Computação II

MAB 225 - EE2/ET2/ER2

SciPy & NumPy

Brunno Goldstein

bfgoldstein@cos.ufrj.br

www.lam.ufrj.br/~bfgoldstein

Ementa

- Programação Orientada a Objetos
- Tratamento de Exceções
- Módulos
- Manipulação de Arquivos
- Interface Gráfica (Tkinter)
- Biblioteca Numérica (Numpy)

Ementa

- Programação Orientada a Objetos
- Tratamento de Exceções
- Módulos
- Manipulação de Arquivos
- Interface Gráfica (Tkinter)
- Biblioteca Numérica (Numpy)

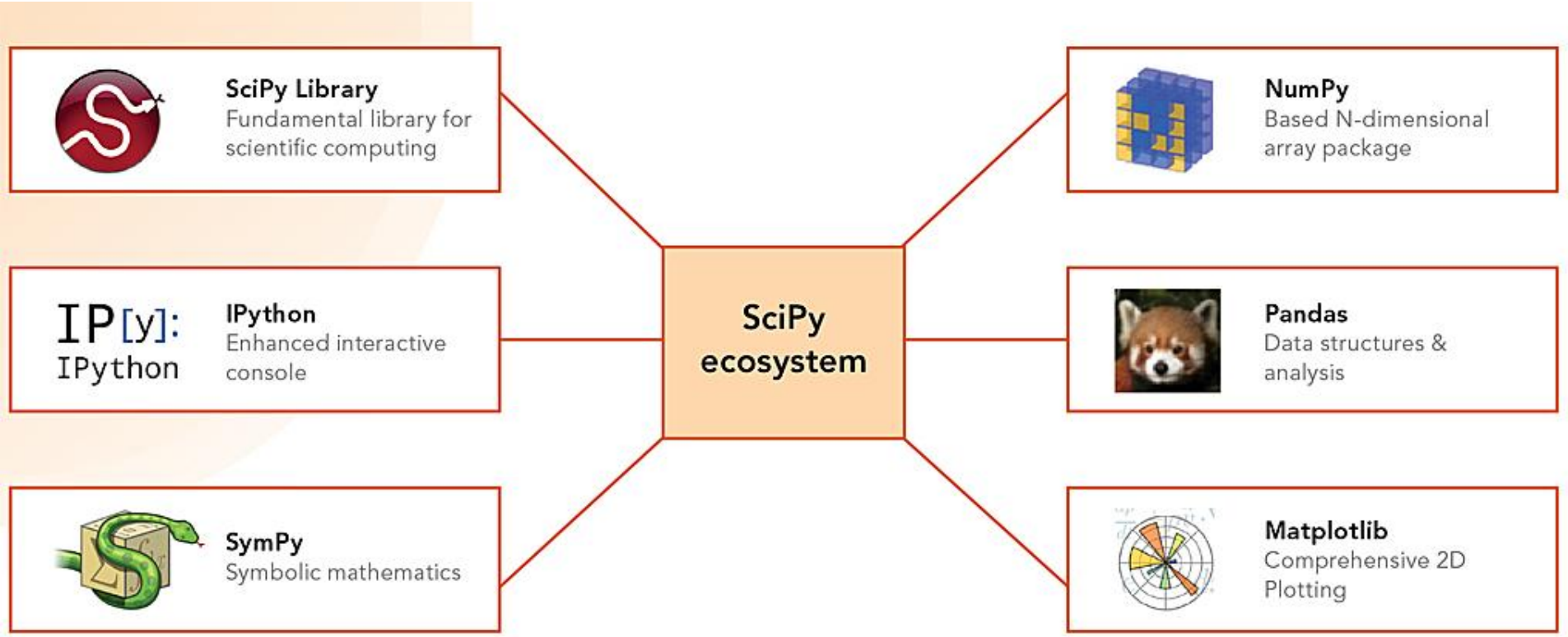
Ementa

- Programação Orientada a Objetos
- Tratamento de Exceções
- Módulos
- Manipulação de Arquivos
- Interface Gráfica (Tkinter)
- Biblioteca Numérica (Numpy)

