

# Numpy

Luís Pedro Coelho

Programming for Scientists

February 19, 2009



University of Pittsburgh

**Carnegie Mellon**

# Installation

- **Linux** Install packages `python-numpy` or similar.
- **Windows** Use Python(x,y)
- **Mac OS** Use Enthought Python Distribution

# Historical

- Numeric (1995)
- Numarray (for large arrays)
- [scipy.core](http://scipy.org) (briefly, around 2005)
- numpy (2005)

# Currently

- numpy 1.2
- **de facto** standard
- very stable

# Basic Type

`numpy.array` or `numpy.ndarray`.

Multi-dimensional array of numbers.

# numpy example

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

# Some Array Properties

```
import numpy as np
A = np.array([
    [0, 1, 2],
    [2, 3, 4],
    [4, 5, 6],
    [6, 7, 8]])
print A.shape
print A.size
```

# Some Array Functions

```
...  
print A.max()  
print A.min()
```

- `max()`: maximum
- `min()`: minimum
- `ptp()`: spread (max - min)
- `sum()`: sum
- `std()`: standard deviation
- ...

# Other Functions

- `np.exp`
- `np.sin`
- ...

All of these work **element-wise!**

# Arithmetic Operations

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

print A + B
print A * B
print A / B
```

# Broadcasting

## Mixing arrays of different dimensions

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

print A + B
print A * B
```

Special case: scalar.

```
import numpy as np
A = np.arange(100)
print A + 2
A += 2
```

`numpy.ndarray` is a homogeneous array of numbers.

## Types

- Boolean
- `int8`, `int16`, ...
- `uint8`, `uint16`, ...
- `float32`, `float64`, ...
- ...

Some types are only available in some platforms (e.g., `float96`).

# Object Construction

```
import numpy as np
A = np.array([0, 1, 1], np.float32)
A = np.array([0, 1, 1], float)
A = np.array([0, 1, 1], bool)
```

# Reduction

```
A = np.array([
    [0, 0, 1],
    [1, 2, 3],
    [2, 4, 2],
    [1, 0, 1]])
print A.max(0)
print A.max(1)
print A.max()
```

prints

```
[2, 4, 3]
```

```
[1, 3, 4, 1]
```

```
4
```

The same is true for many other functions.

# Slicing

```
import numpy as np
A = np.array([
    [0, 1, 2],
    [2, 3, 4],
    [4, 5, 6],
    [6, 7, 8]])
print A[0]
print A[0].shape
print A[1]
print A[:, 2]
```

# Slices Share Memory!

```
import numpy as np
A = np.array([
    [0, 1, 2],
    [2, 3, 4],
    [4, 5, 6],
    [6, 7, 8]])
B = A[0]
B[0] = -1
print A[0, 0]
```

# Pass is By Reference

```
def double(A):  
    A *= 2
```

```
A = np.arange(20)  
double(A)
```

# Pass is By Reference

```
def double(A):  
    A *= 2
```

```
A = np.arange(20)  
double(A)
```

```
A = np.arange(20)  
B = A.copy()
```

# Logical Arrays

```
A = np.array([-1, 0, 1, 2, -2, 3, 4, -2])  
print (A > 0)
```

# Logical Arrays II

```
A = np.array([-1, 0, 1, 2, -2, 3, 4, -2])  
print ( (A > 0) & (A < 3) ).mean()
```

What does this do?

# Logical Indexing

```
A[A < 0] = 0
```

or

```
A *= (A > 0)
```

# Logical Indexing

```
print 'Mean of positives', A[A > 0].mean()
```

# Some Helper Functions

## Constructing Arrays

```
A = np.zeros((10,10), dtype=np.int8)
B = np.ones(10)
C = np.arange(100).reshape((10,10))
...
```

## Multiple Dimensions

```
img = np.zeros((1024,1024,3), dtype=np.uint8)
```

<http://docs.scipy.org/doc/>