Python para Análise de Dados: Dicionários e Matrizes

André Grégio
Paulo Almeida

JUSTIÇA 4.0: INOVAÇÃO E EFETIVIDADE NA REALIZAÇÃO DA JUSTIÇA PARA TODOS
PROJETO DE EXECUÇÃO NACIONAL BRA/20/015

Relembrando...

- Numero = [3, 2, 8, 25, 0]
  - A variável “Numero” é uma **lista**!
  - Lista pode ser qualquer coisa:

    - `UmaLista = [1, "batata", (1.73, 61.0)]`
    - `Tamanho da lista: len(UmaLista) → 3`

1. Dicionários

Dicionários

Um dicionário é uma coleção não-ordenada de “valores” indexados por “chaves”.

Dicionários se parecem com listas, mas são mais versáteis.

Dicionários

Como assim “mais versáteis”?

Tudo tem a ver com a indexação!

Dicionários

Um dicionário é uma coleção não-ordenada de “valores” indexados por “chaves”.

Dicionários se parecem com listas, mas são mais versáteis.

```
L1 = [‘a’, ‘b’, ‘c’]
print(list(enumerate(L1)))
```

```
[(0, 'a'), (1, 'b'), (2, 'c')]
```

Dicionários

Um dicionário é uma coleção não-ordenada de “valores” indexados por “chaves”.

Dicionários se parecem com listas, mas são mais versáteis.

```
d1 = { "chave1": "valor1",
       "chave2": "valor2" }
```

```
L1 = [‘a’, ‘b’, ‘c’]
print(list(enumerate(L1)))
```

```
[(0, 'a'), (1, 'b'), (2, 'c')]
```

Dicionários

Um dicionário é uma coleção não-ordenada, mutável, indexada por “chaves”.

- Dic = {}
  - Dicionário vazio!

Dicionários

Dic = {‘chave1’: ‘valor1’, ‘chave2’: ‘valor2’}

- Dicionário com 2 elementos!

Dicionários

Dic = {‘chave1’: ‘valor1’, ‘chave2’: ‘valor2’}

- Dicionário com 2 elementos!
    - Chave indexa o dicionário
        - Ao acessar o dicionário naquela chave, obtenho o valor associado...

Dicionários

Dic = {‘chave1’: ‘valor1’, ‘chave2’: ‘valor2’}

- Chave indexa o dicionário
  - Ao acessar o dicionário naquela chave, obtenho o valor associado…

```
>>> print(Dic[‘chave2’])
```

Dicionários

Dic = {‘chave1’: ‘valor1’, ‘chave2’: ‘valor2’}

- Chave indexa o dicionário
  - Ao acessar o dicionário naquela chave, obtenho o valor associado…

```
>>> print(Dic[‘chave2’])
valor2
```

Dicionários

Dic = {‘chave1’: ‘valor1’, ‘chave2’: ‘valor2’}

- Pode-se criar uma nova chave:

```
>>> Dic['k3'] = 'x'
>>> print(Dic)
{'chave1': 'valor1', 'chave2': 'valor2',
'k3': 'x'}
```

Dicionários

Dic = {‘chave1’: ‘valor1’, ‘chave2’: ‘valor2’}

- Ou modificar um valor de chave existente:

```
>>> Dic['k3'] = 'valor3'
>>> print(Dic)
{'chave1': 'valor1', 'chave2': 'valor2',
'k3': 'valor3'}
```

Dicionários

Dic = {‘chave1’: ‘valor1’, ‘chave2’: ‘valor2’, ‘k3’: ‘valor3’}

- Remover elemento

```
>>> Dic.pop('chave1')
>>> print(Dic)
{'chave2': 'valor2', 'k3': 'valor3'}
```

Dicionários

Dic = {‘chave2’: ‘valor2’, ‘k3’: ‘valor3’}

- Remover último elemento

```
>>> Dic.popitem()
>>> print(Dic)
{'chave2': 'valor2'}
```

Dicionários

Dic = {‘chave2’: ‘valor2’, ‘k3’: ‘valor3’}

- Remover elemento

```
>>> del Dic['k3']

>>> print(Dic)
{'chave2': 'valor2'}
```

Dicionários

Dic = {‘chave2’: ‘valor2’, ‘k3’: ‘valor3’}

- Remover dicionário

```
>>> del Dic

>>> print(Dic)
NameError: name 'Dic' is not defined
```

