

PYTHON PARA INICIANTEs

MANOELA KOHLER

prof.Manoela@ica.ele.puc-rio.br
www.linkedin.com/in/manoelakohler

RECAP

First Things First:

- Filosofia
- Ferramentas para análise de dados
- Características
- IDEs

Introdução ao Python

- Conceitos básicos
- Criação de ambientes
- Variáveis
- Tipos de dados
- Operações
- Estruturas condicionais
- Estruturas de repetição
- Bibliotecas: instalação e utilização
- Funções

PROGRAMA

Introdução à Python – Parte II:

- Pacotes
- Manipulação de Dados:
 - Numpy

Pacotes

PACOTES

Importa todos os módulos dentro do pacote

```
>> import <nome_do_pacote>
```

Importa um módulo ou função dentro do pacote. Assim posso chamar diretamente pelo nome, sem a necessidade de chamar também o nome do pacote.

```
>> from <nome_do_pacote> import <nome>
```

Importa todas as funções e módulos dentro do pacote e assim como no caso anterior, é possível chama-los pelo nome sem ter que referenciar o nome do pacote.


```
>> from <nome_do_pacote> import *
```

Importa a função nome com um nome alternativo qualquer.

```
>> from <nome_do_pacote> import <nome> as <nome_2>
```

EXERCÍCIOS

1. Importe a classe *pyplot* do pacote *matplotlib*. A classe *pyplot* contém a função a ser usada: *hist()*, capaz de gerar um histograma.
2. Utilize as amostras de diferentes tamanhos geradas com *np.random* e crie um histograma a partir de cada uma dessas amostras.
3. Avalie o gráfico gerado.



Manipulação de Dados



NumPy

O NumPy é o pacote básico fundamental da linguagem Python para computação científica, ele provê muitas das estruturas básicas, fornecendo ferramentas para integração, comunicação com Fortran e C, vetorização, álgebra linear, geração de números aleatórios.

NumPy constitui uma fundação sólida para muitas ferramentas aplicadas em análise de dados.

O que torna o NumPy tão popular é sua eficiência quando comparado às listas. → É MUITO MAIS RÁPIDO TRABALHAR COM NUMPY. O pacote permite o acesso de dados de modo muito mais eficiente.

ESTRUTURAS DE DADOS

Vetores

VETORES

Elementos de um mesmo tipo.

0	1	2	3	4	5	6	7	8	9
50	34	45	68	89	8	445	78	9	7895

Vetor Numérico em Python

ESTRUTURAS DE DADOS

Matrizes

MATRIZES

	1	2	3	4	5
1	50	34	45	68	89
2	5400	65	2	565	9
3	3	43	658	76	8

ESCALARES, VETORES, MATRIZES, TENSORES...

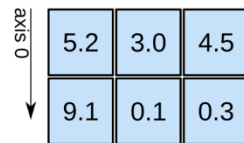
1D array



axis 0 →

shape: (4,)

2D array

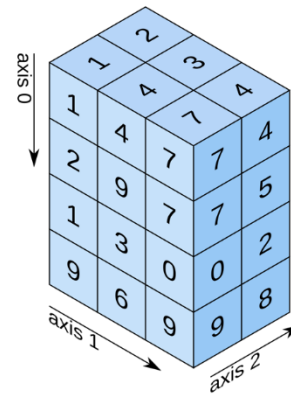


axis 0 ↓

axis 1 →

shape: (2, 3)

3D array



shape: (4, 3, 2)

ESCALARES, VETORES, MATRIZES, TENSORES...

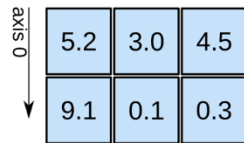
1D array



axis 0 →

shape: (4,)

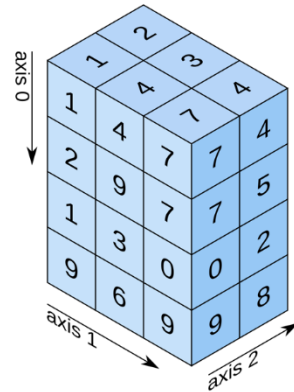
2D array



axis 1 →

shape: (2, 3)

