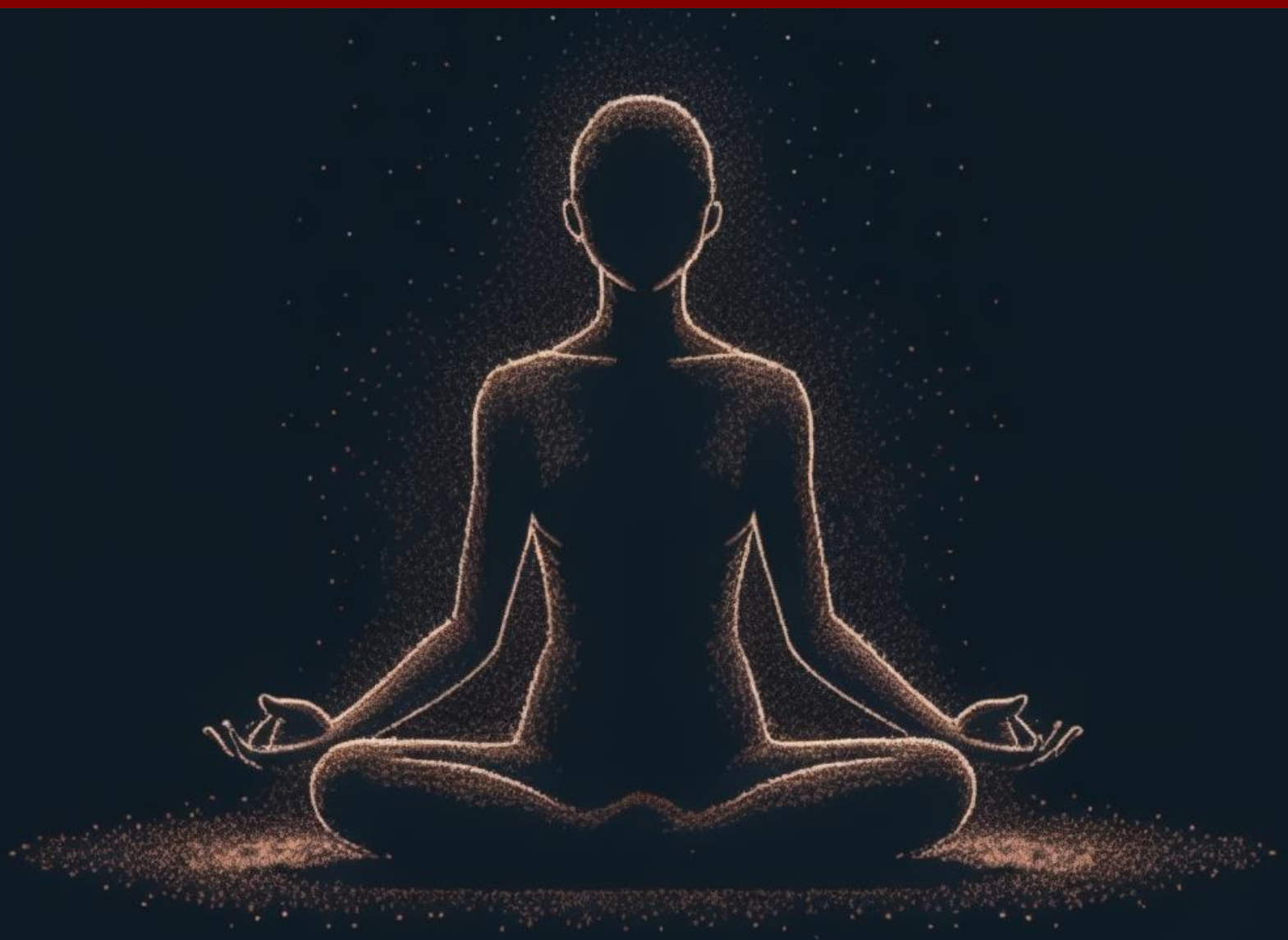
COLEÇÕES NO MUNDO INVERTIDO

EXPLORANDO DICIONÁRIOS EM PYTHON



python™

Aprenda a importância e as vantagens de uma das principais estruturas de dados do python

PEDRO AUGUSTO

INTRODUÇÃO AOS DICIONÁRIOS EM PYTHON

Aprendendo o Básico a Respeito

Os dicionários são uma das estruturas de dados mais importantes e versáteis em Python. Eles permitem armazenar pares de chave (key) e valor (value), oferecendo uma maneira eficiente de organizar e acessar dados. Neste ebook, exploraremos as principais funcionalidades e usos de dicionários com exemplos práticos para que você possa aproveitar ao máximo essa poderosa ferramenta em seus projetos.

01

O QUE SÃO DICIONÁRIOS

Um dicionário em Python é uma coleção não ordenada, mutável e indexada de pares de chave e valor, oferecendo rapidez, eficiência e flexibilidade. Aprenda suas vantagens e seus diferenciais.