Exemplo de dicionário

Dic = {}

- Composto por “chave” e “valor”
  - 'guarda-roupa': 3
  - 'televisão': 2
  - 'cadeira': 4
  - 'mesa: 1 …

Exemplo de dicionário

Dic = {}

- Composto por “chave” e “valor”
  - 'guarda-roupa': 3
  - 'televisão': 2
  - 'cadeira': 4
  - 'mesa: 1 …

```
>>> for chave in dic:
...   print(chave)
...
guarda-roupa
televisão
cadeira
mesa
```

Exemplo de dicionário

Dic = {}

- Composto por “chave” e “valor”
  - 'guarda-roupa': 3
  - 'televisão': 2
  - 'cadeira': 4
  - 'mesa: 1 …

```
>>> for chave in dic:
...   print(dic[chave])
...
3
2
4
1
```

Exemplo de dicionário

```
dic = {'guarda-roupa': 3, 'televisão': 2, 'cadeira': 4,
'mesa': 1}

dic.keys()
```

Exemplo de dicionário

```
dic = {'guarda-roupa': 3, 'televisão': 2, 'cadeira': 4,
'mesa': 1}

dic.keys()
```

```
dict_keys(['guarda-roupa', 'televisão', 'cadeira', 'mesa'])
```

Exemplo de dicionário

```
dic = {'guarda-roupa': 3, 'televisão': 2, 'cadeira': 4,
'mesa': 1}

dic.values()
```

Exemplo de dicionário

```
dic = {'guarda-roupa': 3, 'televisão': 2, 'cadeira': 4,
'mesa': 1}

dic.values()

dict_values([3, 2, 4, 1])
```

Exemplo de dicionário

```
dic = {'guarda-roupa': 3, 'televisão': 2, 'cadeira': 4,
'mesa': 1}

dic.items()
```

Devolve TUPLAS!

Exemplo de dicionário

```
dic = {'guarda-roupa': 3, 'televisão': 2, 'cadeira': 4,
'mesa': 1}

dic.items()
```

```
dict_items([('guarda-roupa', 3), ('televisão', 2),
('cadeira', 4), ('mesa', 1)])
```

Itens em ordem - Dicionários

E se eu quiser ordenar meu dicionário pelas chaves?

Um dicionário em Python garante a ordem de inserção

Itens em ordem - Dicionários

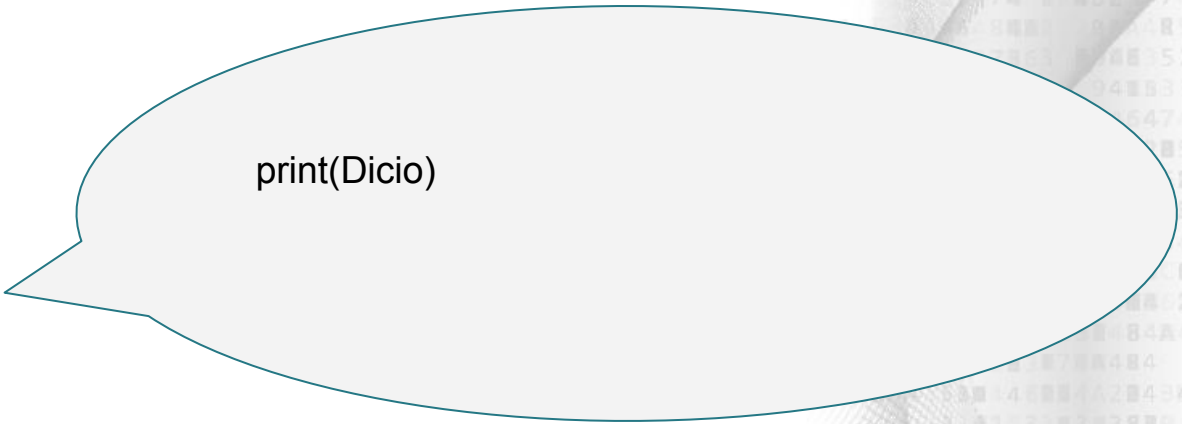
Um dicionário em Python garante a ordem de inserção

```
Dicio = {}

Dicio[2] = “b”
Dicio[1] = “a”
Dicio[3] = “c”
```

