

# Numpy Tutorial

Nicolas Pécheux

`nicolas.pecheux@limsi.fr`

LIMSI/CNRS — Université Paris Sud

January 29, 2014



# Why Numpy ?

- ▶ Numpy is a Python module for scientific computing
- ▶ Written in C
- ▶ Fast vector & matrix operations

Import module using an alias

```
1     import numpy as np
2     import numpy.random as npr
```

# Arrays

- ▶ In numpy we work with **arrays**

```
1     >>> lst = [3, 2, 4, 1]
2     >>> a = np.array(lst) # convert list to numpy array
3     >>> a
4     array([3, 2, 4, 1])
```

- ▶ All element have same type

```
1     >>> a.dtype.name
2     'int64'
```

# Vectors

- ▶ Vectors are just 1d arrays

```
1     >>> np.zeros(5)
2     array([ 0.,  0.,  0.,  0.,  0.])
3
4     >>> np.ones(4)
5     array([ 1.,  1.,  1.,  1.])
6
7     >>> npr.randn(2)  # normal distribution
8     array([ 0.26141913, -0.5324435 ])
9
10    >>> np.linspace(0, 1, 5)  # 5 uniform values in [0, 1]
11    array([ 0. ,  0.25,  0.5 ,  0.75,  1.  ])
```

# Matrices

- ▶ Matrices are just 2d arrays

```
1     >>> np.zeros((2,5))
2     array([[ 0.,  0.,  0.,  0.,  0.],
3            [ 0.,  0.,  0.,  0.,  0.]])
4
5     >>> npr.randn(3, 3)
6     array([[ -0.64043229,  0.68655366,  1.09269517],
7            [ 1.23037221, -0.79673819,  0.01486236],
8            [-1.15268157, -1.36926975, -0.64407652]])
```

## Array shape

- ▶ Array shape can be read and modified

```
1     >>> z = np.zeros(6)
```

```
2
```

```
3     >>> z.shape
```

```
4     (6,)
```

```
5
```

```
6     >>> z.reshape(3, 2)
```

```
7     array([[ 0.,  0.],  
8           [ 0.,  0.],  
9           [ 0.,  0.]])
```

```
10
```

```
11     >>> np.arange(9).reshape(3, 3)
```

```
12     array([[0, 1, 2],  
13           [3, 4, 5],  
14           [6, 7, 8]])
```

# Mathematical operations

- ▶ Arithmetic operators apply *elementwise*

```
1     >>> a = np.arange(6).reshape(2, 3)
2     >>> b = np.ones((2, 3))
3
4     >>> a + b
5     array([[ 1.,  2.,  3.],
6            [ 4.,  5.,  6.]])
7
8     >>> a * b
9     array([[ 0.,  1.,  2.],
10           [ 3.,  4.,  5.]])
11
12    >>> a < 3
13    array([[ True,  True,  True],
14           [False, False, False]], dtype=bool)
```

# Mathematical operations

- ▶ Very usefull !

```
1     >>> x = np.arange(4).reshape(2, 2)
2
3     >>> np.sin(x) * np.exp(-x ** 2 / 2)
4     array([[ 0.          ,  0.30955988],
5            [ 0.12306002,  0.00095086]])
```



## Methods

- ▶ By default apply to the array as though it were a list of number

```
1 >>> a = np.array([[0, 9, 2],
2                   [3, 4, 1]])
3
4 >>> a.min(), a.argmax(), a.sum()
5 (0, 1, 19)
```

- ▶ But one can apply along a specified axis only

```
1 >>> a.sum(axis=1)
2 array([11,  8])
3
4 >>> a.sort(axis=0)  # Modify a !
5 >>> a
6 array([[0, 4, 1],
7        [3, 9, 2]])
```

## Matrix operations

```
1     >>> A = np.arange(6).reshape(2, 3)
2     array([[0, 1, 2],
3            [3, 4, 5]])
4     >>> x = 2 * np.ones(3)
5
6     >>> A.dot(x)   # Ax
7     array([ 6., 24.])
8
9     >>> A.dot(A.T) # AA'
10    array([[ 5, 14],
11           [14, 50]])
12
13    >>> # Warning !
14    >>> A * x   # Broadcast + elementwise
15    array([[ 0.,  2.,  4.],
16           [ 6.,  8., 10.]])
```

## Indexing and slicing

```
1     >>> A = array([[0, 1, 2],
2                   [3, 4, 5]])
3
4     >>> A[1, 2]
5     5
6
7     >>> A[1, :]  # : means all elements of that axis
8     array([3, 4, 5])
9
10    >>> A[:, [0, 2]]
11    array([[0, 2],
12           [3, 5]])
13
14    >>> mask = A.min(axis=0) > 1  # Very useful !
15    >>> A[:, mask]
16    array([[2],
17           [5]])
```

## Statistics

```
1     >>> A = array([[0, 1, 2],
2                   [3, 4, 5]])
3
4     >>> A.mean()
5     2.5
6
7     >>> A.std(axis=0)
8     array([ 1.5,  1.5,  1.5])
9
10    >>> np.median(A, axis=1)
11    array([ 1.,  4.])
```

## Random

```
1     >>> import numpy.random as npr
2
3     >>> npr.seed(42)  # Seed the random generators
4
5     >>> npr.rand(3, 2)  # random in [0, 1)
6         array([[ 0.37454012,  0.95071431],
7                [ 0.73199394,  0.59865848],
8                [ 0.15601864,  0.15599452]])
9
10    >>> letters = np.array(['a', 'b', 'c', 'd'])
11    >>> npr.shuffle(letters)
12    >>> letters
13    array(['a', 'b', 'd', 'c'], dtype='|S1')
14
15    >>> npr.choice(letters, 2, replace=True) # numpy 1.7
16    array(['a', 'c'], dtype='|S1')
```