Linguagem de Programação Python

Dicionários e Tuplas

Professor: Ritomar Torquato

08 Dicionários e Tuplas

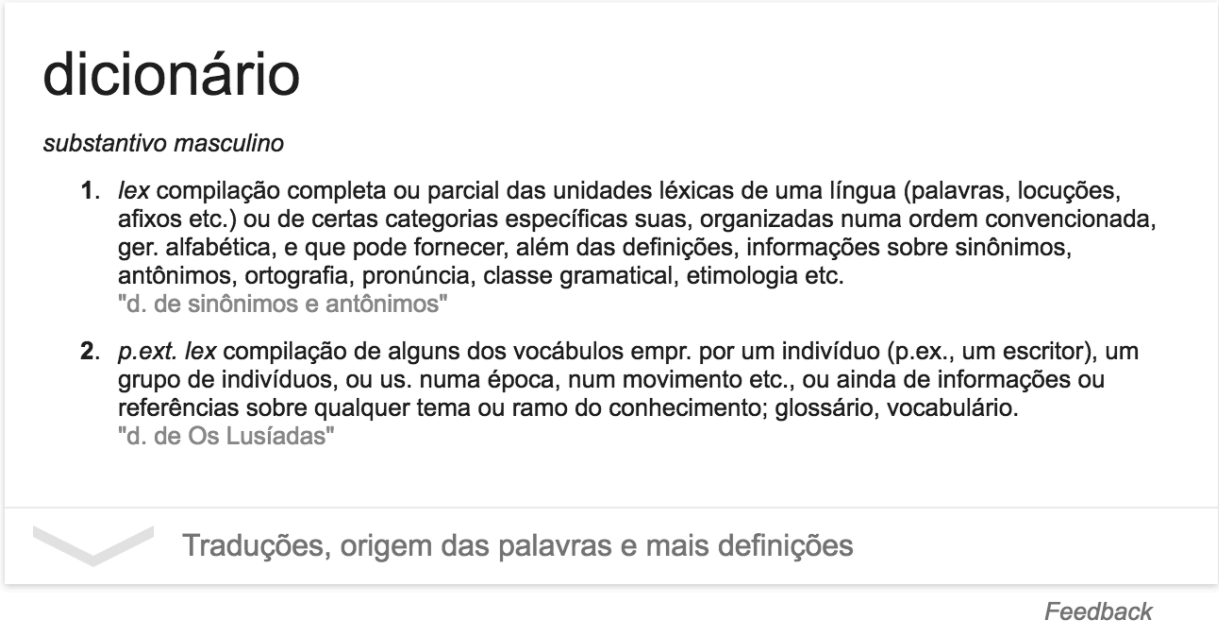
Objetivos: Conhecer os tipos de dados Dicionários e Tuplas;

Introdução

Dicionários são estruturas de dados similar às listas, mas com propriedades de **acesso** diferentes.

Introdução

Dicionários são estruturas de dados similar às listas, mas com propriedades de **acesso** diferentes.



dicionário

substantivo masculino

1. *lex* compilação completa ou parcial das unidades léxicas de uma língua (palavras, locuções, afixos etc.) ou de certas categorias específicas suas, organizadas numa ordem convencionada, ger. alfabética, e que pode fornecer, além das definições, informações sobre sinônimos, antônimos, ortografia, pronúncia, classe gramatical, etimologia etc.
   "d. de sinônimos e antônimos"

2. *p.ext. lex* compilação de alguns dos vocábulos empr. por um indivíduo (p.ex., um escritor), um grupo de indivíduos, ou us. numa época, num movimento etc., ou ainda de informações ou referências sobre qualquer tema ou ramo do conhecimento; glossário, vocabulário.
   "d. de Os Lusíadas"

Traduções, origem das palavras e mais definições

Feedback

Relacionam chave e valor.

Introdução

Nas listas cada valor é acessado por seu índice.

[	Alface	Batata	Tomate	Feijão	]
	0	1	2	3	

[	0,45	1,20	2,30	1,50	]
	0	1	2	3	

Nos dicionários o valor é acessado por sua chave.

