

Dicionários

Prof. Alberto Costa Neto
Programação em Python

O que é uma Coleção?



- Uma **coleção** permite colocar vários valores em um única “**variável**”
- **Coleções** são práticas porque permitem carregar **muitos valores empacotados** de forma conveniente pelo programa
- Temos algumas maneiras de acessar as diferentes partes da variável do tipo coleção

O que **não** é uma “Coleção”?

A maioria de nossas **variáveis** têm um valor apenas – quando um novo valor é colocado nela, o antigo é **sobrescrito**

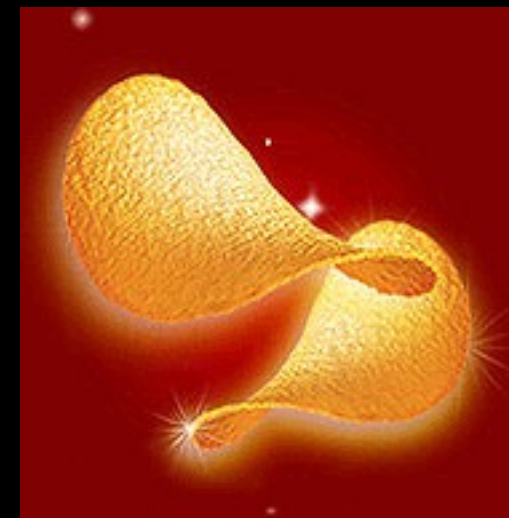
```
$ python  
>>> x = 2  
>>> x = 4  
>>> print x  
4
```



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 identificando-o



Dicionários



calculadora

lenço

perfume

dinheiro

doces



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
 - > 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)

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

Comparando Listas e Dicionários

- Dicionários são com 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
{'course': 182, 'idade': 23}
```

Lista

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

[0]	21
[1]	183

lst

Dicionário

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

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

dic

Literais Dicionário (Constantes)

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

```
>>> dic = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}  
>>> print dic  
{ 'jan': 100, 'chuck': 1, 'fred': 42}  
>>> dic_vazio = { }  
>>> print dic_vazio  
{ }  
>>>
```

Qual é nome mais comum?

marquard	cwen	cwen
zhen	marquard	zhen
csev	zhen	csev
zhen	csev	marquard
		zhen

Qual é o nome mais comum?

Qual é nome mais comum?

marquard	cwen	cwen
zhen	marquard	zhen
csev	zhen	csev
zhen	csev	marquard
		zhen

Qual é nome mais comum?

marquard

cwen

cwen

zhen

zhen

csev

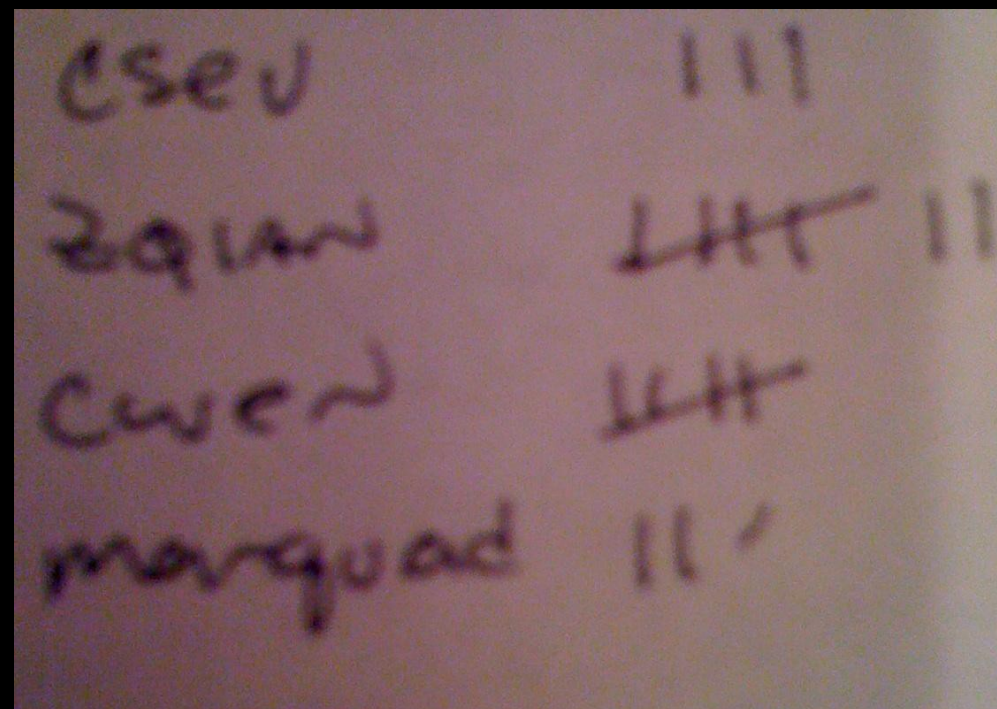
csev

marquard

zhen

csev

zhen



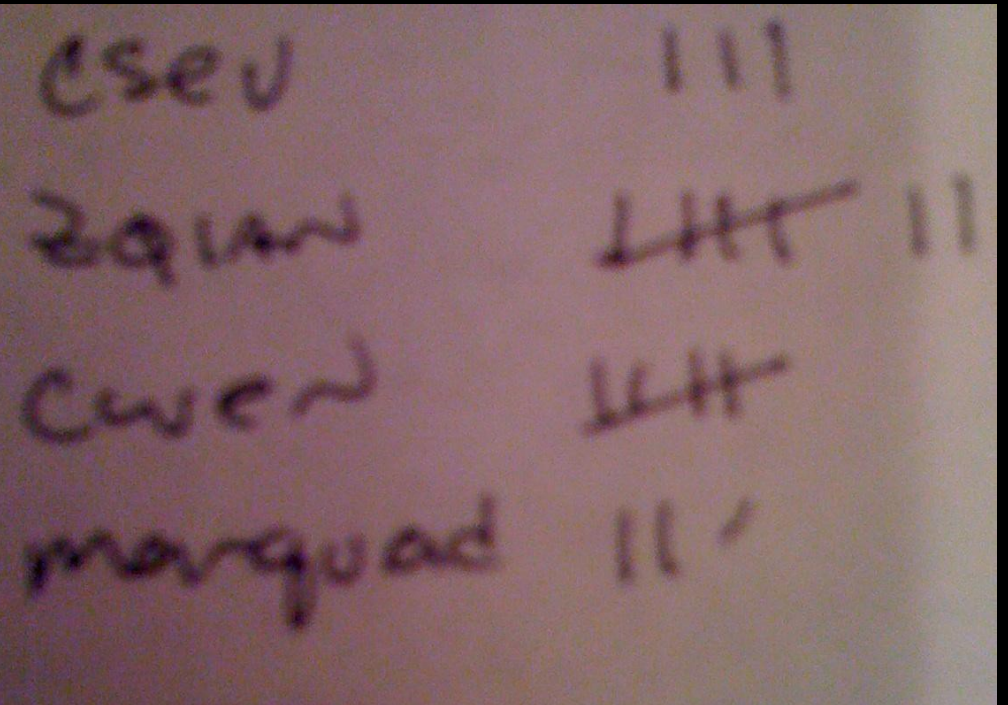
Muitos Contadores com um Dicionários

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

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

Chave

Valor



csev	111
zqian	111
cwen	111
marquod	111

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['csev']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'csev'
>>> print 'csev' 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 = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for nome in nomes:
    if nome not in conts:
        conts[nome] = 1
    else:
        conts[nome] = conts[nome] + 1
print conts
```