3D array



shape: (4, 3, 2)

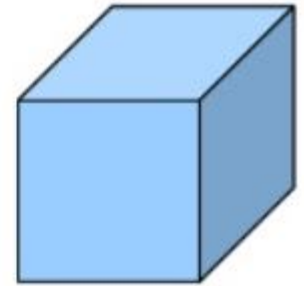
1d-tensor



2d-tensor



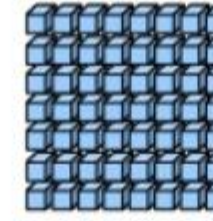
3d-tensor



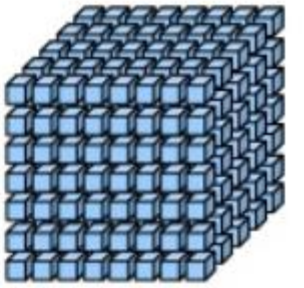
4d-tensor



5d-tensor



6d-tensor



ESCALARES, VETORES, MATRIZES, TENSORES...

Importar pacote NumPy:

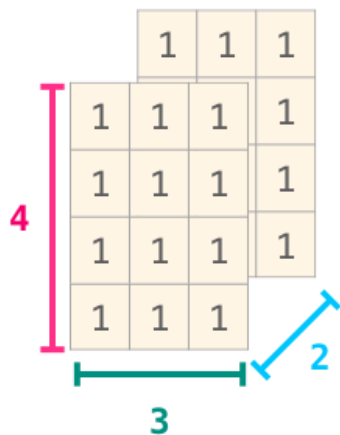
```
IMPORT NUMPY AS NP
```

criação de vetores, matrizes e tensores

Tensor de 3 dimensões inicializado com '1's

```
>> t1 = np.ones(dim1, dim2, dim3)
```

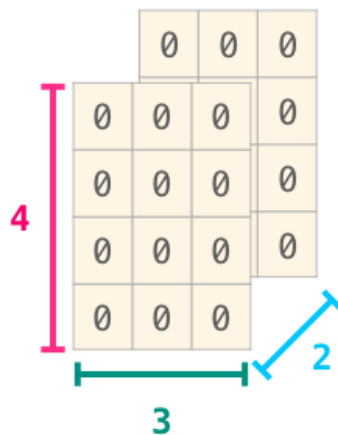
`np.ones((4,3,2))`



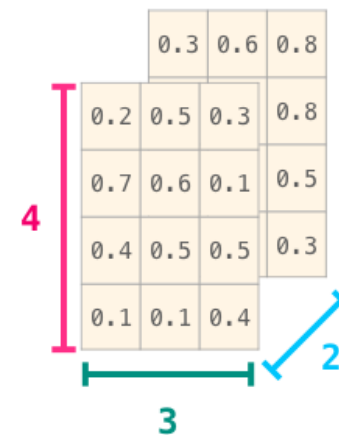
Shape da estrutura:

```
>> t1.shape  
(4, 3, 2)
```

`np.zeros((4,3,2))`



`np.random.random((4,3,2))`

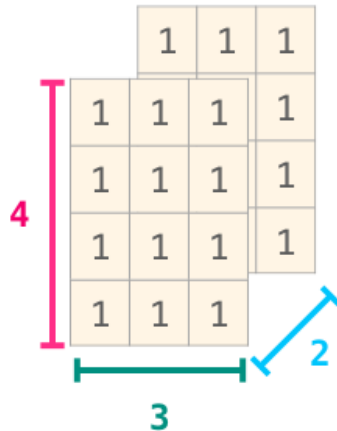


criação de vetores, matrizes e tensores

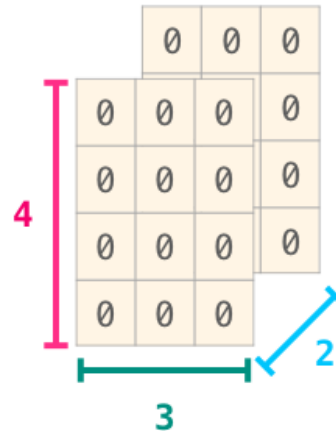
Tensor de 3 dimensões inicializado com '0's

```
>> t2 = np.zeros(dim1, dim2, dim3)
```

