TUPLAS E DICIONÁRIOS

Ruan Pablo Medeiros

TUPLAS – O QUE SÃO?

- Em Python, tuplas são estruturas de dados ordenadas e imutáveis. Isso significa que, uma vez criadas, as tuplas não podem ser alteradas. Elas são definidas entre parênteses () e podem conter elementos de diferentes tipos, como números, strings e até outras tuplas.

TUPLAS – O QUE SÃO?

Imagine uma tupla como uma caixa de bombons:

- Cada bombom na caixa representa um elemento da tupla.
- Cada bombom tem um sabor único, assim como cada elemento da tupla pode ter um tipo diferente.
- Uma vez que a caixa de bombons está fechada, você não pode adicionar mais bombons ou tirar os que já estão lá. Da mesma forma, uma tupla imutável, você não pode modificar seus elementos após sua criação.

TUPLAS - POR QUE USAR TUPLAS?

As tuplas são úteis em diversas situações, como:

- **Armazenar dados que não precisam ser modificados:** Por exemplo, informações de contato como nome, telefone e endereço.
- **Proteger dados:** Como as tuplas são imutáveis, elas garantem que os dados armazenados nelas não sejam acidentalmente alterados.
- **Retornar vários valores de uma função:** Uma função pode retornar uma tupla contendo vários valores.
- **Desempacotar valores:** Tuplas podem ser desempacotadas em variáveis, facilitando a atribuição múltipla.

TUPLAS – CRIANDO TUPLAS

- Criar uma tupla é simples: basta listar os elementos entre parênteses, separados por vírgulas.

```
1 # Tupla com elementos de diferentes tipos
2 minha_tupla = (1, "Olá", 3.14)
3
4 # Tupla vazia
5 tupla_vazia = ()
6
```

TUPLAS – ACESSANDO ELEMENTOS

- Para acessar um elemento específico de uma tupla, utilize o índice do elemento entre colchetes. O índice começa em 0, ou seja, o primeiro elemento da tupla tem índice 0, o segundo elemento tem índice 1 e assim por diante.

```
# Acessando o primeiro elemento da tupla
primeiro_elemento = minha_tupla[0]  # primeiro_elemento = 1

# Acessando o último elemento da tupla
ultimo_elemento = minha_tupla[-1]  # ultimo_elemento = 3.14

```

TUPLAS – FATIAMENTO(SLICE)

- O fatiamento permite obter subtuplas a partir de uma tupla original. Utilize os índices entre colchetes, separados por dois pontos (:).

```
# Subtupla contendo os dois primeiros elementos
subtupla1 = minha_tupla[0:2]  # subtupla1 = (1, "Olá")

# Subtupla contendo o elemento do índice 2 até o final da tupla
subtupla2 = minha_tupla[2:]  # subtupla2 = (3.14,)

```

TUPLAS - OPERAÇÕES

- **Tamanho:** A função len() retorna o tamanho da tupla, ou seja, o número de elementos que ela contém.

```
1 tamanho = len(minha_tupla)  # tamanho = 3
2
```

TUPLAS - OPERAÇÕES

- **Verificação de Pertinência:** Utilize o operador in para verificar se um elemento está presente na tupla.

```
elemento_presente = "Olá" in minha_tupla  # elemento_presente = True
```

TUPLAS - OPERAÇÕES

- **Concatenação:** Utilize o operador + para concatenar tuplas.

```
concatenacao = minha_tupla + minha_tupla  # concatenacao = (1, 'Olá', 3.14, 1, 'Olá', 3.14)
```

TUPLAS - OPERAÇÕES

- **Repetição:** O operador * pode ser usado para repetir uma tupla um determinado número de vezes.

```
tupla_repetida = minha_tupla * 3  # tupla_repetida = (1, "Olá", 3.14) * 3 = (1, "Olá", 3.14, 1, "Olá", 3.14, 1, "Olá", 3.14)
```

TUPLAS – TUPLAS NOMEADAS

- As tuplas nomeadas, também conhecidas como namedtuple, são uma forma especial de tupla que permite atribuir nomes aos seus elementos. Isso facilita a leitura e o uso da tupla, especialmente quando ela contém vários elementos.