FUNDAMENTAÇÃO TEÓRICA

Dicionários são uma ferramenta poderosa em Python, e compreender como usá-los de modo efetivo pode melhorar significativamente a qualidade e desempenho do seu código. Eles são definidos usando chaves ‘ {} ’ e cada par chave-valor é separado por dois pontos ‘ : ’. Lembrando que eles não aceitam chaves duplicadas. Veja um exemplo básico abaixo, que usaremos como base para várias aplicações ao longo deste ebook:

dicionarios.py

```python
# Exemplo básico de um dicionário
aluno = {
    "nome": "João",
    "idade": 20,
    "curso": "Engenharia"
}
print(aluno["nome"])  # Saída: João
```

VANTAGENS DOS DICIONÁRIOS

- Acesso Rápido: Encontrar o valor associado a uma chave é muito rápido.
- Flexibilidade: Podem armazenar qualquer tipo de dado.
- Legibilidade: O uso de chaves descritivas torna o código mais claro.

Por oferecerem acesso rápido aos dados usando chaves únicas, dicionários se tornam mais eficientes do que listas quando é necessário fazer buscas frequentes. Veja abaixo:

dicionarios.py

```python
# Usando uma lista
usuarios = [("Ana", 25), ("Pedro", 30)]
for usuario in usuarios:
    if usuario[0] == "Pedro":
        print(usuario[1])  # Saída: 30

# Usando um dicionário
usuarios_dict = {"Ana": 25, "Pedro": 30}
print(usuarios_dict["Pedro"])  # Saída: 30
```

02

MANIPULAÇÃO DE DICIONÁRIOS

Aprenda como adicionar novos pares chave-valor, atualizar os existentes, ou removê-los.

ADICIONANDO E ATUALIZANDO ITENS

Para adicionar novos pares chave-valor, é preciso chamar o nome do dicionário, seguido da chave, entre colchetes ‘[]‘, correspondente ao valor que se deseja adicionar e, após isso, atribuir esta chave ao novo valor por meio do sinal de atribuição ‘ = ‘.

Já para atualizar algum valor, basta fazer o mesmo; porém, utilizando o nome da chave que se deseja atualizar no interior dos colchetes. Levando em conta o dicionário ‘aluno’ mostrado na página 4, demonstra-se como manipular chave-valor no exemplo abaixo:

dicionarios.py

```python
aluno["nota"] = 8.5  # Adicionando uma nova chave
aluno["idade"] = 21  # Atualizando uma chave existente
print(aluno)
```

REMOVENDO ITENS

É possível remover itens de um dicionário por meio do método pop, ou da palavra-chave del, conforme mostrado a seguir:

dicionarios.py

```
dicionario = {"nome": "Carlos", "idade": 22}
idade = dicionario.pop("idade")
print(idade)  # Saída: 22
print(dicionario)  # Saída: {'nome': 'Carlos'}

del dicionario["nome"]
print(dicionario)  # Saída: {}
```

03

ITERAÇÕES E MÉTODOS ÚTEIS

Os métodos dos dicionários facilitam a obtenção de listas de chaves, valores e itens, tornando a manipulação de dados mais intuitiva.

ITERANDO DICIONÁRIOS

Por meio da iteração, percorremos um dicionário para acessar suas chaves e valores. Por exemplo:

dicionarios.py

```python
for chave in aluno:
    print(chave, aluno[chave])

for chave, valor in aluno.items():
    print(chave, valor)
```

dicionarios.py

```python
dicionario = {"nome": "Carlos", "idade": 22}

for chave, valor in dicionario.items():
    print(f"Chave: {chave}, Valor: {valor}")
# Saída:
# Chave: nome, Valor: Carlos
# Chave: idade, Valor: 22
```

MAIS SOBRE MÉTODOS ÚTEIS

Podemos iterar chaves, valores ou ambos usando métodos como ‘items()’, ‘keys()’ e ‘values()’. Lembrando que o método keys() retorna apenas as chaves; o values(), apenas valores; e o items() retorna ambos.

Explore alguns exemplos práticos de tais métodos e suas respectivas saídas:

dicionarios.py

```python
print(aluno.keys())    # Saída: dict_keys(['nome', 'idade'])
print(aluno.values())  # Saída: dict_values(['João', 21])
print(aluno.items())   # Saída: dict_items([('nome', 'João'), ('idade', 21)])
```

04

APLICAÇÕES PRÁTICAS

Dicionários são extremamente úteis para aplicações práticas como contar ocorrências e agrupar dados, proporcionando soluções eficientes para problemas do dia a dia.

CONTANDO OCORRÊNCIAS

Use dicionários para contar a frequência de elementos em uma lista. No exemplo abaixo, criamos um dicionário nomeado ‘frutas’, que possui uma série de elementos, sendo alguns deles repetidos.

