

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

POLI – Engenharia de Software

12413 - ALGORITMOS DE PROGRAMAÇÃO, PROJETOS E COMPUTAÇÃO

ASSUNTOS:

- dicionário;

Profa. Angela de Mendonça Engelbrecht

angel@puc-campinas.edu.br

profa.angela@gmail.com

DICIONÁRIO - *dictionary*

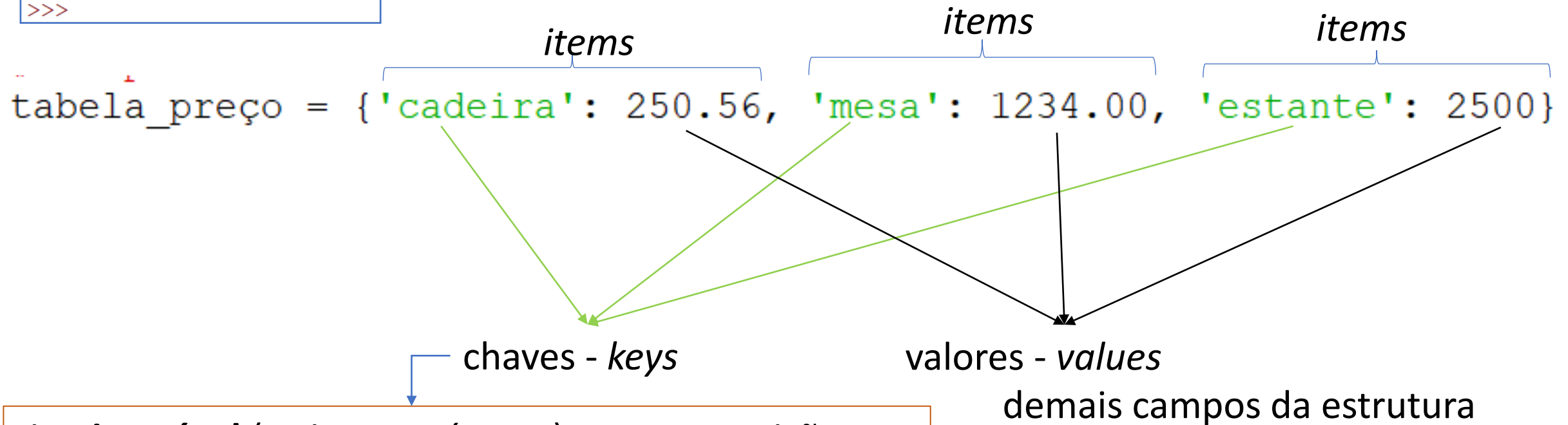
outra estrutura de dados em Python, para armazenar coleções de itens.

- composto de DUAS partes: **{chave: valor(es)}**
- usados **{}** para seus itens, separados por vírgulas e, quando vazios: **{}**, representa um **dicionário vazio**;
- representação – seus elementos são **indexados** por **chaves** – *Keys* – *que podem ser de qualquer tipo imutável – strings e números, por exemplo*;
- as **chaves** são **únicas** num dicionário – se uma chave repetida é inserida, o item anterior é apagado e substituído pelo novo – altera seus valores;
- uma **chave** pode ser apagada/excluída, ela e os valores associados à ela;
- construtor: *dict()*
- *type()* → *<class 'dict'>*;

: separando o campo chave de seus valores

```
>>> #exemplos
>>> tabela_preço = {'cadeira': 250.56, 'mesa': 1234.00, 'estante': 2500}
>>> tabela_preço
{'cadeira': 250.56, 'mesa': 1234.0, 'estante': 2500}
```

```
>>> type (tabela_preço)
<class 'dict'>
>>> len(tabela_preço)
3
>>>
```



tipo **imutável** (*string* ou número) e sem repetição

Seu conteúdo pode ser acessado por qualquer um dos três elementos que o compõe.

```
>>> #exemplos
>>> tabela_preço = {'cadeira': 250.56, 'mesa': 1234.00, 'estante': 2500}
>>> tabela_preço
{'cadeira': 250.56, 'mesa': 1234.0, 'estante': 2500}
```