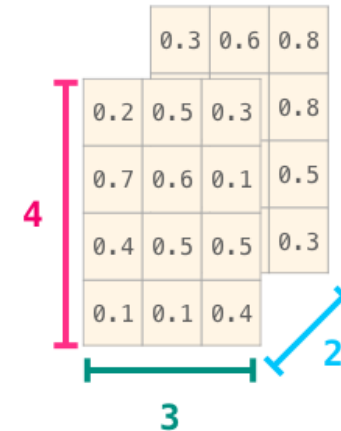
`np.ones((4,3,2))`



`np.zeros((4,3,2))`



`np.random.random((4,3,2))`



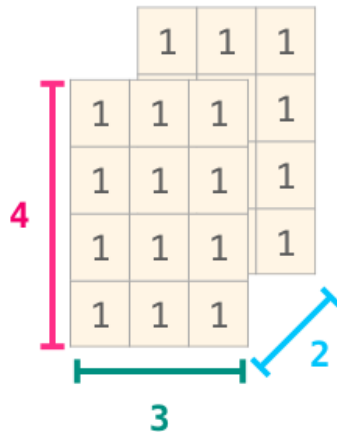
Shape da estrutura:

```
>> t2.shape  
(4, 3, 2)
```

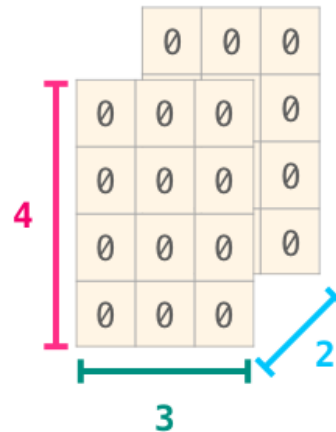
criação de vetores, matrizes e tensores

Tensor de 3 dimensões inicializado aleatoriamente no intervalo de 0 a 1

```
np.ones((4,3,2))
```

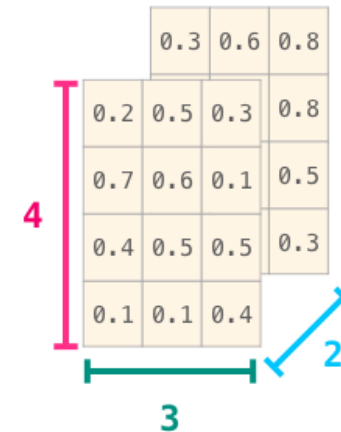


```
np.zeros((4,3,2))
```



```
>> t3 = np.random.random(dim1,  
                           dim2, dim3)
```

```
np.random.random((4,3,2))
```



Shape da estrutura:

```
>> t3.shape  
(4, 3, 2)
```

criação de vetores, matrizes e tensores

Inicializar estrutura com valores definidos pelo usuário:

```
>> np.array()
```

```
In [1]: 1 import numpy as np
```

```
In [9]: 1 X = np.array([[135, 30], [57, 15], [150, 35]])  
        2 X
```

```
Out[9]: array([[135, 30],  
               [ 57, 15],  
               [150, 35]])
```

Shape da estrutura:

```
>> X.shape  
(3, 2)
```

criação de vetores, matrizes e tensores

Inicializar estrutura com valores definidos pelo usuário:

```
>> np.array([1, 2, 3, 4, 5, 6])  
[1, 2, 3, 4, 5, 6]
```

ATRIBUTOS DE UM NUMPY ARRAY

Attribute	What it records
<code>shape</code>	The dimensions of the NumPy array
<code>size</code>	The total number of elements in the NumPy array
<code>ndim</code>	The number of dimensions of the array
<code>dtype</code>	The data type of the elements in the array
<code>itemsize</code>	The length of a single array element in bytes