Por meio de uma iteração, contamos a frequência com que cada um deles se repete. Confira:

dicionarios.py

```python
frutas = ["maçã", "banana", "laranja", "maçã", "banana", "maçã"]
contagem = {}
for fruta in frutas:
    if fruta in contagem:
        contagem[fruta] += 1
    else:
        contagem[fruta] = 1
print(contagem)  # Saída: {'maçã': 3, 'banana': 2, 'laranja': 1}
```

AGRUPAMENTO DE DADOS

Dicionários também podem agrupar dados relacionados. A lista ‘estudantes’ contém três dicionários, cada um representando um estudante com seu nome e curso. ‘Grupos’ é um dicionário vazio que será usado para agrupar os nomes dos estudantes pelos cursos. O loop ‘for’ percorre cada dicionário dentro da lista estudantes.

A parcela ‘grupos[curso].append(estudante["nome"])’ serve para adicionar o nome do estudante à lista de nomes correspondente ao curso.

dicionarios.py

```python
estudantes = [
    {"nome": "Ana", "curso": "Matemática"},
    {"nome": "Pedro", "curso": "História"},
    {"nome": "Maria", "curso": "Matemática"},
]

grupos = {}

for estudante in estudantes:
    curso = estudante["curso"]
    if curso not in grupos:
        grupos[curso] = []
    grupos[curso].append(estudante["nome"])

print(grupos)
# Saída: {'Matemática': ['Ana', 'Maria'], 'História': ['Pedro']}
```

05

DICIONÁRIOS ANINHADOS

Dicionários aninhados permitem a organização de dados complexos em estruturas hierárquicas, facilitando o acesso e manipulação.

DICIONÁRIOS ANINHADOS

Dicionários aninhados são poderosos para representar dados complexos e estruturados de forma organizada e acessível. Eles são amplamente utilizados em sistemas de gerenciamento, catálogos de produtos e muitos outros cenários que exigem uma estrutura hierárquica de informações.

Exemplo 1 – Sistema de Gerenciamento Escolar: Propósito de armazenar informações sobre escolas, incluindo dados de turmas e de alunos, como na estrutura abaixo. Temos um dicionário para agrupar todas as escolas, composto por um outro para cada escola, contendo mais outro para cada turma, e assim por diante. Todos eles estão aninhados dentro do dicionário geral escolas.

dicionarios.py

```
escolas = {
    "Escola A": {
        "Turma 1": {"Aluno1": {"idade": 10, "nota": 8.5}, "Aluno2": {"idade": 11, "nota": 7.8}},
        "Turma 2": {"Aluno3": {"idade": 10, "nota": 9.0}, "Aluno4": {"idade": 12, "nota": 6.5}},
    },
    "Escola B": {
        "Turma 1": {"Aluno5": {"idade": 11, "nota": 7.5}, "Aluno6": {"idade": 12, "nota": 8.3}},
        "Turma 3": {"Aluno7": {"idade": 10, "nota": 8.8}, "Aluno8": {"idade": 11, "nota": 7.1}},
    },
}
```

DICIONÁRIOS ANINHADOS

Exemplo 2 – Catálogo de Produtos de uma Loja Online: Tem como propósito organizar produtos por categorias e subcategorias, contendo informações detalhadas de cada produto.

dicionarios.py

```python
produtos = {
    "Eletrônicos": {
        "Smartphones": {
            "iPhone 12": {"preço": 799, "estoque": 50},
            "Samsung Galaxy S21": {"preço": 699, "estoque": 30},
        },
        "Laptops": {
            "MacBook Pro": {"preço": 1299, "estoque": 20},
            "Dell XPS 13": {"preço": 999, "estoque": 15},
        },
    },
    "Eletrodomésticos": {
        "Geladeiras": {
            "LG Frost Free": {"preço": 1200, "estoque": 10},
            "Samsung Duplex": {"preço": 1500, "estoque": 8},
        },
        "Máquinas de Lavar": {
            "Brastemp 12kg": {"preço": 850, "estoque": 5},
            "Consul 10kg": {"preço": 700, "estoque": 7},
        },
    },
}
```

CONCLUSÃO E AGRADECIMENTOS

OBRIGADO PELA LEITURA!

Espero de coração que os conteúdos deste Ebook tenham te ajudado ao longo de sua caminhada no aprendizado de Python. Todo este material foi gerado com muito carinho para que desenvolvedores iniciantes e juniores que, assim como eu, são apaixonados por Python, tenham um manual de acesso a este assunto essencial para tratamento de dados – Dicionários.

Aprender esta ferramenta é imprescindível para aqueles que desejam avançar em desenvolvimento com essa linguagem. Portanto, fiz questão de gerar um material que ilustrasse bem sua importância e aplicação no dia a dia da programação, e que trouxesse funcionalidades interessantes – para que o conteúdo servisse até mesmo como fonte de consulta em caso de futuras dúvidas.

Material produzido com ajuda de uma Inteligência Artificial e validado por mim, de modo que os conceitos abordados trouxessem uma visão real de como utilizar dicionários de forma clara e prática. Esta é uma versão prévia para fins didáticos, a qual pretendo atualizar em breve para uma versão completa, abordando ainda mais funcionalidades.

https://github.com/pedroaugust096/Creating-an-ebook-using-AI