DICIONARIOS EM PYTHON { }

Guia completo com exemplos, resultados e explicações

Prof. Gustavo Franz (Science/Biology)
Python Developer in Progress
Building Educational Solutions with Python

GitHub: github.com/GustaFranz

O que é um dicionário?

- ✓ Dicionários armazenam dados em pares **chave: valor**.
- ✓ Cada chave deve ser única (e imutável).
- ✓ Os valores podem ser de qualquer tipo.
- ✓ São mutáveis: podemos adicionar, alterar e remover itens.
- ✓ Muito usados para representar dados reais do mundo.

Criando dicionários

Formas de criar um dicionário

```
aluno = {'nome': 'Ana', 'idade': 20, 'curso': 'Python'}  # dados do aluno
vazio = {}  # dicionário vazio
pessoa = dict(nome='Joao', idade=28, cidade='SP')  # usando dict()
copia = aluno.copy()  # cópia rasa do dicionário
```

Operações e análise de dicionários

Exemplos considerando **dados = {'nome': 'Ana', 'idade': 20, 'curso': 'Python'}**.

Comando	Resultado	Explicação
`dados['nome']`	`'Ana'`	Acessa o valor da chave 'nome'.
`dados['idade']`	`20`	Acessa o valor da chave 'idade'.
`dados.get('curso')`	`'Python'`	Retorna o valor da chave. Evita erro se ela não existir.
`dados.get('telefone')`	`None`	Como 'telefone' não existe, retorna None.
`dados.get('tel', 'N/I')`	`'N/I'`	Retorna o valor padrão quando a chave não existe.
`dados['idade'] = 21`	`(altera)`	Altera o valor de uma chave existente.
`dados['email'] = 'a@x.com'`	`(adiciona)`	Adiciona um novo par chave:valor.
`del dados['curso']`	`(remove)`	Remove o par da chave informada.
`dados.pop('idade')`	`20`	Remove o par e retorna o valor removido.

Comando	Resultado	Explicação
`dados.pop('tel', None)`	`None`	Remove a chave se existir; senão retorna o padrão.
`'nome' in dados`	`True`	Verifica se a chave existe no dicionário.
`'tel' in dados`	`False`	A chave 'tel' não existe no dicionário.
`dados.keys()`	`dict_keys([...])`	Retorna todas as chaves do dicionário.
`dados.values()`	`dict_values([...])`	Retorna todos os valores do dicionário.
`dados.items()`	`dict_items([...])`	Retorna pares (chave, valor) como tuplas.
`len(dados)`	`3`	Retorna a quantidade de pares chave:valor.
`dados.clear()`	`{}`	Remove todos os itens do dicionário.
`dados.copy()`	`{...}`	Cria uma cópia rasa do dicionário.
`dados.update({'uf': 'RJ'})`	`{...}`	Atualiza ou adiciona pares de outro dicionário.
`dados.setdefault('pais', 'BR')`	`'BR'`	Adiciona a chave com padrão se não existir.

Percorrendo dicionários

Três formas de iterar

```
# apenas as chaves
for chave in dados:
    print(chave)

# apenas os valores
for valor in dados.values():
    print(valor)

# chaves e valores ao mesmo tempo
for chave, valor in dados.items():
    print(chave, '->', valor)
```

Dicionários aninhados

Dicionário dentro de dicionário

```
aluno = {
    'nome': 'Ana',
    'contato': {
        'email': 'ana@email.com',
        'telefone': '11 99999-9999'
    },
    'notas': [8.5, 9.0, 7.0]
}
print(aluno['contato']['email'])  # ana@email.com
print(aluno['notas'][0])  # 8.5
```

Métodos úteis (resumo)

Método	O que faz
`keys()`	Retorna todas as chaves.
`values()`	Retorna todos os valores.
`items()`	Retorna pares (chave, valor).
`get()`	Acessa um valor com segurança.
`update()`	Atualiza ou adiciona pares.
`pop()`	Remove e retorna o valor.
`popitem()`	Remove o último par inserido.
`clear()`	Remove todos os itens.
`copy()`	Cria uma cópia rasa.

Exercício: sistema de biblioteca

Um sistema que gerencia livros, onde **cada livro é um dicionário**. Permite cadastrar, listar, buscar, emprestar, devolver e remover. Cada livro guarda: título, autor, ano, gênero e disponível (True/False).

biblioteca.py

```
biblioteca = []  # cada livro é um dicionário

def cadastrar():
    livro = {
        'titulo': input('Título: '),
        'autor': input('Autor: '),
        'ano': input('Ano: '),
        'genero': input('Gênero: '),
        'disponivel': True,
    }
    biblioteca.append(livro)
    print('Livro cadastrado!')

def listar():
    if not biblioteca:
        print('Nenhum livro cadastrado.')
        return
    for livro in biblioteca:
        estado = 'Disponível' if livro['disponivel'] else 'Emprestado'
        print(livro['titulo'], '-', livro['autor'], '(', estado, ')')

def buscar():
    termo = input('Título buscado: ').lower()
    for livro in biblioteca:
        if termo in livro['titulo'].lower():
            print(livro)

def emprestar():
    termo = input('Título: ').lower()
    for livro in biblioteca:
        if livro['titulo'].lower() == termo and livro['disponivel']:
            livro['disponivel'] = False
            print('Empréstimo realizado!')
            return
    print('Livro indisponível.')

# (devolver e remover seguem a mesma lógica, mudando o estado/lista)

while True:
    print('1-Cadastrar 2-Listar 3-Buscar 4-Emprestar 7-Sair')
    opcao = input('Escolha: ')
    if opcao == '1': cadastrar()
    elif opcao == '2': listar()
```

```
    elif opcao == '3': buscar()
    elif opcao == '4': emprestar()
    elif opcao == '7': break
```

O que praticamos aqui

- ✓ Usamos dicionários para representar dados de forma estruturada.
- ✓ Cada livro é identificado por suas chaves (título, autor, ano...).
- ✓ Exploramos inclusão, busca, atualização e remoção de itens.
- ✓ Combinamos dicionários com listas, booleanos e funções.

Dica final + boas práticas

- ✓ Dicionários modelam dados do mundo real com clareza.
- ✓ Use chaves significativas e consistentes; evite duplicadas.
- ✓ Prefira .get() para acessos seguros (sem erro).
- ✓ Mantenha o código organizado em funções pequenas.

Prof. Gustavo Franz (Science/Biology)
Python Developer in Progress
Building Educational Solutions with Python

Professor de Ciencias e Biologia desde 2013 — atualmente estudando Programacao.

Eu crio estes resumos para organizar os assuntos e estudar de forma clara e objetiva. Estou compartilhando porque eles tambem podem ajudar outros desenvolvedores que estao comecando. Bons estudos — vamos aprender juntos!

GitHub: github.com/GustaFranz

Exercicios Python: github.com/GustaFranz/python_exercises
Para quem quiser ver a resolucao dos meus exercicios em Python.

EasyAnsi: github.com/GustaFranz/easyansi
Para quem quiser codificar personalizando com cores e estilos de forma mais facil, pratica e intuitiva.