Itens em ordem - Dicionários

Um dicionário em Python garante a ordem de inserção

```
Dicio = {}

Dicio[2] = "b"
Dicio[1] = "a"
Dicio[3] = "c"
```



Itens em ordem - Dicionários

Um dicionário em Python garante a ordem de inserção

```
Dicio = {}

Dicio[2] = “b”
Dicio[1] = “a”
Dicio[3] = “c”
```

print(Dicio)

```
{2: 'b', 1: 'a', 3: 'c'}
```

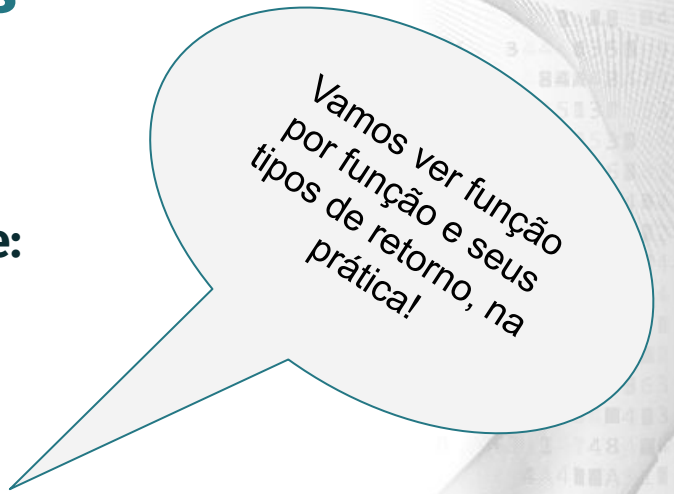
Itens em ordem - Dicionários

Para ordenar por chave:

```
Dicio = {2: 'b', 1: 'a', 3: 'c'}

Dic_ordenado = dict(sorted(Dicio.items()))
```

Itens em ordem - Dicionários

Para ordenar por chave:




```
Dicio = {2: 'b', 1: 'a', 3: 'c'}

Dic_ordenado = dict(sorted(Dicio.items()))
```

Itens em ordem - Dicionários

Para ordenar por valor:

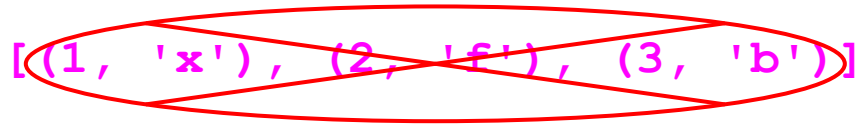
```
Dicio = {2: 'f', 1: 'x', 3: 'b'}

Lista_ordenada = sorted(Dicio.items())
```

```
[(1, 'x'), (2, 'f'), (3, 'b')]
```

Itens em ordem - Dicionários

Para ordenar por valor:

```
Dicio = {2: 'f', 1: 'x', 3: 'b'}

Lista_ordenada = sorted(Dicio.items())
```



Itens em ordem - Dicionários

Para ordenar por valor:

```
dic1 = {2: 'f', 1: 'x', 3: 'b'}
dic_ordenado = {}
chaves_ordenadas = sorted(dic1, key=dic1.get)
```

Itens em ordem - Dicionários

Para ordenar por valor:

```
dic1 = {2: 'f', 1: 'x', 3: 'b'}
dic_ordenado = {}
chaves_ordenadas = sorted(dic1, key=dic1.get)
[3, 2, 1]
```

Itens em ordem - Dicionários

Para ordenar por valor:

```
dic1 = {2: 'f', 1: 'x', 3: 'b'}
dic_ordenado = {}
chaves_ordenadas = sorted(dic1, key=dic1.get)
[3, 2, 1]
for i in chaves_ordenadas:
    dic_ordenado[i] = dic1[i]
```

Itens em ordem - Dicionários

Para ordenar por valor:

```
dic1 = {2: 'f', 1: 'x', 3: 'b'}
dic_ordenado = {}
chaves_ordenadas = sorted(d
[3, 2, 1]
for i in chaves_ordenadas:
    dic_ordenado[i] = dic1[i]
```

```
{3: 'b', 2: 'f', 1: 'x'}
```

Próximo tópico:

Matrizes

2.
Matrizes

Matrizes em Python

Enquanto um *array* é uma estrutura para armazenar dados de forma “linear”, uma matriz é um ***array bi-dimensional***.

