

```
tarefas = ["Login", "Consulta", "Logout"]
for i, tarefa in enumerate(tarefas, start=1):
    print(f"{i}. {tarefa}")
```

Exemplo real: numerar etapas de processos, criar listas ordenadas para exibição em UI/UX.

Desempacotando valores (Unpacking)

Você pode atribuir múltiplos valores de uma vez:

Desvendando Python.py

```python
x, y = [10, 20]
print(x)  # 10
print(y)  # 20
```

Muito útil ao iterar sobre listas de tuplas:

Desvendando Python.py

```python
usuarios = [("joao", "ativo"), ("ana", "inativo")]
for nome, status in usuarios:
    print(f"{nome} está {status}")
```

Exemplo real: processar registros de banco de dados em pares (nome, status).

Filtrando com filter() e funções lambda

filter() aplica uma função para filtrar elementos com base em uma condição. **lambda** cria uma função anônima (rápida).

Desvendando Python.py

```
valores = [120, 80, 300, 50]
acima_de_100 = list(filter(lambda x: x > 100, valores))
print(acima_de_100)
```

Exemplo real: filtrar produtos, transações ou usuários com base em critérios numéricos.