items

```
>>> tabela_preço.items()
dict_items([('cadeira', 250.56), ('mesa', 1234.0), ('estante', 2500)])
```

keys

```
>>> tabela_preço.keys()
dict_keys(['cadeira', 'mesa', 'estante'])
```

values

```
>>> tabela_preço.values()
dict_values([250.56, 1234.0, 2500])
```

Considere o exemplo:

```
>>> tabela_preço = {'cadeira': 250.56, 'mesa': 1234.00, 'estante': 2500}  
>>> tabela_preço  
{'cadeira': 250.56, 'mesa': 1234.0, 'estante': 2500}
```

incluindo item no dicionário:

indexado pela chave - *key*



resultado →

```
>>> tabela_preço['sofa']=3500  
>>> tabela_preço  
{'cadeira': 250.56, 'mesa': 1234.0, 'estante': 2500, 'sofa': 3500}  
>>>
```

Continuando com o exemplo:

```
>>> tabela_preço  
{'cadeira': 250.56, 'mesa': 1234.0, 'estante': 2500, 'sofa': 3500}
```

alterando o valor de um item - pela chave - *key*:

```
>>> tabela_preço['mesa'] = 1800  
>>> tabela_preço  
{'cadeira': 250.56, 'mesa': 1800, 'estante': 2500, 'sofa': 3500}  
>>>
```

Obs.: se a chave não existir – entenderá que é uma inserção.

Continuando com o exemplo:

```
>>> tabela_preço  
{'cadeira': 250.56, 'mesa': 1800, 'estante': 2500, 'sofa': 3500}
```

excluindo um item pela chave - key:

método del

```
>>> del tabela_preço['estante']  
>>> tabela_preço  
{'cadeira': 250.56, 'mesa': 1800, 'sofa': 3500}  
>>> |
```

dicionários em processos repetitivos – pode ser por qualquer um dos 3 elementos:

item

2 elementos: chave e valor

```
>>> tabela_preço
{'cadeira': 250.56, 'mesa': 1800, 'sofa': 3500}
>>> #loop e os dicts
>>> for produto, preço in tabela_preço.items():
    print(f' Produto: {produto} \nPreço: {preço} ')

Produto: cadeira
Preço: 250.56
Produto: mesa
Preço: 1800
Produto: sofa
Preço: 3500
>>> |
```


dicionários em processos repetitivos – pode ser por qualquer um dos 3 elementos:

```
>>> for produto in tabela_preço.keys():  
    print(f' Produto: {produto}')
```

chave

```
Produto: cadeira  
Produto: mesa  
Produto: sofa
```

```
>>> for preço in tabela_preço.values():  
    print(f' Preço: {preço}')
```

valor

```
Preço: 250.56  
Preço: 1800  
Preço: 3500
```

```
>>> |
```

Exemplo compondo tipos de estruturas diferentes:

Supor que queremos guardar, **além** do **preço** de cada produto, a **quantidade** em estoque:

Cada **item** terá: { chave : **[lista com valores]** }

```
>>>
>>> Estoque = {'cadeira':[250.56, 10], 'mesa':[1800,3], 'estante':[2500,2]}
>>> Estoque
{'cadeira': [250.56, 10], 'mesa': [1800, 3], 'estante': [2500, 2]}
>>> len(Estoque)
3
>>> Estoque ['cadeira']
[250.56, 10]
>>> Estoque ['cadeira'] [0]
250.56
>>> Estoque ['cadeira'] [1]
10
>>> |
```

	listas	
Estoque['cadeira']	250.56	10
	[0]	[1]
Estoque['mesa']	1800	3
	[0]	[1]
Estoque['estante']	2500	2
	[0]	[1]

Outros métodos/operações para *dict*:

Excluir: além do *del* já visto anteriormente, há o método *.pop()*

del

```
>>> tabela_preço
{'cadeira': 250.56, 'mesa': 1800, 'estante': 2500, 'sofa': 3500}
>>> del tabela_preço['estante']
>>> tabela_preço
{'cadeira': 250.56, 'mesa': 1800, 'sofa': 3500}
>>> |
```