{	Alface	Batata	Tomate	Feijão	}
	0,45	1,20	2,30	1,50	

Dicionários

Python são { }:

Produto	Preço
Alface	R$ 0,45
Batata	R$ 1,20
Tomate	R$ 2,30
Feijão	R$ 1,50

```
tabela = { "Alface" : 0.45,
           "Batata" : 1.20,
           "Tomate" : 2.30,
           "Feijão" : 1.50 }
```

Cada elemento é uma combinação de chave e valor.

O valor é acessado por sua chave:

```
>>> tabela["Tomate"]
2.3
>>> print("Preço do feijão: %.2f" % tabela["Feijão"])
Preço do feijão: 1.50
```

Dicionários

Quando atribuímos um valor à uma chave:

Caso 1) A chave não existe

```
>>> tabela = { "Alface": 0.45,
               "Batata": 1.20,
               "Feijão": 1.50 }
>>> tabela
{'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
>>> tabela["Tomate"] = 2.50
>>> print(tabela)
{'Tomate': 2.5, 'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
>>> print(tabela["Tomate"])
2.5
```

Uma nova chave é incluída com o valor atribuído.

Dicionários

Quando atribuímos um valor à uma chave:

Caso 2) A chave já existe

```
>>> tabela = { "Alface": 0.45,
               "Batata": 1.20,
               "Tomate": 2.30,
               "Feijão": 1.50 }
>>> tabela
{'Tomate': 2.3, 'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
>>> tabela["Tomate"] = 2.50
>>> print(tabela)
{'Tomate': 2.5, 'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
>>> print(tabela["Tomate"])
2.5
```

O valor da chave é alterado.

Dicionários

Um erro é lançado ao tentar acessar uma chave inexistente:

```
>>> tabela
{'Tomate': 2.5, 'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
>>> tabela["Cebola"]
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    tabela["Cebola"]
KeyError: 'Cebola'
>>> tabela["tomate"]
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    tabela["tomate"]
KeyError: 'tomate'
>>> tabela["Tomate"]
2.5
```

Dicionários

Verificação da existência de uma chave com operador **`in`**:

```
>>> "Manga" in tabela
False
>>> if "Batata" in tabela:
        print("Sopa de batatas")


Sopa de batatas
>>> tabela
{'Tomate': 2.5, 'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
```

Dicionários

Verificação da existência de uma chave com operador **`in`**:

```
>>> for item in tabela:
        print(item + ": R$ %.2f" % tabela[item])


Tomate: R$ 2.50
Batata: R$ 1.20
Alface: R$ 0.45
Feijão: R$ 1.50
>>> print("Feijão" in tabela)
True
```

Dicionários

Obtenção de uma lista de chaves e valores.

```
>>> tabela = { "Alface": 0.45,
               "Batata": 1.20,
               "Tomate": 2.30,
               "Feijão": 1.50 }
>>> tabela.keys()
dict_keys(['Tomate', 'Batata', 'Alface', 'Feijão'])
>>> tabela.values()
dict_values([2.3, 1.2, 0.45, 1.5])
```

Os métodos **`keys()`** e **`values()`** retornam geradores().

Geradores são funções que se comportam como iteráveis e podem ser utilizadas diretamente em loops **`for`**.

Dicionários

Obtenção de uma lista de chaves e valores.

```
>>> tabela.keys()
dict_keys(['Tomate', 'Batata', 'Alface', 'Feijão'])
>>>
>>> tabela.values()
dict_values([2.3, 1.2, 0.45, 1.5])
>>>
>>> chaves = list(tabela.keys())
>>> valores = list(tabela.values())
>>> chaves
['Tomate', 'Batata', 'Alface', 'Feijão']
>>> valores
[2.3, 1.2, 0.45, 1.5]
```

Podem ser transformados em listas usando **`list()`**.

Dicionários

Excluir uma entrada do dicionário com **`del`**.

```
>>> tabela = { "Alface": 0.45,
               "Batata": 1.20,
               "Tomate": 2.30,
               "Feijão": 1.50 }
>>> tabela
{'Tomate': 2.3, 'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
>>> del tabela["Tomate"]
>>> tabela
{'Batata': 1.2, 'Alface': 0.45, 'Feijão': 1.5}
```

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo outro dicionário.

```
estoque = { "tomate": [ 1000, 2.30],
            "alface": [  500, 0.45],
            "batata": [ 2001, 1.20],
            "feijão": [  100, 1.50] }

ImprimeEstoque("Estoque Atual", estoque)

venda = [ ["tomate", 5], ["batata", 10], ["alface", 5] ]
total = 0
```

A chave (produto) está associada a uma lista com quantidade e valor.

