

Python

Sequencias Multidimensionais

Aparecido Vilela Junior

Aparecido.vilela@unicesumar.edu.br

Conteúdo

Dicionários

Conceitos

Operações

Métodos

Exercícios

Dicionários

São estruturas de dados que implementam ***mapeamentos***

Um mapeamento é uma **coleção** de associações entre **pares** de valores

O primeiro elemento do par é chamado de ***chave*** e o outro de ***valor***

chave	valor
chave	valor
chave	valor

Dicionários

- Um mapeamento é uma **generalização** da idéia de acessar dados por **índices**, exceto que, num mapeamento, os índices (ou chaves) podem ser de **qualquer tipo**
 - *Geralmente strings e inteiros*

Dicionários

- Dicionários representam outra **estrutura de dados** interna de Python
 - **Hash tables**
- **Listas** → indexadas por **inteiros**
- **Dicionários** → indexados por **chaves** (keys), que podem ser de qualquer tipo imutável (como strings e inteiros)

Dicionários

- Têm **comprimento variável**, são **heterogêneos** e podem ser **aninhados**
- São delimitados por **{ }**
- Lista de pares **chave/valor** separados por vírgulas dentro dos delimitadores (**{ }**)

Operações

Criando o dicionário e seus elementos

```
>>> tel = {'carlos': 4098, 'sergio': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sergio': 4139, 'carlos': 4098, 'guido': 4127}
>>> tel['irving'] = 4127
>>> tel
{'sergio': 4139, 'irving': 4127, 'carlos': 4098, 'guido': 4127}
>>> tel['pedro'] = 8712
>>> tel
{'pedro': 8712, 'sergio': 4139, 'irving': 4127, 'carlos': 4098, 'guido': 4127}
```

- Inserções em posições aleatórias

Operações

As chaves dos dicionários não são armazenadas em qualquer ordem específica

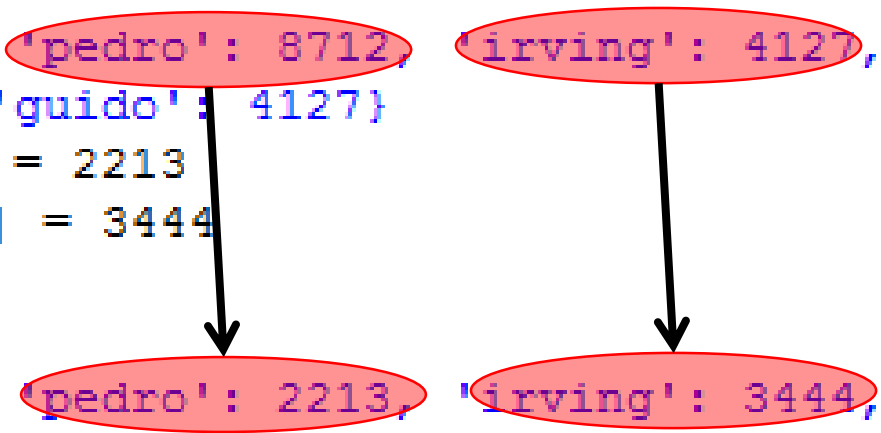
Na verdade, dicionários são implementados por tabelas de espalhamento (*Hash Tables*)

A falta de ordem é **proposital**

Operações

Modificando elementos

```
>>> tel
{'sergio': 4139, 'pedro': 8712, 'irving': 4127,
 'carlos': 4098, 'guido': 4127}
>>> tel['pedro'] = 2213
>>> tel['irving'] = 3444
>>>
>>> tel
{'sergio': 4139, 'pedro': 2213, 'irving': 3444,
 'carlos': 4098, 'guido': 4127}
```



Operações

- Elementos heterogêneos/removendo elementos

```
>>> tel[21] = 5561
>>> tel
{'pedro': 2213, 'sergio': 4139, 'irving': 3444,
21: 5561, 'guido': 4127, 'carlos': 4098}
>>> del tel[21]
>>> del tel['sergio']
>>> tel
{'pedro': 2213, 'irving': 3444, 'guido': 4127,
'carlos': 4098}
```

Operações

Listas de chaves e valores

```
>>> tel
{'pedro': 2213, 'irving': 3444, 'guido': 4127,
 'carlos': 4098}
>>> tel.keys()
['pedro', 'irving', 'guido', 'carlos']
>>> tel.values()
[2213, 3444, 4127, 4098]
>>> tel.has_key('guido')
True
>>> tel.has_key('amanda')
False
```

items() retorna uma **lista** com todos os pares **chave/valor** do dicionário

Mais Operações

Acesso a valores e chaves

```
>>> D1 = {}      # dicionario vazio
>>> D2 = {'spam' : 2, 'eggs' : 3}  # dicionario de tamanho = 2
>>>
>>> len(D2)
2
>>> D2.has_key('eggs')
True
>>> 'eggs' in D2
True
>>> print "chaves", D2.keys()
chaves ['eggs', 'spam']
>>> print "valores", D2.values()
valores [3, 2]
>>> D2.get('eggs')
3
```

