

# Aplicações de Dicionários

Prof. Alberto Costa Neto  
Programação em Python

# Qual é o nome mais comum?

alberto

jose

jose

pedro

maria

luiz

maria

lucia

jose

jose

clara

pedro

# Qual é nome mais comum?

alberto

jose

jose

pedro

maria

luiz

maria

lucia

jose

jose

clara

pedro

# Qual é nome mais comum?

alberto

pedro

maria

jose

jose

mar

lucia

clar

alberto

pedro

maria

jose

lucia

clara

luiz

carlos

jose

luiz

jose

pedro

|  |
|--|
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# Muitos Contadores com um Dicionário

- Um uso bastante comum de um dicionário é contar quantas vezes algo ocorre ou foi encontrado

```
>>> dic = dict()
>>> dic['pedro'] = 1
>>> dic['maria'] = 1
>>> print(dic)
{'pedro': 1, 'maria': 1}
>>> dic['pedro'] = dic['pedro'] + 1
>>> print(dic)
{'pedro': 2, 'maria': 1}
```

| Chave   | Valor |
|---------|-------|
| alberto |       |
| pedro   |       |
| maria   |       |
| jose    |       |
| lucia   |       |
| clara   |       |
| luiz    |       |
| carlos  |       |

# Tracebacks de Dicionários

- Um erro comum é referenciar uma chave que não está no dicionário
- Para evitá-lo, podemos usar o operador `in` para testar se a chave está contida no dicionário

```
>>> dic = dict()
>>> print(dic['alberto'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'alberto'
>>> print('alberto' in dic)
False
```

# E quando surge um novo valor?

- Quando encontramos um novo nome, precisamos adicionar uma nova entrada no **dicionário**.
- Da segunda vez em diante que o **nome** é encontrado, basta adicionar 1 ao respectivo contador no dicionário.

```
conts = dict()
nomes = ['jose', 'pedro', 'maria', 'jose', 'maria']
for nome in nomes:
    if nome not in conts:
        conts[nome] = 1
    else:
        conts[nome] = conts[nome] + 1
print(conts)
```

**{'jose': 2, 'pedro': 1, 'maria': 2}**

# O método `get` dos dicionários

- O padrão de checar antecipadamente se a `chave` já está no dicionário e assumir um valor padrão se a `chave` não estiver lá é tão comum que existe um `método` chamado `get()` que faz isso por nós



Adota um valor padrão se a chave não existir (e sem Traceback).

```
{'jose': 2, 'pedro': 1, 'maria': 2}
```

```
if nome in conts:  
    x = conts[nome]  
else :  
    x = 0
```

```
x = conts.get(nome, 0)
```



# Contagem simplificada com `get`

- Podemos usar `get()` e prover o valor padrão 0 quando a chave não estiver ainda no dicionário – e então somente adicionar 1

```
conts = dict()
nomes = ['jose', 'pedro', 'maria', 'jose', 'maria']
for nome in nomes :
    conts[nome] = conts.get(nome, 0) + 1
print(conts)
```

Valor  
padrão



`{'jose': 2, 'pedro': 1, 'maria': 2}`

# Padrão de Contagem

```
conts = dict()

print('Digite uma linha de texto:')
linha = input()
palavras = linha.split()

print('Palavras:', palavras)

print('Contando...')
for p in palavras:
    conts[p] = conts.get(p, 0) + 1
print('Contadores:', conts)
```

O padrão geral para contar as palavras em uma linha de texto é **particionar** a linha em palavras, então iterar pelas palavras e usar um **dicionário** para contabilizar o número de ocorrências de cada palavra independentemente.

# Contando Palavras

```
python contarpalavras.py
```

Digite uma linha de texto:

```
O doce perguntou pro doce qual é o doce mais doce. O  
doce respondeu pro doce que o doce mais doce é o doce  
de batata doce.
```

```
Palavras: ['O', 'doce', 'perguntou', 'pro', 'doce',  
'qual', 'é', 'o', 'doce', 'mais', 'doce.', 'O',  
'doce', 'respondeu', 'pro', 'doce', 'que', 'o',  
'doce', 'mais', 'doce', 'é', 'o', 'doce', 'de',  
'batata', 'doce.']
```

Contando...

```
Contadores: {'O': 2, 'doce': 8, 'perguntou': 1, 'pro':  
2, 'qual': 1, 'é': 2, 'o': 3, 'mais': 2, 'doce.': 2,  
'respondeu': 1, 'que': 1, 'de': 1, 'batata': 1}
```



<https://img.cybercook.uol.com.br/imagens/receitas/425/doce-de-batata-doce-1.jpg>

# Laços Definidos e Dicionários

Ainda que **dicionários** não sejam armazenados em ordem, nós podemos escrever um laço **for** que percorre todas as **entradas** em um **dicionário**

- Na prática iteramos por todas as **chaves** contidas no **dicionário** e **buscamos** o **valor** que está associado a cada chave no dicionário

```
>>> conts = { 'jose' : 1 , 'fred' : 42, 'maria': 100}
>>> for chave in conts:
...     print(chave, conts[chave])
...
jose 1
fred 42
maria 100
>>>
```

# Recuperando listas de Chaves e Valores

- Você pode obter uma lista de **chaves**, **valores**, ou **itens** (contém ambos juntos) de um dicionário

```
>>> dic = {'jose': 1, 'fred': 42, 'maria': 100}
>>> print(dic)
{'jose': 1, 'fred': 42, 'maria': 100}
>>> print(list(dic.keys()))
['jose', 'fred', 'maria']
>>> print(list(dic.values()))
[1, 42, 100]
>>> print(list(dic.items()))
[('jan', 100), ('chuck', 1), ('fred', 42)]
>>>
```

↑  
O que é uma 'tupla'?  
em breve...



# Bônus: 2 Variáveis de Iteração!

- Podemos iterar sobre os pares (**chave-valor**) em um dicionário usando **\*duas\*** variáveis de iteração
- Em cada iteração, a primeira variável de iteração recebe uma **chave** e a segunda o **valor** correspondente a esta chave

```
>>> dic = { 'jose':1, 'fred':42, 'maria':100}
>>> for chave,valor in dic.items() :
...     print(chave, valor)
...
jose 1
fred 42
maria 100
>>>
```

| Chave   | Valor |
|---------|-------|
| [jose]  | 1     |
| [fred]  | 42    |
| [maria] | 100   |

# Mais exemplos de aplicação de Dicionários

- Guardar o número de assentos livres em um voo.
  - O dicionário poderia guardar o número do voo como chave e o valor associado seria o número de assentos livres
- Guardar um cardápio de um restaurante.
  - A chave seria o nome do prato e o valor seria o preço
- Obter os dados para gerar um Histograma

# Obter os dados para um Histograma