.pop()

```
>>> Estoque
{'cadeira': [250.56, 10], 'mesa': [1800, 3], 'estante': [2500, 2]}
>>> elim = Estoque.pop('cadeira')
>>> elim
[250.56, 10]
>>> Estoque
{'mesa': [1800, 3], 'estante': [2500, 2]}
>>> |
```

pop() retorna valores do item eliminado

Juntando dois *dicts* – método *.update()*

```
>>> Estoque
{'mesa': [1800, 3], 'estante': [2500, 2]}
>>> Estoque2 = {'sofá': [2500, 3], 'tapete': [200, 10], 'poltrona': [299.99, 6]}
>>> Estoque.update(Estoque2)
>>> Estoque
{'mesa': [1800, 3], 'estante': [2500, 2], 'sofá': [2500, 3], 'tapete': [200, 10], 'poltrona': [299.99, 6]}
>>> Estoque2
{'sofá': [2500, 3], 'tapete': [200, 10], 'poltrona': [299.99, 6]}
>>>
```

Verificar se uma **chave existe** no conjunto:

in/ not in: retorna *True/False*

```
>>> Estoque
{'mesa': [1800, 3], 'estante': [2500, 2], 'sofa': [2500, 3], 'tapete': [200, 10], 'poltrona': [299.99, 6]}
>>> 'sofa' in Estoque
True
>>> 'cama' not in Estoque
True
```

Método *.get()*:

(key, default=None, /)
Return the value for key if key is in the dictionary, else default.

```
>>> Estoque
{'mesa': [1800, 3], 'estante': [2500, 2], 'sofá': [2500, 3], 'tapete': [200, 10], 'poltrona': [299.99, 6]}
>>> Estoque.get('sofá')
[2500, 3]
>>> Estoque.get('sofa')
None
>>> print(Estoque.get('sofa'))
None
>>> Estoque.get('sofa', False)
False
>>> Estoque.get('cama', 'Não existe a chave!')
'Não existe a chave!'
>>> |
```

.fromkeys(list, value) – retorna um dicionário onde as chaves são os elementos de uma lista e os valores das chaves são todos iguais ao valor especificado.

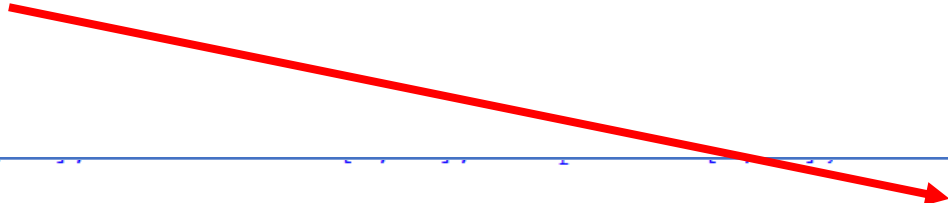
exemplo: criar um dicionário a partir de uma lista e com valores zerados:

```
>>> Lista=['mesa', 'cadeira', 'sofá', 'estante', 'tapete']
>>> Est = {}
>>> Est = Est.fromkeys(Lista,[0,0])
>>> Est
{'mesa': [0, 0], 'cadeira': [0, 0], 'sofá': [0, 0], 'estante': [0, 0], 'tapete': [0, 0]}
```

e, posteriormente modificar os valores iniciados com zero.

```
>>> Est['mesa'][0]= 256.99
>>> Est['mesa'][1] = 5
>>> Est
{'mesa': [256.99, 5], 'cadeira': [0, 0], 'sofá': [0, 0], 'estante': [0, 0], 'tapete': [0, 0]}
>>> Est['cadeira']=[199.99,10]
>>> Est
{'mesa': [256.99, 5], 'cadeira': [199.99, 10], 'sofá': [0, 0], 'estante': [0, 0], 'tapete': [0, 0]}
>>> |
```

.setdefault(key) – insere um novo item no *dict* com a chave especificada e com valor *default: None (nada)*



```
>>> Est.setdefault('poltrona')
>>> Est
{'mesa': [256.99, 5], 'cadeira': [199.99, 10], 'sofá': [0, 0], 'estante': [0, 0], 'tapete': [0, 0], 'poltrona': None}
>>> |
```

há outras operações/métodos para trabalhar com dicionário – busque mais informações em python.org

Exemplo: CADASTRO DE ALUNO

Considerar um conjunto de dados (RA,NOME e NOTA) **de N** alunos (**carga inicial**), estruturados como um **Dicionário**, com **RA** sendo a **chave**.