Métodos

clear()

Remove todos os elementos do dicionário

```
>>> x = { "Joao":"a", "Maria":"b" }  
>>> x.clear()  
>>> print x  
{}
```

Métodos

- **copy()**
 - Retorna um outro dicionário com os mesmos pares chave/conteúdo

```
>>> x = {"Joao":[1,2], "Maria":[3,4]}
>>> y = x.copy()
>>> y ["Pedro"]=[5,6]
>>> print x
{'Joao': [1, 2], 'Maria': [3, 4]}
>>> print y
{'Pedro': [5, 6], 'Joao': [1, 2], 'Maria': [3, 4]}
```

Métodos

- **pop(chave)**
 - Obtém o **valor** correspondente à **chave** e **remove** o par chave/valor do dicionário

```
>>> y
{'Pedro': [5, 6], 'Joao': [1, 2], 'Maria': [3, 4]}
>>> y.pop('Joao')
[1, 2]
>>> y
{'Pedro': [5, 6], 'Maria': [3, 4]}
>>>
```

Métodos

- **iteritems()**
 - Possibilita que cada chave/valor sejam recuperados em um for

```
>>> x
{'a': 1, 'c': 3, 'b': 7, 'z': 9}
>>> for chave,elem in x.iteritems():
        print chave, elem
```

```
a 1
c 3
b 7
z 9
```


Métodos

update(dic)

Atualiza um dicionário com os elementos de outro

Os itens em *dic* são **atualizados** ou **adicionados** um a um ao dicionário original

```
>>> x = {'a':1, 'b':2, 'c':3}
>>> y = {'z':9, 'b':7}
>>> x.update(y)
>>> x
{'a': 1, 'c': 3, 'b': 7, 'z': 9}
>>> x.update(a=7, c='ceca', d=18)
>>> x
{'a': 7, 'c': 'ceca', 'b': 7, 'd': 18, 'z': 9}
```

Mais Operações

Aninhamento

```
>>> D2 = {'spam' : 2, 'eggs' : 3}
>>> D3 = {'food': {'ham': 1, 'egg': 2}}    # aninhado
>>>
>>>
>>> D3['food']
{'egg': 2, 'ham': 1}
>>> D3['food']['ham']
1
>>> D2.update(D3)

>>> D2
{'food': {'egg': 2, 'ham': 1}, 'eggs': 3, 'spam': 2}
>>> len(D2)
3
>>> del D2['eggs']
>>> D2
{'food': {'egg': 2, 'ham': 1}, 'spam': 2}
```

Dados

Utilizado também para criar estruturas usadas em manipulação de dados

Exemplo:

Estruturas de registros

Registros de Dados

```
>>> #records
>>> bob = {'name': 'Bob Smith', 'age':42, 'pay':30000, 'job':'dev'}
>>> sue = {'name': 'Sue Jones', 'age':45, 'pay':40000, 'job':'mus'}
>>> tom = {'name': 'Tom', 'age':50, 'pay':0, 'job':None}
>>>
>>> #database
>>> db = {}
>>> db['bob'] = bob
>>> db['sue'] = sue
>>> db['tom'] = tom
>>>
>>> db
{'bob': {'job': 'dev', 'pay': 30000, 'age': 42, 'name': 'Bob Smith'}, 'sue': {'job': 'mus', 'pay': 40000, 'age': 45, 'name': 'Sue Jones'}, 'tom': {'job': None, 'pay': 0, 'age': 50, 'name': 'Tom'}}
```

Exercícios

Exercícios

1 – Dada a tabela a seguir, crie um dicionário que a represente:

Lanchonete	
Produtos	Preços R\$
Salgado	R\$ 4.50
Lanche	R\$ 6.50
Suco	R\$ 3.00
Refrigerante	R\$ 3.50
Doce	R\$ 1.00

2 – Considere um dicionário com 5 nomes de alunos e suas notas. Escreva um programa que calcule a média dessas notas.

Exercícios

2. Crie um dicionário que é uma agenda e coloque nele os seguintes dados: chave, nome, idade, telefone. O programa deve ler um número indeterminado de dados, criar a agenda e imprimir todos os itens do dicionário no formato *chave: nome-idade-fone*.