

MAC2166 - Introdução à Computação (1º Semestre de 2025)

Grande áreas Civil, Mecânica, Química e Petróleo

Guia de Estudos - Semana 13

Tópicos

- criação de dicionários do Python
- forma de acesso aos valores e chaves de um dicionário
- restrições das chaves de um dicionário
- exemplos de dicionários tendo strings como chaves
- exemplos de dicionários tendo tuplas como chaves

Dicionários do Python

Dicionários correspondem a um novo tipo de dado da classe `dict`. Um dicionário é uma sequência de pares de elementos: chave e valor. Dicionários são usados para o mapeamento eficiente de uma chave ao seu valor correspondente associado.

A estrutura de dados interna de um dicionário é dada por uma "tabela de espalhamento" (hash table), cuja explicação detalhada é ensinada em disciplinas mais avançadas de computação. Aqui estamos apenas mostrando as aplicações de uso e sintaxe dos dicionários do Python.

Criação de dicionários do Python

Dicionários são delimitados por chaves `{}` e possuem uma lista de pares **chave:valor** separados por vírgulas.

Para criar um dicionário vazio usamos simplesmente `{}`

Exemplo:

```
>>> D = {}
>>> type(D)
<class 'dict'>
```

Um dicionário pode ser criado gradualmente, começando de um dicionário vazio, adicionando, a cada passo, um novo par: chave e valor. **Exemplo:**

```
>>> D = {}
>>> D["mamão"] = 30
>>> D["abacaxi"] = 50
>>> D["melancia"] = 15
>>> print(D)
{'mamão': 30, 'abacaxi': 50, 'melancia': 15}
```

No exemplo acima, o dicionário representa o inventário de estoque de um mercado de frutas, armazenando para cada fruta sua quantidade no estoque. O mesmo dicionário pode ser também criado alternativamente de modo direto, já inicializado com um conjunto de chaves e valores iniciais.

Exemplo:

```
>>> D = {"mamão": 30, "abacaxi": 50, "melancia": 15}
>>> print(D)
{'mamão': 30, 'abacaxi': 50, 'melancia': 15}
```

Acesso aos valores e chaves de um dicionário

Os valores de um dicionário podem ser acessados usando a chave correspondente como índice, usando a sintaxe: `nome_dicionário[chave]`

Exemplos:

```
>>> print( D["mamão"] )
30
>>> print( D["abacaxi"] )
50
>>> print( D["melancia"] )
15
```

A seguinte construção com o laço `for` pode ser usada para iterar por todas as chaves do dicionário.

```
1 | for chave in D.keys():
2 |     print(chave)
```

Como saída do programa temos:

```
mamão
abacaxi
melancia
```

Alternativamente, para obter a mesma saída, podemos usar simplesmente:

```
1 | for chave in D:
2 |     print(chave)
```

Para calcular o total de frutas no estoque, podemos usar o seguinte laço:

```
1 | total = 0
2 | for valor in D.values():
3 |     total += valor
4 | print(total)
```

Como saída do programa temos o total 95.

Alternativamente, para obter a mesma saída, podemos usar também:

```
1 total = 0
2 for chave in D:
3     total += D[chave]
4 print(total)
```

Para testar se uma chave pertence a um dicionário, podemos usar o comando: `chave in nome_dicionário.keys()`

Exemplo:

```
>>> D = {"mamão": 30, "abacaxi": 50, "melancia": 15}
>>> "abacaxi" in D.keys()
True
>>> "banana" in D.keys()
False
```

Alternativamente, para obter a mesma saída, podemos usar simplesmente: `chave in nome_dicionário`

Exemplo:

```
>>> D = {"mamão": 30, "abacaxi": 50, "melancia": 15}
>>> "abacaxi" in D
True
>>> "banana" in D
False
```

Restrições das chaves de um dicionário

O dicionário do Python apenas aceita como chaves tipos de dados imutáveis (ex: strings, tuplas), para garantir que a consistência da estrutura de dados do dicionário não será quebrada por edições nas chaves. Por exemplo, listas do Python do tipo `list` não podem ser usadas como chave de dicionário.