FUNÇÕES PARA ARRAYS

data

1	2
3	4
5	6

`.max()` = 6

data

1	2
3	4
5	6

`.min()` = 1

data

1	2
3	4
5	6

`.sum()` = 21

OPERAÇÕES EM NUMPY ARRAYS

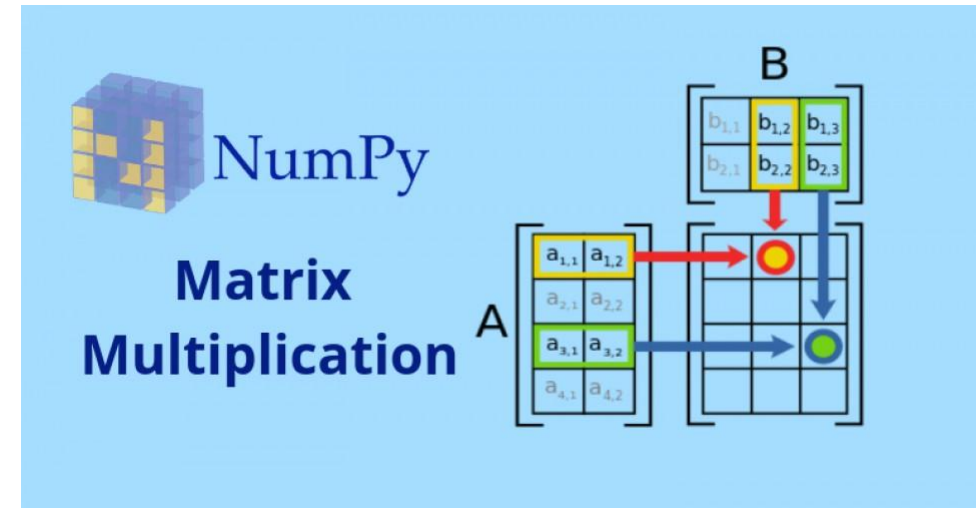
Operações:

A:

1	2
4	5

B:

10	20
30	40



Scalar Multiplication:

```
>>> 2 * A
```

2	4
8	10

Element-Wise Multiplication:

```
>>> A * B
```

10	40
120	200

Matrix Multiplication:

```
>>> A.dot(B)
```

70	100
190	280

EXERCÍCIO

1. Operações vetorizadas são extremamente mais eficientes do que a utilização de loops para realização de tais operações. Faça um teste para provar a eficiência da biblioteca em relação ao uso de listas nativas do Python. Crie dois vetores com 100000 elementos e faça um 'dot product'. Dica: use a função `time.process_time()` do pacote 'time' para salvar o tempo antes e depois da operação. O tempo decorrido em segundos pode ser calculado com a diferença entre estes valores.

Transformação de uma lista em um numpy array:

```
a = np.asarray(a)
```

$$a \cdot b = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix}_{(1 \times n)} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}_{(n \times 1)} = \left\{ a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + a_5b_5 \right\}$$

Dot Product

EXERCÍCIO: ANÁLISE DE DEMONSTRAÇÃO FINANCEIRA

Cenário: Você é um cientista de dados trabalhando para uma empresa de consultoria. Um de seus colegas do departamento de auditoria pediu sua ajuda para avaliar o demonstrativo financeiro de uma organização X.

```
receitas = np.array([14574.49, 7606.46, 18611.41, 19175.41, 8758.65, 8105.44, 11496.28, 9766.09, 10305.32, 18379.96, 10713.97, 15433.50])  
custos = np.array([12051.82, 5695.07, 12319.20, 12089.72, 8658.57, 840.20, 3285.73, 5821.12, 6976.93, 16618.61, 10054.37, 3803.96])
```

EXERCÍCIO: ANÁLISE DE DEMONSTRAÇÃO FINANCEIRA

Você recebeu dois vetores de dados: receita mensal e despesa mensal para o ano em questão. Seu trabalho é calcular as seguintes métricas financeiras:

- lucro de cada mês;
- lucro depois de descontado imposto para cada mês (imposto é de 30% - sobre o lucro)
- margem de lucro de cada mês (lucro depois de descontado o imposto dividido pela receita)
- meses 'bons' (onde o lucro depois da taxa  o foi maior que o lucro m  dio do ano) (dica: fun  o 'np.where')
- meses 'ruins' (contr  rio de meses 'bons')
- o melhor m  s (descontado imposto)
- o pior m  s (descontado imposto)

Fun  es que podem ser   teis:
mean(); max(); min(); np.where()