Advanced Institute for Artificial Intelligence

Programação em Python

https://advancedinstitute.ai

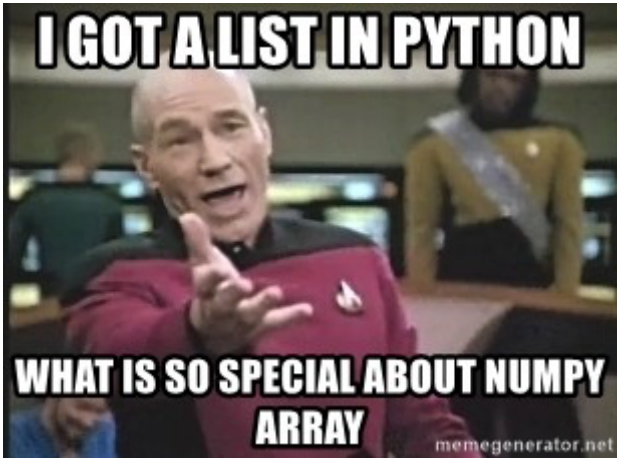
Programação Python

Operações Matriciais com o ***Numpy***

Referências e Fontes das Imagens

- NumPy Illustrated: The Visual Guide to NumPy
- Parallel and High Performance Computing (Book)
- Learning Numpy Array (Book)

The Elephant in the Room



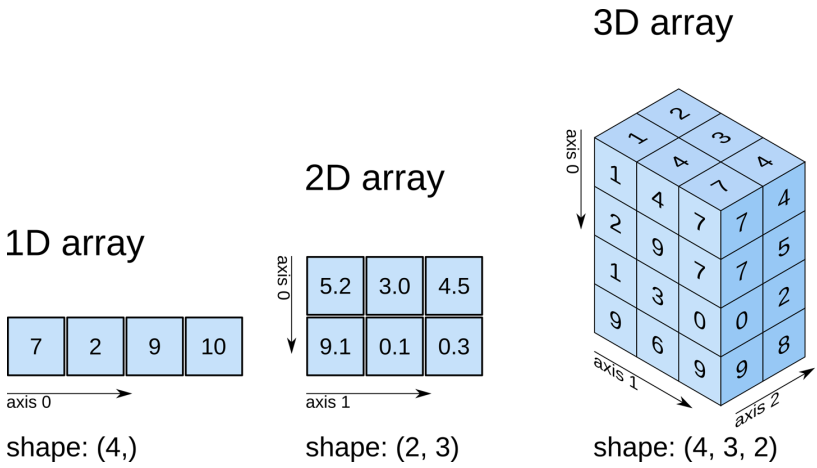
Introdução

- Originalmente como parte do *SciPy* e depois foi destacado como uma **biblioteca fundamental**
- **Código é mais limpo** do que o código Python "direto" que tenta realizar a mesma tarefa
- **Menos *loops* necessários**, porque as operações **funcionam diretamente em *arrays* e matrizes**;

Introdução Cont.

- **Funções matemáticas** tornam a vida mais fácil;
- Algoritmos subjacentes foram projetados com **alto desempenho** em mente;
- Os arrays do NumPy são **armazenados de forma mais eficiente**
- Grandes partes do NumPy são **escritas em C**. Isso torna o NumPy **mais rápido do que o código Python puro**

Armazenamento Otimizado em Memória



Operações diretas

- Soma de vetores (utilizando listas):

```
1  for i in range(len(a)):
2      a[i] = i ** 2
3      b[i] = i ** 3
4      c.append(a[i] + b[i])
```

- Utilizando *Numpy Arrays*

```
1  c = a + b
```

Arrays N-dimensionais

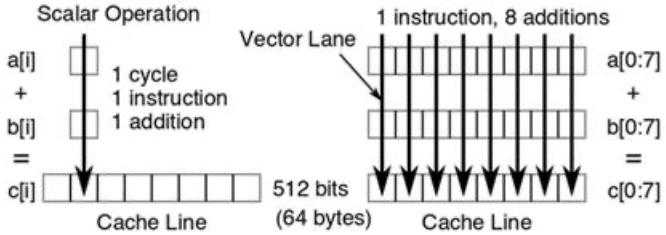
- Arrays de propósito geral e homogêneos;
  - Mais fácil determinar o tamanho de armazenamento necessário;
- Conseguem **vetorizar** operações;
- O número de dimensões e itens em um array é definido por sua forma (atributo `shape`
  - Tupla de N inteiros não negativos que especificam os tamanhos de cada dimensão.
- Tipos especiais associados aos elementos (`dtype`):
  - `numpy.intc` (`numpy.int32`), `numpy.int_` (`numpy.int64`), `numpy.single` (`numpy.float32`), `numpy.str_`, `numpy.bool_`

Arrays N-dimensionais Cont.

