

Dicionários

Prof. Alberto Costa Neto
Programação em Python

Dois tipos de Coleções

- List (Lista)

- > Uma coleção linear de valores que ficam em ordem

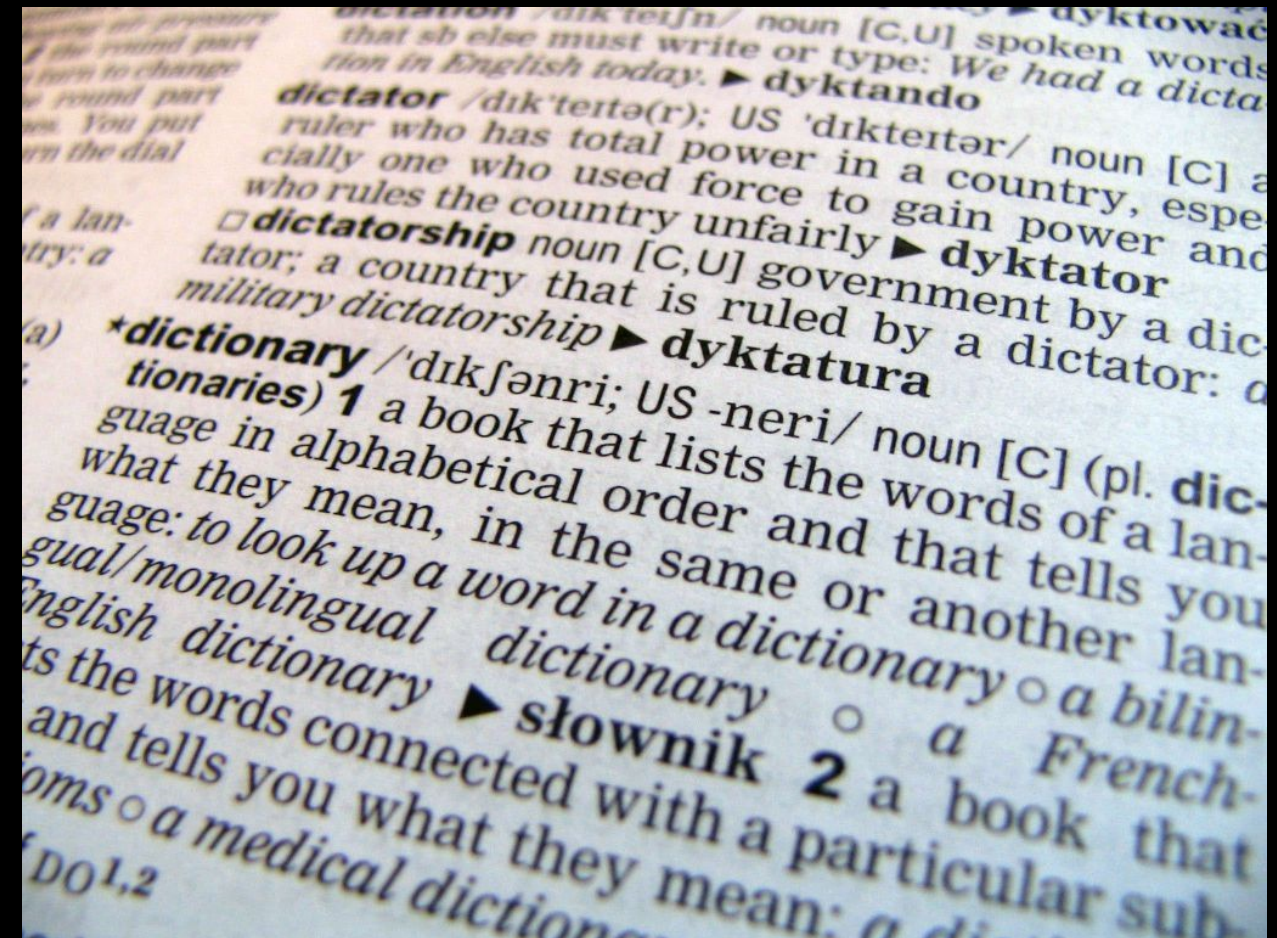


- Dictionary (Dicionário)