Exemplo:

```
>>> D = { [1,2]: "azul", [0,3]: "vermelha" }
TypeError: unhashable type: 'list'
```

Exemplos de dicionários tendo strings como chaves

Problema 1:

Escreva um programa que leia uma linha de texto e conte o número de ocorrências de cada letra presente no texto.

Solução 1:

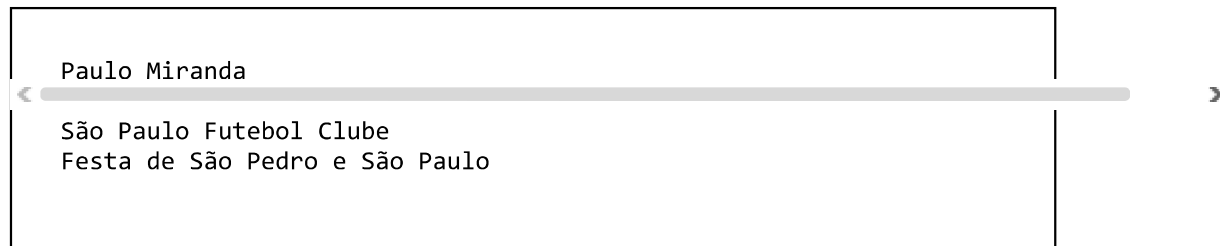
Na solução abaixo, um dicionário é criado, tal que as letras do texto são usadas como chaves e seus valores correspondentes do dicionário são usados para contar o número de ocorrências.

```
1 def main():
2     ignorados = " ,. ; ! \n"
3     texto = input("Digite um texto: ")
4     D = {}
5     for c in texto:
6         if c not in ignorados:
7             if c in D:
8                 D[c] += 1
9             else: #c not in D
```

```
10         D[c] = 1
11     for chave in D:
12         if D[chave] == 1:
13             print("letra '%s' ocorre %d vez"%(chave, D[chave]))
14         else:
15             print("letra '%s' ocorre %d vezes"%(chave, D[chave]))
16
```

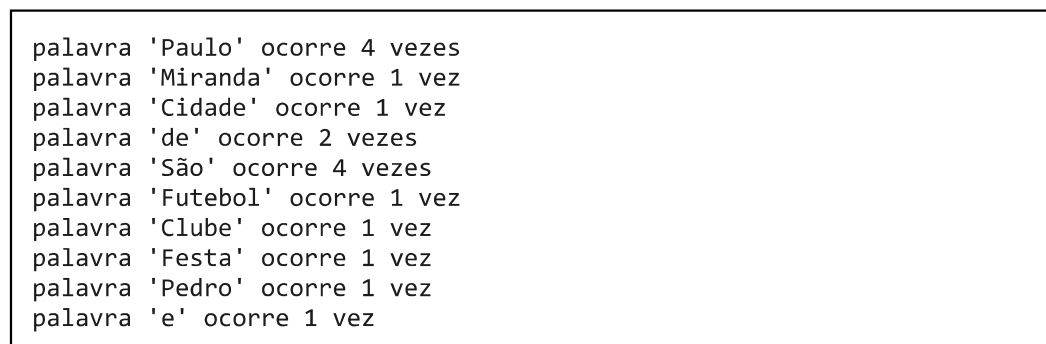
Problema 2:

Escreva um programa que conte o número de ocorrências de cada palavra presente em um arquivo texto de nome **"texto.txt"**. Por exemplo, considere o conteúdo abaixo para o arquivo.