- `ndarrays` podem compartilhar dados, de modo que as alterações feitas em um podem ser visíveis em outro;
  - Um ndarray pode ser uma *view* para outro, i.e., mostrando somente linhas e colunas selecionadas
- Segmento unidimensional contíguo de memória de computador;
  - Esquema de indexação que mapeia N inteiros para a localização de um item no bloco
  - `Stride` (*offset)*
- Múltiplas formas de inicialização

Vetorização de operações

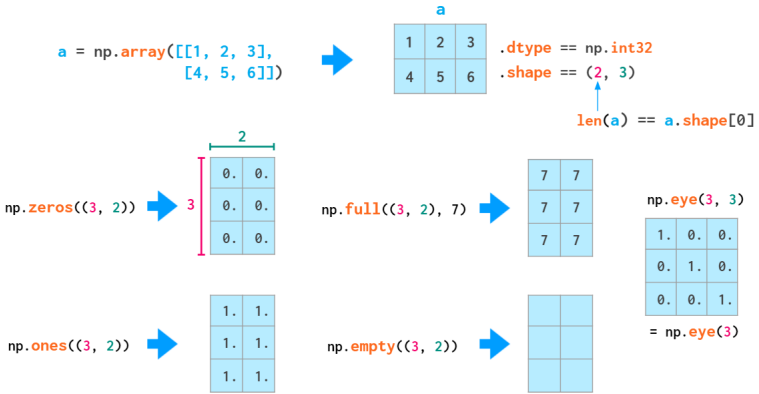
- Paradigma ***S**ingle **S**truction **M**ultiple **D**ata* - SIMD
  - Forma de **paralelismo de dados**



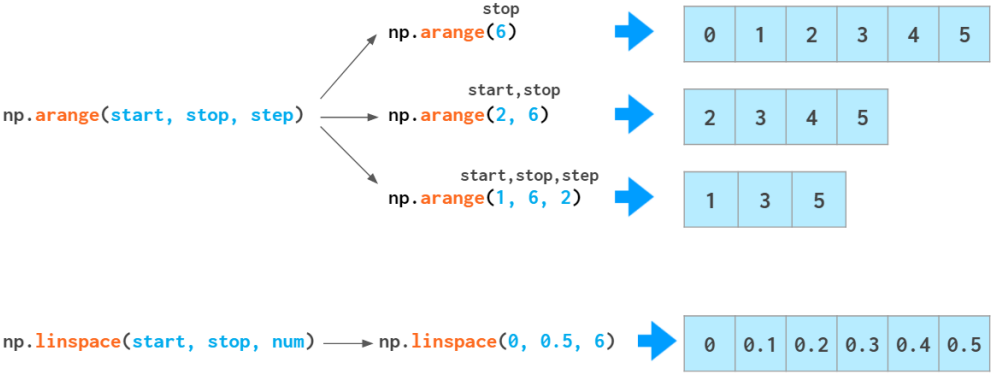
Criação de `ndarrays`

- **`numpy.array`** - Converte outro objeto (list,tupla, etc) para **`ndarray`**
- **`numpy.empty`** - constrói um **`ndarray`** sem inicializar seus elementos;
- **`numpy.eye`** - constrói um **`ndarray`** com valores *1* na sua diagonal (identidade)
- **`numpy.ones`**- cria matriz de tamanho n apenas com valores 1
- **`numpy.zeros`** - cria **`ndarray`** de tamanho *n* apenas com valores 0
- **`numpy.full`** - cria **`ndarray`** preenchido com valor específico
- Criação a partir de um protótipo: **`numpy.ones_like, numpy.zeros_like, numpy.empty_like, numpyt.full_like`**