TUPLAS – TUPLAS NOMEADAS

```python
from collections import namedtuple

# Criando uma tupla nomeada
Ponto2D = namedtuple('Ponto2D', ['x', 'y'])

# Criando um ponto 2D com tupla nomeada
ponto = Ponto2D(10, 20)

# Acessando elementos por nome
coordenada_x = ponto.x  # coordenada_x = 10
coordenada_y = ponto.y  # coordenada_y = 20

```

DICIONÁRIOS - O QUE SÃO?

- **Um dicionário em Python é uma coleção ordenada de pares chave-valor.** Imagine um dicionário real, onde cada palavra (chave) tem uma definição (valor) associada. No Python, funciona da mesma forma.

DICIONÁRIOS - CARACTERÍSTICAS

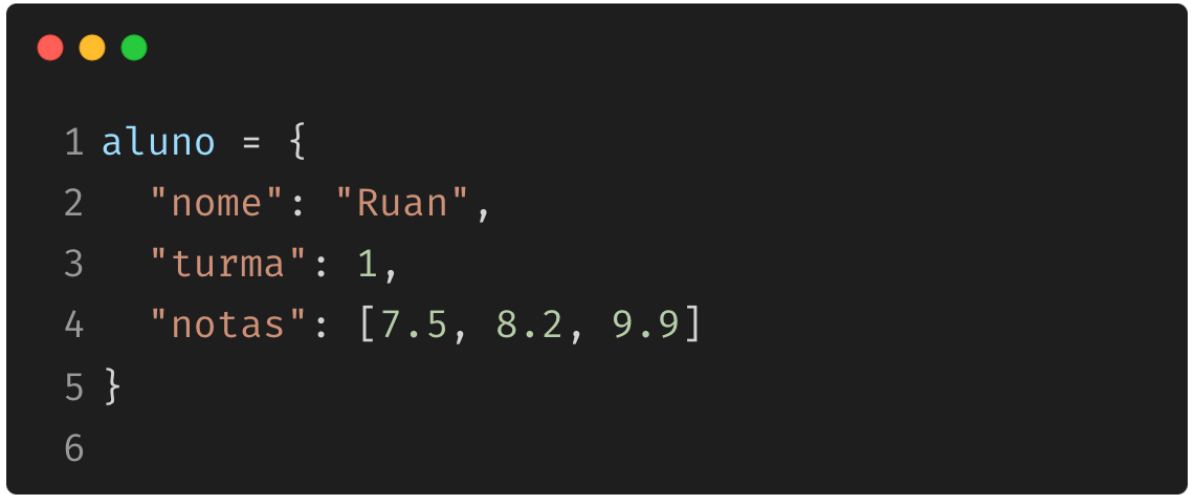
- **Ordenado:** As chaves são armazenadas em uma ordem específica, mas não a ordem de inserção.
- **Mutável:** Os valores podem ser modificados após a criação do dicionário.
- **Heterogêneo:** As chaves e os valores podem ser de qualquer tipo de dado em Python (números, strings, listas, tuplas, etc.).

DICIONÁRIOS - CRIANDO UM DICIONÁRIO

Existem duas maneiras principais de criar um dicionário:

1. Usando chaves literais:

- Dentro das chaves {}, cada par é escrito como "chave:valor".
- Diversos tipos de dados podem ser usados como chaves e valores.



```
aluno = {
  "nome": "Ruan",
  "turma": 1,
  "notas": [7.5, 8.2, 9.9]
}
```

DICIONÁRIOS - CRIANDO UM DICIONÁRIO

Existem duas maneiras principais de criar um dicionário:

1. ~~Usando chaves literais:~~
2. Usando a função dict():
   - A função dict() recebe pares chave-valor como argumentos.
   - Mais flexibilidade para criar dicionários a partir de outras estruturas de dados.

```
1 aluno = dict(nome="Ruan", turma=1, notas=[7.5,
  8.2, 9.9])
2
```

DICIONÁRIOS - ACESSANDO ELEMENTOS

Para acessar um valor específico no dicionário, basta usar a chave correspondente entre colchetes:

```
valor = aluno["nome"] # Obtém o valor "Ruan" associado à chave "nome"
```

Lembre-se: as chaves devem ser escritas exatamente como no momento da criação.