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo à outro dicionário.

```
ESTOQUE ATUAL:

| ------------- | ---------- | -------------- | ------------- |
| Descrição     | Quantidade | Preço Unitário | Total Estoque |
| ------------- | ---------- | -------------- | ------------- |
| feijão        |        100 |           1.50 |        150.00 |
| alface        |        500 |           0.45 |        225.00 |
| tomate        |       1000 |           2.30 |       2300.00 |
| batata        |       2001 |           1.20 |       2401.20 |
| ------------- | ---------- | -------------- | ------------- |
```

Na função ImprimeEstoque o rótulo é exibido acima do estoque que é mostrado e um formato especial de tabela.

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo à outro dicionário.

```
print("\n  VENDA: \n")
for operação in venda:
    produto, quantidade = operação
    preço = estoque[produto][1]
    custo = preço * quantidade
    print("%12s: %3d x %6.2f = %6.2f" %
          (produto, quantidade, preço, custo))
    estoque[produto][0] -= quantidade
    total += custo

print("\n CUSTO TOTAL: %21.2f\n" % total)

ImprimeEstoque("Novo Estoque", estoque)
```

Para cada operação da venda...

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo à outro dicionário.

```
print("\n  VENDA: \n")
for operação in venda:
    produto, quantidade = operação
    preço = estoque[produto][1]
    custo = preço * quantidade
    print("%12s: %3d x %6.2f = %6.2f" %
          (produto, quantidade, preço, custo))
    estoque[produto][0] -= quantidade
    total += custo

print("\n CUSTO TOTAL: %21.2f\n" % total)

ImprimeEstoque("Novo Estoque", estoque)
```

...fazemos o desempacotamento da operação em produto e quantidade

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo à outro dicionário.

```
print("\n  VENDA: \n")
for operação in venda:
    produto, quantidade = operação
    preço = estoque[produto][1]
    custo = preço * quantidade
    print("%12s: %3d x %6.2f = %6.2f" %
          (produto, quantidade, preço, custo))
    estoque[produto][0] -= quantidade
    total += custo

print("\n CUSTO TOTAL: %21.2f\n" % total)

ImprimeEstoque("Novo Estoque", estoque)
```

Recuperamos o preço da Lista de Dados do Produto.

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo à outro dicionário.

```
print("\n  VENDA: \n")
for operação in venda:
    produto, quantidade = operação
    preço = estoque[produto][1]
    custo = preço * quantidade
    print("%12s: %3d x %6.2f = %6.2f" %
          (produto, quantidade, preço, custo))
    estoque[produto][0] -= quantidade
    total += custo

print("\n CUSTO TOTAL: %21.2f\n" % total)

ImprimeEstoque("Novo Estoque", estoque)
```

Atualizamos o estoque, subtraindo a quantidade vendida.

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo à outro dicionário.

```
NOVO ESTOQUE:

| ------------- | ---------- | -------------- | ------------- |
| Descrição     | Quantidade | Preço Unitário | Total Estoque |
| ------------- | ---------- | -------------- | ------------- |
| feijão        |        100 |           1.50 |        150.00 |
| alface        |        495 |           0.45 |        222.75 |
| tomate        |        995 |           2.30 |       2288.50 |
| batata        |       1991 |           1.20 |       2389.20 |
| ------------- | ---------- | -------------- | ------------- |
```

Após o processamento da venda a função ImprimeEstoque é novamente chamada para exibir o estoque atualizado.

Dicionários com listas

Podemos ter dicionários com chaves associadas à listas ou mesmo à outro dicionário.

```
def ImprimeEstoque(rótulo, estoque):
    print("\n  " + rótulo.upper() + ":\n")
    print("| ------------- | --------- | ------------- | ------------ |")
    print("| Descrição     | Quantidade | Preço Unitário | Total Estoque |")
    print("| ------------- | --------- | ------------- | ------------ |")
    for chave, dados in estoque.items():
        print('| {0:14} | {1:10d} | {2:14.2f} | {3:13.2f} |'.format(
            chave, dados[0], dados[1], dados[0] * dados[1]))
    print("| ------------- | --------- | ------------- | ------------ |")
```

A função ImprimeEstoque recebe por parâmetro um rótulo e o dicionário com o estoque.