SciPy

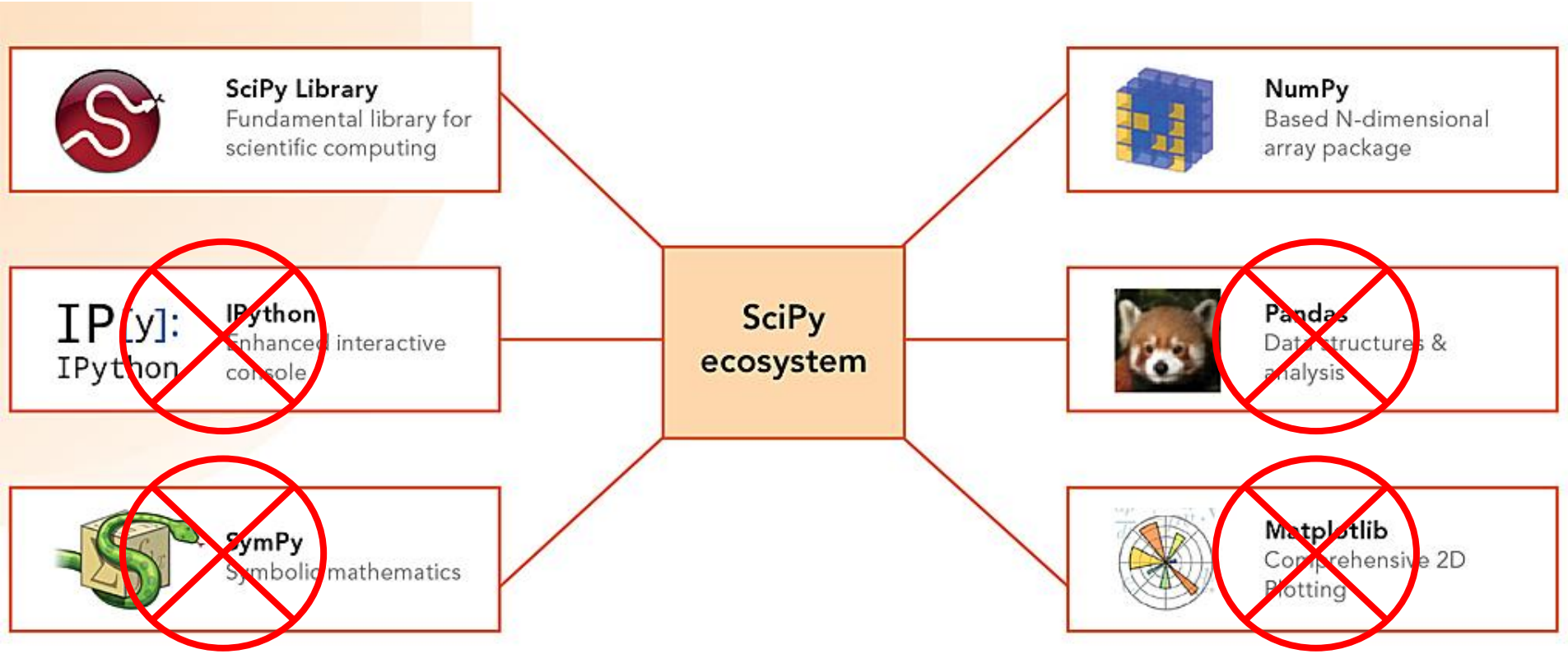
- Conjunto de bibliotecas - Stack;
- Dentre as bibliotecas:
  - SciPy - Biblioteca para computação científica;
  - NumPy - Biblioteca para computação científica;
  - Matplotlib - Biblioteca para criação de gráficos 2D.
- Referências https://www.scipy.org/
- Download: http://www.scipy.org/install.html

SciPY



Fonte: http://www.esri.com/~/media/Images/Content/news/arcuser/0115/scipy_2-lg.jpg

SciPY



⊗ Não vamos ver

Fonte: http://www.esri.com/~/media/Images/Content/news/arcuser/0115/scipy_2-lg.jpg

