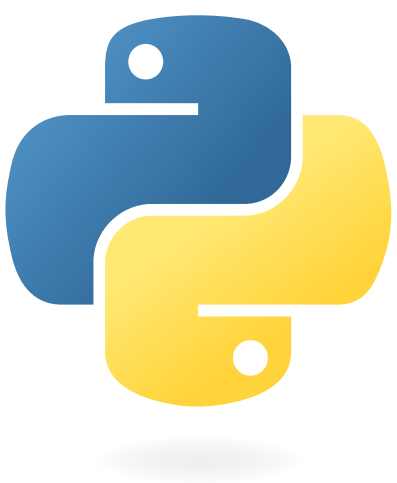
Introdução à Programação em



python™

https://evandro-crr.github.io/intro-python

Evandro Chagas Ribeiro da Rosa

Dicionário

Um dicionário mapeia uma **chave** para um **valor**.

```
>>> contatos = {
...     "Ana"  : "(11) 91234-5678",
...     "Bruno": "(21) 99876-5432",
...     "Clara": "(31) 98765-4321",
... }
>>>
>>> print(contatos["Clara"])
(31) 98765-4321
```

- Um elemento do dicionário é composto por um par chave-valor.
- A chave pode ser de qualquer tipo "hasheável".
- O valor pode ser de qualquer tipo.
- É possível armazenar valores de tipos diferentes no mesmo dicionário.

Adicionando e Removendo do Dicionário

Adicionando Elementos:

```
>>> contatos = {}
>>> contatos["Ana"]   = "(11) 91234-5678"
>>> contatos["Bruno"] = "(21) 99876-5432"
>>> contatos["Clara"] = "(31) 98765-4321"
>>> print(contatos["Clara"])
(31) 98765-4321
```

Removendo Elementos:

```
>>> del contatos["Bruno"]
>>> print(contatos)
{'Ana': '(11) 91234-5678', 'Clara': '(31) 98765-4321'}
```

- A instrução de atribuição adiciona um novo par chave-valor.
- Se a chave já existe, o valor será sobrescrito.
- Para remover uma entrada, usamos a instrução `del`.

Verificando a Existência de uma Chave

```
>>> contatos = {"Ana": "(11) 91234-5678", "Bruno": "(21) 99876-5432"}
>>> "Clara" in contatos
False
>>> "Ana" in contatos
True
```

O operador `in` verifica se uma chave está presente no dicionário.

Evitando Erros ao Acessar Valores

```
>>> contatos = {"Ana": "(11) 91234-5678", "Bruno": "(21) 99876-5432"}
>>> print(contatos.get("Pedro", "não definido"))
não definido
```

- O método `get` é útil para evitar erros quando uma chave não existe no dicionário. Ele retorna um valor padrão se a chave não for encontrada.

Iterando Sobre um Dicionário

Iterando Sobre as Chaves

```
>>> for nome in contatos:
...     valor = contatos[nome]
...     print(f"Contato de {nome}: {valor}")
Contato de Ana: (11) 91234-5678
Contato de Bruno: (21) 99876-5432
```

Iterando Sobre Chave-Valor

```
>>> for chave, valor in contatos.items():
...     print(f"Contato de {chave}: {valor}")
Contato de Ana: (11) 91234-5678
Contato de Bruno: (21) 99876-5432
```

- O método `items()` retorna pares chave-valor.

Dicionários Aninhados

```
>>> contatos = {
...     'Ana': {
...         'telefone': '(11) 91234-5678',
...         'email': 'ana@example.com',
...         'endereço': 'Rua A, 123'
...     },
...     'Bruno': {
...         'telefone': '(21) 99876-5432',
...         'email': 'bruno@example.com',
...         'endereço': 'Avenida B, 456'
...     },
... }
>>> print(contatos['Bruno']['telefone'])
(21) 99876-5432
```

- Dicionários podem conter outros dicionários, o que é útil para armazenar dados estruturados.

⚙ Exercícios 1

Sistema de Registro de Presença

- Implemente um sistema de registro de presença que funcione em três fases:
  1. **Cadastro de alunos**: Inicialmente, o usuário deverá cadastrar os alunos, um a um. O cadastro será concluído quando o usuário decidir parar de adicionar alunos.
  2. **Registro de presença**: Após o cadastro dos alunos, o sistema solicitará o número de dias de aula. Em seguida, para cada dia, o sistema registrará se cada aluno esteve presente ou ausente.
  3. **Relatório final**: No final, o sistema exibirá um relatório com a porcentagem de presença de cada aluno.

⚙ Exercício 2

Análise de Notas dos Alunos

- Implemente um sistema que permita registrar as notas dos alunos em uma disciplina e, ao final, gere um relatório com a média das notas de cada aluno e a média geral da turma:
  1. **Cadastro de alunos**: O usuário deve cadastrar os alunos e a disciplina oferecidas. O cadastro será concluído quando o usuário decidir parar de adicionar alunos.
  2. **Registro de notas**: Após o cadastro dos alunos, o sistema solicitará o número avaliações. Em seguida, o sistema deve solicitar as notas de cada aluno.
  3. **Relatório final**: No final, o sistema exibirá:
     - A nota média de cada aluno.
     - A média geral da turma em cada avaliação.

⚙ Exercícios 3

Sistema de Gerenciamento de Estoque Iterativo

- Implemente um programa interativo de gerenciamento de estoque para uma loja. O programa deve exibir um menu de opções e permitir que o usuário realize as seguintes operações repetidamente, até que ele opte por sair:

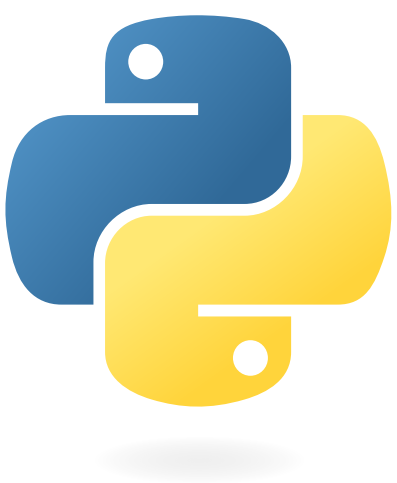
1. Cadastrar produto: O usuário poderá cadastrar um novo produto informando seu **nome**, **quantidade** e **preço unitário**. Se o produto já existir no estoque, a quantidade será **somada** e o preço poderá ser modificado.
2. Remover produto
3. Consultar produto
4. Relatório completo
5. Valor total em estoque
6. Sair

⚙ Exercício 4

Sistema de Biblioteca

- Implemente um sistema de gerenciamento de uma biblioteca, com as seguintes funcionalidades:
  1. **Cadastro de livros**: O usuário deverá cadastrar os livros disponíveis na biblioteca, fornecendo informações como **título**, **autor** e **número de exemplares**.
  2. **Empréstimo de livros**
  3. **Devolução de livros**
  4. **Relatório final**

Introdução à Programação em



python™

https://evandro-crr.github.io/intro-python

Evandro Chagas Ribeiro da Rosa