Um **array bi-dimensional** é um *array* dentro de outro *array*.

Todas as operações feitas em um *array* podem ser feitas usando listas...

Matrizes em Python

Logo, a matriz é uma lista de listas.

Em uma matriz, cada item/dado é acessado por 2 índices:

Linha 0, Coluna 0	Linha 0, Coluna 1	Linha 0, Coluna 2
Linha 1, Coluna 0	Linha 1, Coluna 1	Linha 1, Coluna 2
Linha 2, Coluna 0	Linha 2, Coluna 1	Linha 2, Coluna 2
Linha 3, Coluna 0	Linha 3, Coluna 1	Linha 3, Coluna 2

Matrizes em Python

Declaração de uma matriz:

```
Matriz = [ [e00, e01, e02], [e10, e11, e12] ]
```

Matrizes em Python

Declaração de uma matriz:

```
Matriz = [ [e00, e01, e02], [e10, e11, e12] ]
```

Lembrem-se: cada variável tem que estar declarada e atribuída para essa construção funcionar!

Matrizes em Python

Declaração de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]
```

Matrizes em Python

Declaração de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]
```

Como acessar os elementos?

Matrizes em Python

Elementos de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[0])
```

Matrizes em Python

Elementos de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[0]) # imprime os elementos da linha 0

[0, 1, 2]
```

Matrizes em Python

Elementos de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[1])
```

Matrizes em Python

Elementos de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[1]) # imprime os elementos da linha 1

[3, 4, 5]
```

Matrizes em Python

Acesso a um elemento específico de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[1][1])
```

Matrizes em Python

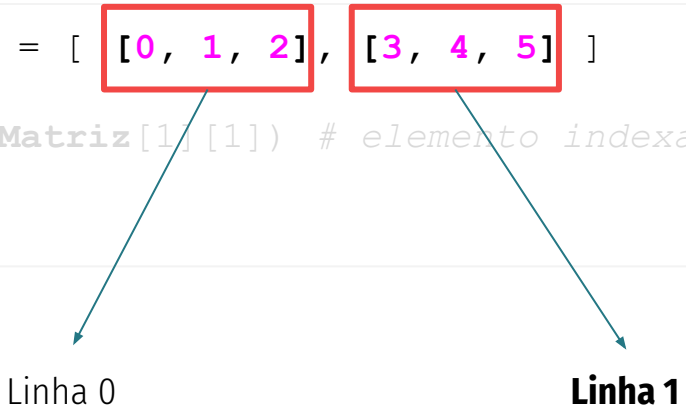
Acesso a um elemento específico de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[1][1]) # elemento indexado pela "linha 1", "coluna 1"

4
```

Matrizes em Python

Acesso a um elemento específico de uma matriz:

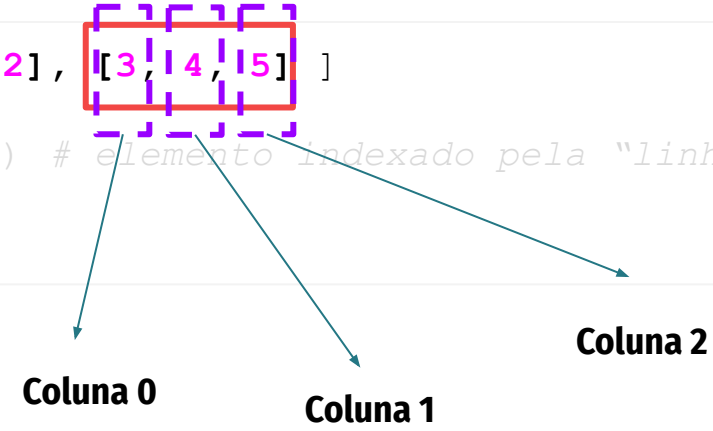
```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[1][1]) # elemento indexado pela "linha 1", "coluna 1"

4
```



Matrizes em Python

Acesso a um elemento específico de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz[1][1]) # elemento indexado pela "linha 1", "coluna 1"

4
```



Matrizes em Python

Impressão de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(Matriz)

[[0, 1, 2], [3, 4, 5]]
```

Matrizes em Python

Tipo de uma matriz:

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

print(type(Matriz))

<class 'list'>
```

Inserção em Matriz

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz.append([6, 7, 8])

print(Matriz)
```

Inserção em Matriz

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz.append([6, 7, 8])

print(Matriz)