- > Um “saco” de valores, onde cada valor tem um rótulo (chave) identificando-o



Dicionários



Dicionários

- São as **coleções de dados mais poderosas** de Python
- Permitem operações rápidas no estilo de banco de dados em Python
- Têm diferentes denominações em outras linguagens de programação
 - > **Arrays Associativos** - Perl / PHP / Lua
 - > **Properties, Map e Hashtable** - Java
 - > **Property Bag** - C# / .Net

Dicionários

- Listas **indexam** suas entradas baseado-se na posição na lista
- **Dicionários** são como sacolas – sem noção de ordem
- Então a **indexação** é através das “lookup tags” (rótulos de busca/chaves)

```
>>> bolsa = dict()
>>> bolsa['dinheiro'] = 12
>>> bolsa['doce'] = 3
>>> bolsa['lenço'] = 75
>>> print(bolsa)
{'dinheiro': 12, 'doce': 3, 'lenço': 75}
>>> print(bolsa['doce'])
3
>>> bolsa['doce'] = bolsa['doce'] + 2
>>> print(bolsa)
{'dinheiro': 12, 'lenço': 75, 'doce': 5}
```

Literais Dicionário (Constantes)

- Usam **chaves** e têm uma lista de pares (**chave** : **valor**)
- Para criar um **dicionário vazio** usamos chaves **{ }**

```
>>> dic_vazio = { }
>>> print(dic_vazio)
{}
>>> dic = {'jose' : 1 , 'fred' : 42, 'maria': 100}
>>> print(dic)
{'jose': 1, 'fred': 42, 'maria': 100}
>>>
```

Comparando

Listas e Dicionários

- Dicionários são como Listas exceto que usam chaves (rótulos de busca) no lugar de números para buscar valores

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> dic = dict()
>>> dic['idade'] = 21
>>> dic['curso'] = 182
>>> print(dic)
{'curso': 182, 'idade': 21}
>>> dic['idade'] = 23
>>> print(dic)
{'curso': 182, 'idade': 23}
```

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> dic = dict()
>>> dic['idade'] = 21
>>> dic['curso'] = 182
>>> print(dic)
{'curso': 182, 'idade': 21}
>>> dic['idade'] = 23
>>> print(dic)
{'curso': 182, 'idade': 23}
```

Lista

Chave	Valor
-------	-------

[0]	21
[1]	183

lst

Dicionário

Chave	Valor
-------	-------

['curso']	183
['idade']	21

dic

Chaves do tipo inteiro nos Dicionários

- Dicionários suportam até chaves do tipo inteiro. Neste caso, ficam com sintaxe de uso muito semelhante à das listas

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> dic = dict()
>>> dic[1] = 21
>>> dic[20] = 182
>>> print(dic)
{1: 21, 20: 182}
>>> dic[20] = 23
>>> print(dic)
{1: 21, 20: 23}
```

Copiando/Limpando Dicionários e Apagando itens

```
>>> dic = {'jose' : 1 , 'fred' : 42, 'maria': 100}
>>> dic1 = dic
>>> dic2 = dic.copy()
>>> valor = dic.pop('jose')
>>> print('Removido', valor, dic1)
Removido 1 {'fred': 42, 'maria': 100}
>>> dic.clear()
>>> print(dic1)
{}
>>> print(dic2)
{'jose': 1, 'fred': 42, 'maria': 100}
>>>
```

Atualizando dicionário a partir de outro dicionário

```
>>> d1 = {'jose' : 1, 'fred' : 42}
>>> d2 = {'jose' : 10, 'maria' : 100}
>>> d3 = {}
>>> d3.update(d1)
>>> print(d3)
{'jose': 1, 'fred': 42}
>>> d3.update(d2)
>>> print(d3)
{'jose': 10, 'fred': 42, 'maria': 100}
>>>
```

- Ao aplicar `update` sobre `d3`, passando `d1` como parâmetro, as **chaves e valores de `d1`** serão incluídos em `d3`.
- Quando `d3` já contém alguma chave contida em `d1`, o **valor atual em `d3` será substituído pelo valor de `d1`**