


Live de Python #29

Collections #2 - Counter, defaultdict, OrderedDict

Roteiro

- O que são e como usamos dicionários??
- `__hash__` e eu com isso?
- Dicionários do collections de hoje
 - Counter
 - defaultdict
 - OrderedDict
- Referências
 - www.sharats.me/posts/the-python-dictionary





O que são e como
usamos
dicionários?

O que são dicionários? [0]

Os dicionários (type dict) são uma estrutura de dados muito poderosa, não apenas no Python. Eles estão presentes em quase todas as linguagens modernas de alto nível, às vezes chamadas de mapas, hashes ou matrizes associativas.



O que são dicionários? [1]

Os dicionários são compostos por chaves e valores e podem ser construídos das seguintes formas:

```
In [1]: dicionario_1 = {'chave1': 'valor1', 'chave2': 'valor2'}  
In [2]: dicionario_2 = dict(chave1='valor1', chave2='valor2')
```



O que são dicionários? [3]

E seus valores podem ser acessados usando as chaves

```
In [3]: dicionario_1['chave1']  
Out[3]: 'valor1'  
  
In [4]: dicionario_2['chave2']  
Out[4]: 'valor2'
```

O que são dicionários? [4]

Tipos de dados que podem ser chaves:

Na maioria das vezes usamos como chaves uma string, mas, na verdade, podem ser usados qualquer tipo de dados que sejam imutáveis ou hashable

```
In [5]: dicionario_2[5] = 7
```

```
In [6]: dicionario_2
```

```
Out[6]: {5: 7, 'chave1': 'valor1', 'chave2': 'valor2'}
```

Como usamos? [0]

```
In [1]: abobrinhas = {'feijão': 'arroz', 'macarrão': 'molho'}
```

```
In [2]: abobrinhas.keys()
```

```
Out[2]: dict_keys(['feijão', 'macarrão'])
```

```
In [3]: abobrinhas.items()
```

```
Out[3]: dict_items([('feijão', 'arroz'), ('macarrão', 'molho')])
```

```
In [4]: abobrinhas.values()
```

```
Out[4]: dict_values(['arroz', 'molho'])
```

```
In [5]: abobrinhas.update({'cebola': 'salsa'})
```

```
In [6]: abobrinhas
```

```
Out[6]: {'cebola': 'salsa', 'feijão': 'arroz', 'macarrão': 'molho'}
```


Como usamos? [1]

```
In [15]: digitos = [0, 1, 2, 3]
```

```
In [16]: strings = ['zero', 'um', 'dois', 'três']
```

```
In [17]: numeros = {digito:string for digito, string in zip(digitos, strings)}
```

```
In [18]: numeros
```

```
Out[18]: {0: 'zero', 1: 'um', 2: 'dois', 3: 'três'}
```

```
In [19]: dict((digito, string) for digito, string in zip(digitos, strings))
```

```
Out[19]: {0: 'zero', 1: 'um', 2: 'dois', 3: 'três'}
```

`__hash__`

__hash__ e eu com isso? [0]

Mas nem tudo são flores, os objetos hashable em python implementam o método '.__hash__'. Mas alguns casos, o hash pode nos induzir a erros.

```
In [12]: hash(1+0j)
```

```
Out[12]: 1
```

```
In [13]: hash(1.0)
```

```
Out[13]: 1
```

```
In [14]: hash(True)
```

```
Out[14]: 1
```

```
In [15]: hash(1)
```

```
Out[15]: 1
```

```
In [16]: dicionario_2[1] = True
```

```
In [17]: dicionario_2[True]
```

```
Out[17]: True
```

```
In [18]: dicionario_2[1.0]
```

```
Out[18]: True
```

```
In [19]: dicionario_2[1+0j]
```

```
Out[19]: True
```

__hash__ e eu com isso? [1]

Esses são ou não são o mesmo objeto?

```
In [7]: class Pessoa:
...:     def __init__(self, nome):
...:         self.nome = nome
...:     def __hash__(self):
...:         return hash(self.nome)
...:
```

```
In [8]: du = Pessoa('eduardo')
```

```
In [9]: dudu = Pessoa('eduardo')
```

```
In [10]: hash(du) == hash(dudu)
```

```
Out[10]: True
```

__hash__ e eu com isso? [2]

Esses são ou não são o mesmo objeto?

```
In [11]: du is dudu
```

```
Out[11]: False
```

```
In [12]: id(du), id(dudu)
```

```
Out[12]: (140372903472656, 140372894378976)
```

The background is a solid pink color. In the top right corner, there is a geometric pattern consisting of several squares and triangles in different shades of pink, creating a stepped or architectural effect.

collections

OrderedDict

Problema a ser resolvido: Dicionários não mantêm ordem.

```
In [28]: dict((digito, string) for digito, string in zip(strings, digitos))  
Out[28]: {'dois': 2, 'três': 3, 'um': 1, 'zero': 0}
```

```
In [29]: from collections import OrderedDict
```

```
In [30]: OrderedDict((digito, string) for digito, string in zip(strings, digitos))  
Out[30]: OrderedDict([('zero', 0), ('um', 1), ('dois', 2), ('três', 3)])
```



Counter

Problema a ser resolvido: contar iteráveis

```
In [31]: string = 'Eduardo'
```

```
In [32]: from collections import Counter
```

```
In [33]: Counter(string)
```

```
Out[33]: Counter({'E': 1, 'a': 1, 'd': 2, 'o': 1, 'r': 1, 'u': 1})
```



defaultDict

Problema a ser resolvido: Não estourar exceções quando o valor não estiver no dicionário e inserir-lo com um valor default

```
In [38]: from collections import defaultdict
```

```
In [39]: xpto = defaultdict(lambda: None)
```

```
In [40]: xpto[7]
```

```
In [41]: xpto
```

```
Out[41]: defaultdict(<function __main__.<lambda>>, {7: None})
```

```
In [42]: xpto['eduardo']
```

```
In [43]: xpto
```

```
Out[43]: defaultdict(<function __main__.<lambda>>, {7: None, 'eduardo': None})
```