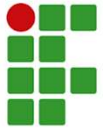
Bibliotecas em Python

NumPy



INSTITUTO FEDERAL
Sudeste de Minas Gerais

Arrays

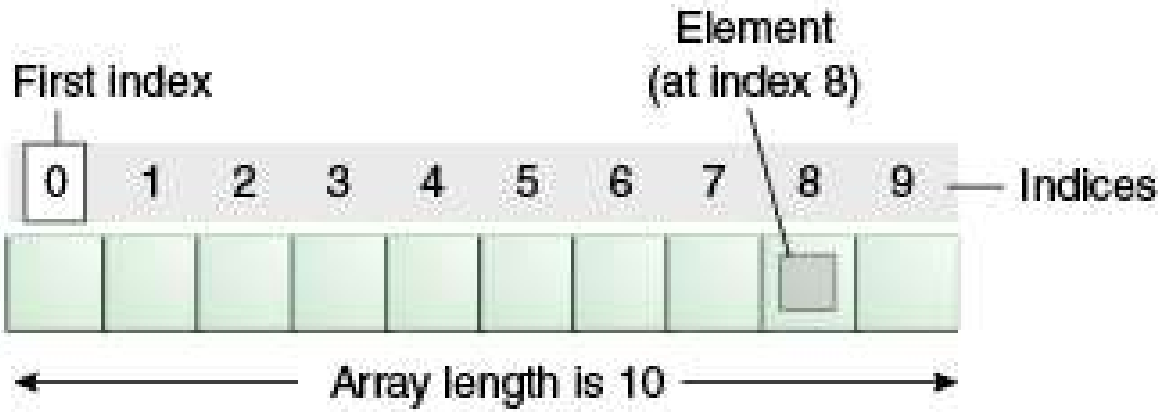
- Nativamente, a linguagem Python não possui arrays.
- Arrays no âmbito computacional são estruturas de dados semelhantes às listas do Python, mas não tão flexíveis.
- Em um array todos os elementos devem ser de um mesmo tipo, tipicamente numérico, como int ou float.
- Além disso, o tamanho de um array não pode ser modificado, ao contrário de listas que podem crescer dinamicamente.
- Em contrapartida, o uso de arrays é muito mais eficiente e facilita a computação de grandes volumes de dados numéricos.
- Isso faz com que arrays sejam particularmente úteis em computação científica.

Arrays

- Arrays são úteis para armazenar:
  - Valores de um experimento/simulação em tempos discretos.
  - Sinal gravado por um equipamento de medição, por exemplo, ondas sonoras.
  - Pixels de uma imagem, escala de cinza ou coloridos.
  - Dados tridimensionais medidos em posições X,Y,Z diferentes, ou até mesmo imagens de uma ressonância magnética, etc…