DICIONÁRIOS - ALTERANDO O VALOR DE UMA CHAVE EXISTENTE

Você pode alterar o valor de uma chave existente da seguinte forma:

```
1 # dicionario["chave_existente"] = novo_valor
2 aluno["nome"] = "José"
3
```

DICIONÁRIOS - ADICIONANDO NOVO PAR CHAVE VALOR

Você pode adicionar um novo item "chave:valor" da seguinte forma:

```
# dicionario["nova_chave"] = valor
aluno["idade"] = 18

```

DICIONÁRIOS - DELETANDO UM ITEM "CHAVE:VALOR"

Você pode deletar um item "chave:valor" da seguinte forma:

```
# del dicionario["chave_que_deseja_deletar"]
del aluno["idade"]

```

DICIONÁRIOS - PERCORRENDO O DICIONÁRIO

Duas maneiras principais de percorrer um dicionário e seus pares chave-valor:

1. Usando um loop for em chaves:

- Imprime cada chave e seu valor correspondente.

```
for chave in aluno:
    valor = aluno[chave]
    print(f"Chave: {chave}, Valor: {valor}")

```

DICIONÁRIOS - PERCORRENDO O DICIONÁRIO

Duas maneiras principais de percorrer um dicionário e seus pares chave-valor:

1. ~~Usando um loop for em chaves:~~
2. Usando um loop for em itens (pares chave-valor):
   - Similar ao loop anterior, mas usando o método items() para obter os pares diretamente.

```python
for item in aluno.items():
    chave, valor = item
    print(f"Chave: {chave}, Valor: {valor}")

```

DICIONÁRIOS - FUNÇÕES ÚTEIS

- len(dicionario): Retorna o número de pares chave-valor no dicionário.
- dicionario.keys(): Retorna uma lista com todas as chaves do dicionário.
- dicionario.values(): Retorna uma lista com todos os valores do dicionário.
- dicionario.items(): Retorna uma lista de tuplas contendo cada par chave-valor.
- chave in dicionario: Verifica se uma chave existe no dicionário.
- valor in dicionario.values(): Verifica se uma chave existe no dicionário.
- dicionario.get(chave, valor_padrao): Obtém o valor associado à chave especificada. Se a chave não existir, retorna o valor_padrao.
- dicionario.update(outro_dicionario): Atualiza o dicionário com pares chave-valor de outro dicionário.
- dicionario.copy() : Cria uma cópia do dicionário original.

EXERCÍCIOS

Exercício 1: Coisas Favoritas:

- Crie um dicionário chamado coisas_favoritas para armazenar informações sobre suas coisas favoritas em diferentes categorias (comida, cor, filme, etc.). Cada chave deve ser o nome da categoria (string) e o valor deve ser o nome do item correspondente (string).
- Imprima o dicionário inteiro usando print(coisas_favoritas).
- Acesse e imprima o valor associado a uma chave específica (por exemplo, "comida") usando print(coisas_favoritas["comida"]).
- Adicione um novo par chave-valor para outra coisa favorita (por exemplo, "livro").
- Atualize o valor de uma chave existente (por exemplo, altere o filme favorito).
- Remova um par chave-valor usando a palavra-chave del (por exemplo, remova "cor").

EXERCÍCIOS

Exercício 2: Classificações de Filmes

- Crie um dicionário chamado classificacoes_filmes para armazenar títulos de filmes (strings) como chaves e suas classificações correspondentes (inteiros entre 1 e 5) como valores.
- Peça ao usuário que digite o título de um filme e use um loop while para continuar perguntando até que o usuário digite "sair".
- Para cada título digitado, verifique se ele está no dicionário usando o operador in:

  Se sim, imprima a classificação do filme.

  Se não, peça ao usuário uma classificação (1-5) e adicione o título do filme e a classificação ao dicionário.

EXERCÍCIOS

Exercício 3: Contador de Frequência de Palavras

- Escreva um programa que lê uma string de texto como entrada e mostra ao final do programa um dicionário onde cada chave é uma palavra única no texto (convertida para minúsculas) e o valor é o número de vezes que essa palavra aparece.
- Use um loop for para iterar sobre as palavras no texto (divididas por espaços em branco), um método dict.get() para verificar se uma palavra já está no dicionário e atualizar a contagem de acordo.