Biblioteca SciPy

- Biblioteca de computação científica;
- Possui diversos pacotes como:
  - Special functions (**scipy.special**)
  - Integration (**scipy.integrate**)
  - Optimization (**scipy.optimize**)
  - Interpolation (**scipy.interpolate**)
  - Fourier Transforms (**scipy.fftpack**)
  - Signal Processing (**scipy.signal**)
  - Linear Algebra (**scipy.linalg**)

Biblioteca SciPy

- Biblioteca de computação científica;
- Possui diversos pacotes como:
  - Special functions (**scipy.special**)
  - Integration (**scipy.integrate**)
  - Optimization (**scipy.optimize**)
  - Interpolation (**scipy.interpolate**)
  - Fourier Transforms (**scipy.fftpack**)
  - Signal Processing (**scipy.signal**)
  - Linear Algebra (**scipy.linalg**)

Biblioteca Numpy

- Biblioteca de computação científica;

- Responsável por:
  - Tipos de dados - arrays (listas), matrizes, etc;
  - Operações com esses tipos - Indexação, ordenação, etc.

- Possui também funções de Álgebra Linear (contidas na biblioteca SciPy).

Biblioteca Numpy

- Numpy manipula uma estrutura especial de dados;

- Classe **ndarray** ou apenas **array:**
  - Representa a estrutura fornecida pelo Numpy
  - Estrutura parecida com as Listas;
  - Classe possui diversos atributos e funções;

Biblioteca Numpy

- Alguns atributos da classe **ndarray**:
  - ndarray.ndim - Numero de dimensões do array
    - Ex.: Matriz m x n poussi ndim = 2
  - ndarray.shape - Tupla com o tamanho dos array que representa cada dimensão
    - Ex.: Matrix m x n possui shape (m,n)
  - ndarray.size - Quantidade de elementos no array
    - Ex.: Matrix m x n possui size = m * n
  - ndarray.dtype - Tipo de dados que estão armazenados no array
    - Ex.: int32, int16, float64

Biblioteca Numpy

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
```

```
>>> type(a)
<type 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<type 'numpy.ndarray'>
```

Biblioteca Numpy

- A ideia então é criar estruturas **ndarray** e manipulá-las;
- A biblioteca Numpy fornece funções para manipulação desses dados;
- Vamos ver funções para:
  - Criação de array;
  - Operações básicas;
  - Manipulação da estrutura;
  - AlgLin - Multiplicação de arrays.

Biblioteca Numpy

- Criação de arrays:
  - numpy.arange
  - numpy.array
  - numpy.identity
  - numpy.ones
  - numpy.zeros

Biblioteca Numpy

- numpy.arange([start,]stop, [step,]dtype=None)
  - Cria um array começando 0 (ou em start) até o valor **stop** incrementando em 1 (ou em **step**)

```
>>> import numpy as np
>>> A = np.arange(10)
>>> A
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>>
```

Biblioteca Numpy

- numpy.array(object)
  - Cria um array a partir de um outro objeto (listas, tuplas, etc)

```
>>> import numpy as np
>>> A = np.array([0,1,2,3])
>>> A
array([0, 1, 2, 3])
>>> B = np.array((5,6,7))
>>> B
array([5, 6, 7])
```

Biblioteca Numpy

- numpy.identity(n, dtype=None)
  - Cria um array de dimensão n com a matriz identidade.

```
>>> import numpy as np
>>> I = np.identity(2)
>>> I
array([[ 1.,  0.],
       [ 0.,  1.]])
>>>
```

Biblioteca Numpy

- numpy.ones(shape, dtype=None, order='C')
  - Cria um array de dimensão m x n (definido no shape) com todos os valores iguais a 1

```
>>> import numpy as np
>>> O = np.ones(10)
>>> O
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
>>> O_2 = np.ones((2,2))
>>> O_2
array([[ 1.,  1.],
       [ 1.,  1.]])
>>>
```

Biblioteca Numpy

- numpy.zeros(shape, dtype=None, order='C')
  - Cria um array de dimensão m x n (definido no shape) com todos os valores iguais a 0

```
>>> import numpy as np
>>> O = np.zeros(10)
>>> O
array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
>>> O_2 = np.zeros((2,2))
>>> O_2
array([[ 0.,  0.],
       [ 0.,  0.]])
>>>
```

Biblioteca Numpy

- Operações básicas:
  - numpy.sum
  - numpy.prod
  - numpy.cumsum
  - numpy.cumprod

Biblioteca Numpy

- numpy.sum(a, axis=None, dtype=None, out=None)
  - Soma os valores de um array para uma determinada dimensão.