Paulo Miranda
São Paulo Futebol Clube
Festa de São Pedro e São Paulo

Neste caso, a saída esperada será:



```
palavra 'Paulo' ocorre 4 vezes
palavra 'Miranda' ocorre 1 vez
palavra 'Cidade' ocorre 1 vez
palavra 'de' ocorre 2 vezes
palavra 'São' ocorre 4 vezes
palavra 'Futebol' ocorre 1 vez
palavra 'Clube' ocorre 1 vez
palavra 'Festa' ocorre 1 vez
palavra 'Pedro' ocorre 1 vez
palavra 'e' ocorre 1 vez
```

Solução 1:

Na solução abaixo, um dicionário é criado, tal que as palavras presentes no arquivo texto são usadas como chaves e seus valores correspondentes do dicionário são usados para contar o número de ocorrências.

```
1 def main():
2     arquivo = open("texto.txt", "r")
3     D = {}
4     for linha in arquivo:
5         palavras = linha.split()
6         for palavra in palavras:
7             if palavra in D.keys():
8                 D[palavra] += 1
9             else:
10                D[palavra] = 1
11     arquivo.close()
12     for chave in D:
13         if D[chave] == 1:
14             print("palavra '%s' ocorre %d vez"%(chave, D[chave]))
15         else:
16             print("palavra '%s' ocorre %d vezes"%(chave, D[chave]))
17
18 main()
```

Exemplos de dicionários tendo tuplas como chaves

Tuplas são uma sequência de elementos separados por vírgulas, delimitados ou não por parênteses.

Exemplo:

```
>>> T = (2,3,5,7,9)
>>> type(T)
<class 'tuple'>
```

Alternativamente, podemos usar simplesmente:

```
>>> T = 2,3,5,7,9
>>> type(T)
<class 'tuple'>
```

Tuplas são similares a listas, permitindo o acesso aos seus elementos a partir de seus índices (iniciando em zero), usando a sintaxe: `nome_tupla[índice]`

Exemplo:

```
>>> T = (2,3,5)
>>> print( T[0] )
2
>>> print( T[1] )
3
>>> print( T[2] )
5
```

Além da diferença de sintaxe, as tuplas se diferem das listas por serem um tipo de dado imutável, ou seja, um tipo de dado cujo valor não pode ser alterado após sua criação. Portanto, os valores dos elementos de uma tupla criada não podem ser alterados por comandos de atribuição e novos elementos não podem ser adicionados a uma tupla existente via método `append`.

Exemplo:

```
>>> T = (2,3,5)
>>> T[0] = 8
TypeError: 'tuple' object does not support item assignment
>>> T.append(7)
AttributeError: 'tuple' object has no attribute 'append'
```

Por serem imutáveis, as tuplas garantem que os dados permaneçam consistentes e não sejam inadvertidamente modificados. Portanto, tuplas podem ser usadas como chaves de um dicionário, garantindo a integridade de sua estrutura de dados.

Problema 3:

Considere um jogo de Batalha naval simplificado, em que um único jogador tenta afundar todas as embarcações presentes em um tabuleiro. A cada jogada, uma posição (x,y) é selecionada para ataque e uma mensagem é exibida indicando se uma unidade inimiga foi ou não atingida. O jogo termina quando todas as embarcações forem destruídas. Escreva um programa do jogo acima usando um dicionário para representar o tabuleiro, tal que as chaves correspondam às posições das embarcações, com os valores indicando a pontuação correspondente de acerto.

Solução 1:

A solução abaixo assume um tabuleiro fixo com um submarino, um cruzador e um porta-aviões. Para cada acerto, a pontuação da unidade atingida é contabilizada e zerada no tabuleiro.

```
1  def carrega_mapa():
2      return {(2,5):100, # submarino
3              (20,60):30, (21,60):30, # cruzador
4              (42,71):20, (43,71):20, (44,71):20, (45,71):20 # pe
5              }
6
7  def total_pontos(mapa):
8      total = 0
9      for valor in mapa.values():
10         total += valor
11     return total
12
13  def main():
14     mapa = carrega_mapa()
15     total = total_pontos(mapa)
16     pontuacao_jogador = 0
17     while pontuacao_jogador < total:
18         x = int(input("Digite x: "))
19         y = int(input("Digite y: "))
20         chave = (x,y)
21         if chave in mapa:
22             if mapa[chave] > 0:
23                 print("Acertou!")
24                 pontuacao_jogador += mapa[chave]
25                 mapa[chave] = 0
26             else:
27                 print("Acerto repetido.")
28         else:
29             print("Errou!")
30     print("Fim.")
31
32  main()
```