Construir um programa (carga inicial), que faz a leitura de dados de N alunos e os armazena na estrutura de um dicionário.

Após a leitura, fazer a impressão dos dados dos alunos em **formato** de **tabela** com uma coluna com o **Status**: Aprovado ou Reprovado.

Construir o programa com a seguinte representação:

items

Aluno = { 12345: ['Ana Maria', 8.5], 3248: ['Rosa Maria', 5.5], ..., 3455: ['Maria Rosa', 3.5] }

File Edit Format Run Options Window Help

```
''' exercício - dict '''
```

```
Aluno = {} # RA será o campo chave e como valor: lista [nome, nota]
```

```
TotalAlunos = int(input('Numero de Alunos:'))  
print(' <<< CARGA INICIAL >>>')
```

```
for i in range>TotalAlunos):
```

```
    ra = input('RA: ')
```

```
    # COMO ra é CHAVE precisamos saber se RA digitado já existe
```

```
    if ra in Aluno.keys():
```

```
        print('RA já existe')
```

```
    else:
```

```
        nome = input('Nome: ')
```

```
        nota = float(input('Nota: '))
```

```
        Aluno[ra] = [nome, nota]
```

```
# imprimir tabela de alunos com status de Aprovado/Reprovado
```

```
print('<<< DADOS DOS ALUNOS >>>')
```

```
print('\n  RA\tNOME\t\tNota\tStatus')
```

```
print('-----')
```

```
for ra, dadosaluno in Aluno.items():
```

```
    print(f'\n{ra}  {dadosaluno[0]:<15}  {dadosaluno[1]:4.1f}  ' +
```

```
          ('Aprovado' if dadosaluno[1] >= 5 else 'Reprovado'))
```

```
print('\n-----')
```

```
print('\n <<< Fim de Programa >>>')
```

leitura do RA - chave

leitura dos demais campos

incluindo no dicionário a
lista com os demais dados

Apresentação do resultado do programa

File Edit Format Run Options Window Help

```
''' exercício - dict '''
```

```
Aluno = {} # RA será o campo chave e como valor: lista [nome, nota]
```

```
TotalAlunos = int(input('Numero de Alunos:'))
print(' <<< CARGA INICIAL >>>')
```

```
for i in range(TotalAlunos):
    ra = input('RA: ')
    # COMO ra é CHAVE precisamos saber se RA digitado já existe
    if ra in Aluno.keys():
        print('RA já existe')
    else:
```

```
        nome = input('Nome: ')
        nota = float(input('Nota: '))
        Aluno[ra] = [nome, nota]
```

```
# imprimir tabela de alunos com status de Aprovado/Reprovado
print('<<< DADOS DOS ALUNOS >>>')
print('\n RA\tNOME\t\tNota\tStatus')
print('-----')
```

```
for ra, dadosaluno in Aluno.items():
    print(f'\n{ra} {dadosaluno[0]:<15} {dadosaluno[1]:4.1f} ' +
          ('Aprovado' if dadosaluno[1] >= 5 else 'Reprovado'))
print('\n')
print('\n <<< Fim de Programa>>>')
```

```
imprimir.py
Numero de Alunos:2
<<< CARGA INICIAL >>>
RA: 1234
Nome: Ana Maria
Nota: 8.5
RA: 2345
Nome: Rosa Maria
Nota: 3.5
<<< DADOS DOS ALUNOS >>>

RA      NOME                Nota      Status
-----
1234    Ana Maria              8.5       Aprovado
2345    Rosa Maria              3.5       Reprovado
-----

<<< Fim de Programa>>>
>>> |
```