```
>>> import numpy as np
>>> A = np.arange(4)
>>> A
array([0, 1, 2, 3])
>>> A.sum()
6
>>>
```

Biblioteca Numpy

- numpy.prod(a, axis=None, dtype=None, out=None)
  - Multiplica os valores de um array para uma determinada dimensão.

```
>>> import numpy as np
>>> A = np.array([536870910, 536870910, 536870910, 536870910])
>>> np.prod(A)
69175290104612126 88
>>>
```

Biblioteca Numpy

- numpy.cumsum(a, axis=None, dtype=None, out=None)
  - Soma cumulativa dos valores de um array para uma determinada dimensão;
- numpy.cumprod(a, axis=None, dtype=None, out=None)
  - Multiplicação cumulativa dos valores de um array para uma determinada dimensão;

```
>>> import numpy as np
>>> A = np.arange(1,4)
>>> A
array([1, 2, 3])
```

```
>>> np.cumsum(A)
array([1, 3, 6])
>>> np.cumprod(A)
array([1, 2, 6])
>>>
```

Biblioteca Numpy

- Manipulação da estrutura:
  - numpy.mat
  - numpy.reshape
  - numpy.transpose

Biblioteca Numpy

- numpy.mat(data, dtype=None)
  - Transforma o objeto data em uma matriz

```
>>> import numpy as np
>>> A = np.array([[1, 2], [3, 4]])
>>> A
array([[1, 2],
       [3, 4]])
>>> M = np.mat(A)
>>> M
matrix([[1, 2],
        [3, 4]])
>>> M[0,1]
2
```

Biblioteca Numpy

- numpy.reshape(A, newshape, order='C')
  - Transforma as dimensões do objeto A para o shape = newshape

```
>>> import numpy as np
>>> A = np.arange(6)
>>> A
array([0, 1, 2, 3, 4, 5])
>>> A.reshape((3, 2))
array([[0, 1],
       [2, 3],
       [4, 5]])
>>>
```

Biblioteca Numpy

- numpy.transpose(data, dtype=None)
  - Faz a permutação das dimensões

```
>>> import numpy as np
>>> A = np.array([[1, 2], [3, 4]])
>>> A
array([[1, 2],
       [3, 4]])
>>> np.transpose(A)
array([[1, 3],
       [2, 4]])
>>>
```

Biblioteca Numpy

- Operações de algebra linear:
  - numpy.dot

Biblioteca Numpy

- numpy.dot(data, dtype=None)
  - Multiplicação de dois arrays.
  - Com arrays 2D é equivalente a multiplicação de matrizes.
  - Com arrays 1D é equivalente ao produto interno dos vetores.

Biblioteca Numpy

```
>>> import numpy as np
>>> np.dot(3, 4)
12
>>> A = np.array([[1, 0], [0, 1]])
>>> A
array([[1, 0],
       [0, 1]])
>>> B = np.array([[4, 1], [2, 2]])
>>> B
array([[4, 1],
       [2, 2]])
>>> np.dot(A,B)
array([[4, 1],
       [2, 2]])
```

Biblioteca Numpy

```
>>> import numpy as np
>>> B = np.array([[4, 1], [2, 2]])
>>> B
array([[4, 1],
       [2, 2]])
>>> C = np.array([[2, 2], [2, 2]])
>>> np.dot(B,C)
array([[10, 10],
       [ 8,  8]])
>>>
```

Exercícios

1. Crie um array com 10 elementos usando a função arange;

2. Transforme esse array de 1D (1x10) para 2D (2x5) usando a função reshape;

3. Crie duas matrizes a partir de dois arrays;

4. Crie duas matrizes e realize a multiplicação das mesmas.