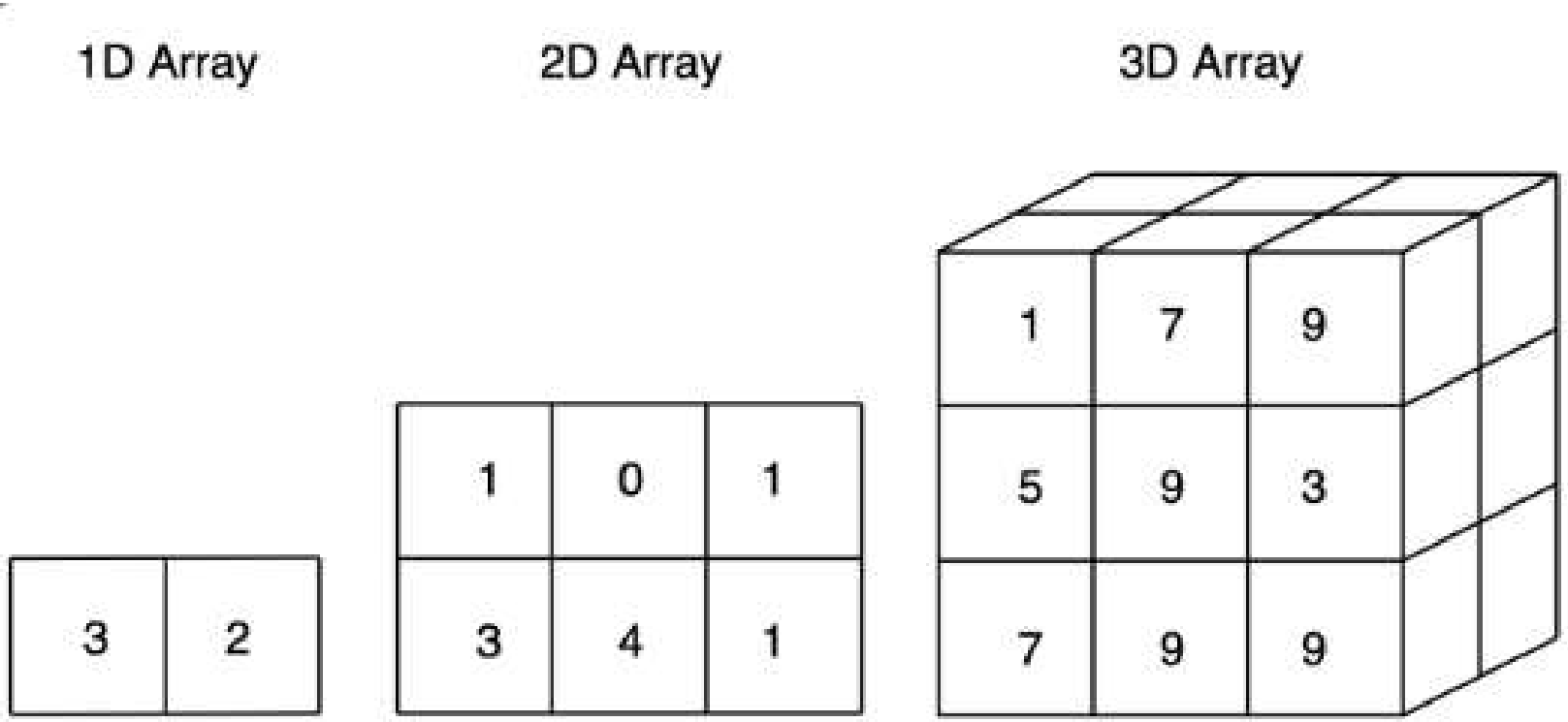
Arrays

- Visualmente, arrays podem ser representados como:



An array of 10 elements.

Arrays



INSTITUTO FEDERAL
Sudeste de Minas Gerais

Numpy

- Numpy é uma biblioteca (ou pacote) Python que nos fornece ferramentas robustas para realização de cálculos numéricos em arrays multidimensionais.
- O propósito dessa biblioteca é realizar cálculos com uma sintaxe simples e grande desempenho, visto que, grande parte da biblioteca é implementada em C/C++ que por trás dos panos funciona com uma linguagem mais adequada para otimizações (mas não entraremos nesse mérito durante o curso).
- Por que utilizar NumPy:
    - Fornece ferramentas adicionais para manipulação de arrays multidimensionais.
    - Focado em desempenho e projetado para computação científica.

Numpy

- Numpy é um pacote para operações com matrizes.
- O Numpy apresenta centenas de funções matemáticas.
- Dessa forma, o Numpy é a base para a computação numérica no Python.
- Durante o desenvolvimento dos modelos de aprendizagem de máquina, muitas vezes, necessitamos utilizar algumas funções do pacote para preparar dados para ser usados em outras funções.

Numpy

- Veja abaixo um código que utiliza a biblioteca Numpy.

```
 8 #importar a biblioteca numpy como np
 9 import numpy as np
10
11 #cria uma matriz 3 x 3.
12 m1 = np.array([[1,6,8], [2,7,2],[1,4,10]])
13
14 #cria uma matriz de 1 x 3
15 m2 =  np.array( [[1, 2, 5]] )
16
17 #criar uma matriz de zeros
18 zeros  = np.zeros((3,3))  #deve entrar com o shape(que deve ser uma tupla)
19
20 #criar uma matriz de uns.
21 ones = np.ones((3,3))
22
23 #criar uma matriz de uma dimensão com valores uniformemente distribuidos
24 x = np.arange(1, 12, 0.5) #inicio, fim e passo
```

Numpy

- As matrizes criadas (objetos) apresentam vários atributos. Veja abaixo um exemplo de alguns.

```
26 #O objeto apresenta vários atributos, veja abaixo
27 print m1.shape #dimensões da matriz
28 print m1.size  #numero de elementos da matriz
29 print m1.dtype #tipo do dado, podemos converter de int32 em outros tipos.
30 print m1.max() #veja no simbolo é amarelo e é um f(função), então temos que colocar parênteses
31
32 print m2.ndim, x.ndim #veja que as dimensões das m2 é diferente da x. cuidado com isso!!!
```

Numpy

- Podemos acessar os elementos das matrizes de várias formas, veja abaixo.

```
34 #acessar um elemento de uma matriz
35 print m1[0,0]
36 print m1[1,1]
37
38 #fatiar a matriz.
39 print m1[0:2,:] #linha 0 e 1 e todas as colunas
40
41 #fazer loop sobre os elementos da matriz
42 for linha in range(3):
43     for coluna in range(3):
44         print m1[linha,coluna]
```

Numpy

- Várias funções podem ser utilizadas para manipulação das matrizes. Veja alguns exemplos no código abaixo.

```
46 #transforma (reshape) a matriz 3 x 3 em 9 x 1
47 re_m1 = m1.reshape((9,1))
48
49 #transforma a matriz em uma dimensão
50 onedin = m1.flatten()
51
52 #multiplicar um escalar por uma matriz
53 escalar = 2 * m2
54
55 #elevar a matriz a um escalar
56 elevar = m1**2
57
58 #seno dos elementos
59 seno = np.sin(m1) #elementos em radiano
60
61 #multiplicar uma matriz por uma matriz
62 mx = np.dot(m2, m1)
63
64 #transposta de uma matriz
65 trans = np.transpose(m1)
66 trans = m1.T
67
68 #inversa
69 inv = np.linalg.inv(m1)
70
71 #determinante
72 det = np.linalg.det(m1
```

INSTITUTO FEDERAL
Sudeste de Minas Gerais