[[0, 1, 2], [3, 4, 5], [6, 7, 8]]
```

Inserção posicional em Matriz

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz.insert(1, [6, 7, 8])

print(Matriz)
```

Inserção posicional em Matriz

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz.insert(1, [6, 7, 8])

print(Matriz)

[[0, 1, 2],  [6, 7, 8], [3, 4, 5]]
```

Inserção posicional em Matriz

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz.insert(0, [6, 7, 8])

print(Matriz)
```

Inserção posicional em Matriz

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz.insert(0, [6, 7, 8])

print(Matriz)

[[6, 7, 8], [0, 1, 2], [3, 4, 5]]
```

Atualizando uma Matriz

Atualiza uma linha inteira (troca lista)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz[0] = [-3, -2, -1]
```

Atualizando uma Matriz

Atualiza uma linha inteira (troca lista)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz[0] = [-3, -2, -1]

print(Matriz)

[[-3, -2, -1], [3, 4, 5] ]
```

Atualizando uma Matriz

Atualiza um elemento (substitui dado)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Matriz[0][2] = -3
```

Atualizando uma Matriz

Atualiza um elemento (substitui dado)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

        Matriz[0][2] = -3
```

Atualizando uma Matriz

Atualiza um elemento (substitui dado)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

        Matriz[0][2] = -3
```

Atualizando uma Matriz

Atualiza um elemento (substitui dado)

```
Matriz = [ [0, 1, -3], [3, 4, 5] ]

        Matriz[0][2] = -3
```

Removendo valores de uma Matriz

Remove uma linha (apaga uma lista)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

del(Matriz[1])
```

Removendo valores de uma Matriz

Remove uma linha (apaga uma lista)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

del(Matriz[1])
```

Removendo valores de uma Matriz

Remove uma linha (apaga uma lista)

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

del(Matriz[1])

print(Matriz)

[[0, 1, 2]]
```

Tamanho de uma Matriz

```
Matriz = [ [0, 1, 2], [3, 4, 5] ]

Tamanho = len(Matriz)
```

print(Tamanho)

2

Fatiamento de uma Matriz

Mostra um pedaço

```
Matriz = [ [0, 1, 2], [3, 4, 5], [6, 7, 8] ]

Matriz2 = Matriz[1:5]
```

Fatiamento de uma Matriz

Mostra um pedaço

```
Matriz = [ [0, 1, 2], [3, 4, 5], [6, 7, 8] ]

Matriz2 = Matriz[2:]
```

Fatiamento de uma Matriz

Mostra um pedaço

```
Matriz = [ [0, 1, 2], [3, 4, 5], [6, 7, 8] ]

Matriz2 = Matriz[:1]
```

Cópia de uma Matriz

M1 = [[1, 1], [2, 2]]

M2 = M1

M1[0] = [0, 0]

M2?

Cópia de uma Matriz

import copy

M1 = [[1, 1], [2, 2]]

M2 = copy.copy(M1)

M1[0] = [0, 0]

M2?

Cópia de uma Matriz

```
import copy
M1 = [ [1, 1], [2, 2] ]

M2 = copy.copy(M1)

M1[0][0] = 0
```

M2?

Cópia de uma Matriz

```
import copy
M1 = [ [1, 1], [2, 2] ]

M2 = copy.deepcopy(M1)

M1[0][0] = 0
```

M2?

Teste seus conhecimentos

Exercício

1. Considere um arquivo de texto a ser passado pelo usuário via argumento na linha de comando:
   a. Crie um contador de palavras onde cada palavra é uma chave em um dicionário e a frequência de ocorrência é o valor a se armazenar na chave.
   b. Crie um contador de caracteres (todos, incluindo sinais de pontuação e espaço) que mostre, ao final, o número total de caracteres do texto e a frequência individual de cada um (use um dicionário).
   c. Imprima o dicionário de palavras em ordem alfabética inversa (de Z para A)
   d. Imprima o dicionário de caracteres em ordem alfabética.

Exercício

2. Escreva um programa que:
   a. Receba uma matriz passada pelo usuário linha por linha;
   b. Mostre a matriz de maneira organizada (linhas e colunas)
   c. Conte a quantidade total de elementos contidos em uma matriz bi-dimensional baseada em listas e apresente o valor da conta.

Obrigado!

Bons Estudos!!!