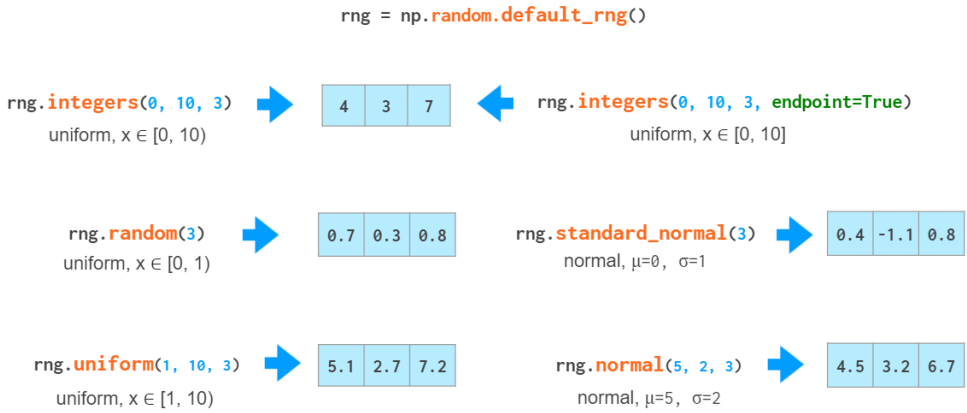
Criação de ndarrays



Inicialização monotônica de `ndarrays`

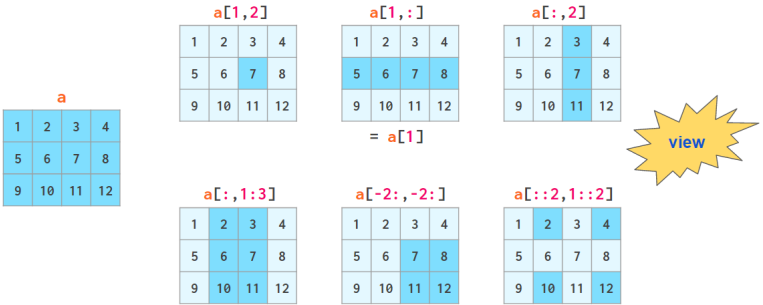


Inicialização aleatória de `ndarrays`

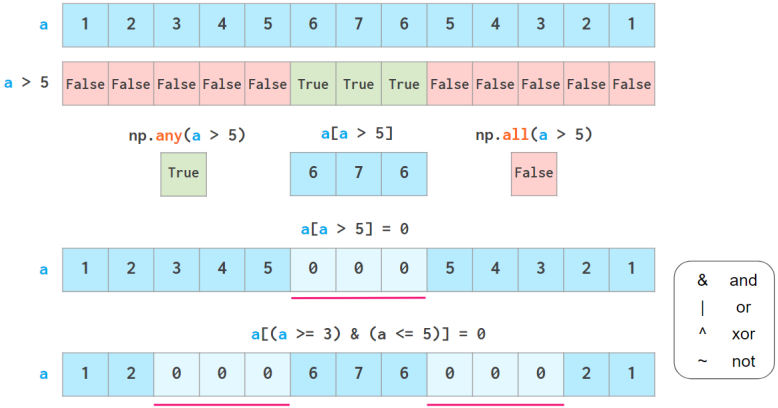


Slicing

- Para o caso de *arrays* NumPy unidimensionais funciona exatamente como o *slicing* de listas Python
- Sintaxe similar para mais de uma dimensão



Indexação utilizando booleanos



Propriedades do `ndarray`

- `T`: matriz transposta
- `dtype`: tipo de dados armazenado na matriz
- `size`: quantidade de elementos na matriz
- `shape`: dimensões da matriz
- `ndim`: número de dimensões da matriz
- `nbytes`: número de *bytes* utilizados para armazenar a matriz em memória
- `imag`: a parte imaginária dos elementos (complexos) da matriz
- `real`: a parte real dos elementos (complexos) armazenados na matriz

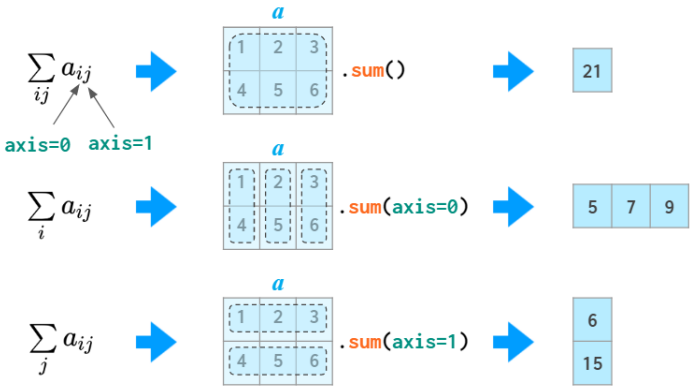
Algumas operações interessantes

- `max()`: retorna o maior elemento da matriz
- `min()`: retorna o menor elemento da matriz
- `sum()`: retorna a soma dos elementos da matriz
- `mean()`: retorna a média dos elementos da matriz
- `var()`: retorna a variância dos elementos da matriz
- `std`: retorna o desvio padrão dos elementos na matriz
- `prod()`: retorna o produto dos elementos da matriz
- `nonzero()`: retorna os elementos da matriz cujos valores são diferentes de zero

Eixos

- *axis* são definidos para matrizes com mais de uma dimensão
- O número do índice em questão: O primeiro índice é eixo $= 0$, o segundo é eixo $= 1$ e assim por diante;
- Uma matriz bidimensional possui dois eixos correspondentes:
  - O primeiro correndo verticalmente para baixo nas linhas (eixo 0)
  - O segundo correndo horizontalmente nas colunas (eixo 1)

Eixos


$\sum_{ij} a_{ij}$
axis=0
axis=1
a
1 2 3
4 5 6
.sum()
21
$\sum_{i} a_{ij}$
.sum(axis=0)
5 7 9
$\sum_{j} a_{ij}$
.sum(axis=1)
6
15

Alterando o formato de um `ndarray`

- Operação **`reshape()`**: criação de novo **`ndarray`** com novo shape.
  - Para versão *inplace*, utilizar **`resize()`**

```
>>> a = np.arange(1,10)
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> a.shape
(9,)
>>> a.reshape(3,3)
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Dúvidas?