{'csev': 2, 'zqian': 1, 'cwen': 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

Valor padrão se a chave não existir (e sem Traceback).

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

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

```
{'csev': 2, 'zqian': 1, 'cwen': 2}
```

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 = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for nome in nomes:
    conts[nome] = conts.get(nome, 0) + 1
print conts
```

Valor
padrão



`{'csev': 2, 'zqian': 1, 'cwen': 2}`

Contagem simplificada com get

```
conts = dict()
nomes = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for nome in nomes :
    conts[nome] = conts.get(nome, 0) + 1
print conts
```



Padrão de Contagem

```
conts = dict()
print 'Digite uma linha de texto:'
linha = raw_input(' ')

palavras = linha.split()

print 'Palavras:', palavras

print 'Contando...'
for palavra in palavras:
    conts[palavra] = conts.get(palavra, 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 wordcount.py`

Digite uma linha de texto:

`the clown ran after the car and the car ran into the tent
and the tent fell down on the clown and the car`

Palavras: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']

Contando...

Contadores {'and': 3, 'on': 1, 'ran': 2, 'car': 3, 'into':
1, 'after': 1, 'clown': 2, 'down': 1, 'fell': 1, 'the': 7,
'tent': 2}



```
conts = dict()
print 'Digite uma linha de texto:'
linha = raw_input(' ')

palavras = linha.split()

print 'Palavras:', palavras

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



python wordcount.py

Digite uma linha de texto:

the clown ran after the car and the car
ran into the tent and the tent fell down
on the clown and the car

Palavras: ['the', 'clown', 'ran', 'after',
'the', 'car', 'and', 'the', 'car', 'ran', 'into',
'the', 'tent', 'and', 'the', 'tent', 'fell',
'down', 'on', 'the', 'clown', 'and', 'the',
'car']

Contando...

Contadores {'and': 3, 'on': 1, 'ran': 2,
'car': 3, 'into': 1, 'after': 1, 'clown': 2,
'down': 1, 'fell': 1, 'the': 7, 'tent': 2}

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 passamos todas as **chaves** no **dicionário** e **buscamos** os respectivos **valores**

```
>>> conts = { 'chuck' : 1 , 'fred' : 42, 'jan' : 100}
>>> for chave in conts:
...     print chave, conts[chave]
...
jan 100
chuck 1
fred 42
>>>
```


Recuperando listas de Chaves e Valores

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

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

Chave	Valor
[jan]	100
[chuck]	1
[fred]	42

```
nome = raw_input('Nome Arquivo:')
arq = open(nome)
texto = arq.read()
palavras = texto.split()

conts = dict()
for palavra in palavras:
    conts[palavra] = conts.get(palavra,0) + 1

freq_cont = None
palavra_cont = None
for palavra,cont in conts.items():
    if freq_cont is None or cont > freq_cont:
        palavra_cont = palavra
        freq_cont = cont

print palavra_cont, freq_cont
```

```
python palavras.py
Nome Arquivo: words.txt
to 16
```

```
python palavras.py
Nome Arquivo: clown.txt
the 7
```



Acknowledgements / Contributions

Agradecimentos / Contribuições



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and open.umich.edu and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information



These slides were translated and adapted by Alberto Costa Neto (albertocn.sytes.net) of the Federal University of Sergipe