Utilizamos o método items do dicionário que retorna uma tupla contendo a chave e o valor de cada item do dicionário.

Tuplas

( ,  , )

Uma tupla é uma lista de elementos **imutáveis**.

Tuplas

Tuplas em Python são ( ):

```
>>> tupla = ("Mickey Mouse", "Fred Flintstone", "Bart Simpson")
>>> tupla
('Mickey Mouse', 'Fred Flintstone', 'Bart Simpson')
```

Como nas listas, o valor é acessado por seu índice:

```
>>> tupla[0]
'Mickey Mouse'
```

Mas não podem ser seu valor modificado:

```
>>> tupla[0] = 'Frajola'
Traceback (most recent call last):
  File "<pyshell#132>", line 1, in <module>
    tupla[0] = 'Frajola'
TypeError: 'tuple' object does not support item assignment
```

Tuplas

Tuplas suportam a maior parte das operações de listas, como fatiamento e indexação:

```
>>> tupla = ("a", "b", "c")
>>> tupla
('a', 'b', 'c')
>>> tupla[0]
'a'
>>> tupla[2]
'c'
>>> tupla[ 1: ]
('b', 'c')
>>> tupla * 2
('a', 'b', 'c', 'a', 'b', 'c')
>>> len(tupla)
3
```

Podem ser usadas com **`for`**:

```
>>> for elemento in tupla:
        print(elemento)


a
b
c
```

Tuplas

Podemos criar tuplas sem o uso dos parênteses explícitos :

```
>>> tupla = 100, 200, 300
>>> tupla
(100, 200, 300)
```

Podem ser utilizadas para o desempacotamento de valores:

```
>>> a, b, = 10, 20
>>> a
10
>>> b
20
```

Também podemos trocar o valor de variáveis:

```
>>> a, b = 10, 20
>>> a, b, = b, a
>>> a
20
>>> b
10
```

Tuplas

Existe uma sintaxe especial para criação de tuplas **com apenas um elemento**. Usamos (,) vírgula:

```
>>> t2 = (1,)
>>> t2
(1,)
>>> t3 = 1,
>>> t3
(1,)
```

Sem o uso da vírgula, temos:

```
>>> t1 = (1)
>>> t1
1
```

Um número Inteiro.

Tuplas

Podemos criar uma tupla vazia usando apenas os parenteses:

```
>>> t4 = ()
>>> t4
()
>>> len(t4)
0
```

Tuplas podem ser criadas a partir de Listas com **`tuple`**

```
>>> L = [1, 2, 3]
>>> T = tuple(L)
>>> T
(1, 2, 3)
```

Tuplas

Não podemos alterar uma tupla depois da criação mas podemos concatená-las, gerando novas tuplas:

```
>>> t1 = 1, 2, 3
>>> t2 = 4, 5, 6
>>> t3 = t1 + t2
>>> t3
(1, 2, 3, 4, 5, 6)
```

Tuplas

Se uma tupla contiver um elemento que pode ser alterado, como uma lista, este continuará com seu funcionamento normal:

```
>>> tupla = ('a', ['b', 'c', 'd'])
>>> tupla
('a', ['b', 'c', 'd'])
>>> len(tupla)
2
>>> len(tupla[1])
3
>>> tupla[1]
['b', 'c', 'd']
>>> tupla[1].append('e')
>>> tupla
('a', ['b', 'c', 'd', 'e'])
```

A tupla não foi alterada.

A lista que ela contém foi alterada.

Prática

- Modifique o exemplo dos Slides de 15 a 22 para que seja solicitado do usuário o produto e a quantidade vendida. Verifique se o nome do produto existe no dicionário e só então efetue a venda.

Prática

- Dada uma tupla, verifique se um determinado valor está dentro dela.
- Dada uma tupla, itere sobre a tupla, imprimindo cada um de seus elementos.