

# Session 04: Introduction to Numpy

October 9th, 2017 | Wouter Klijn

# Overview

- Introduction
- ‘Hello world’
- Arrays
  - Creating
  - Interacting
  - Copying
  - Differences with Matlab
- Matrixes vs Array
  - Why
  - Why not
- Matlib module
- Linear algebra

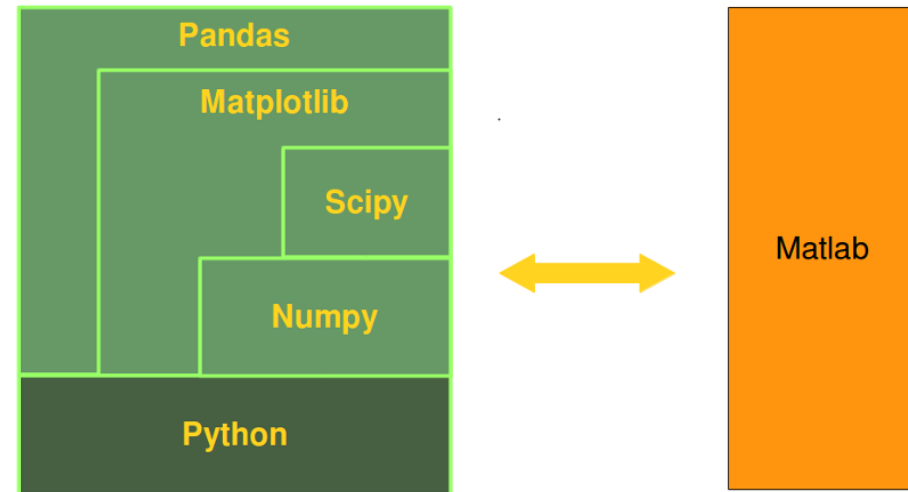
# Introduction

Python is free (free beer and freedom)!

Python is a complete programming language,

Numpy adds array oriented computing

Scipy mathematical algorithms and convenience functions on top of numpy



<https://www.python-course.eu/numpy.php>

# Hello world

```
#!/usr/bin/python3
import numpy as np

values = [23.1, 45.8, 94.4, 44.2]

np_array = np.array(values)

print np_array

print (np_array * 9 / 5 + 32)
```

From now on assume `python3` and “`import numpy as np`”

## Jupyter notebook s04

# Creating arrays

Scalar:

```
np.array(13)
```

A vector with two int:

```
np.array([43, 23])
```

A vector with float(64):

```
np.array([43.0, 92])
```

Manually select the type:

```
np.array([54, 39],  
dtype=np.float32)
```

Create nested arrays:

```
np.array([[24.0, 12],  
          [99.0, 33]])
```

## Jupyter notebook s04

# Interacting with arrays

The type of the array: `A.dtype`

Dimensionality: `.ndim` -or-  
`np.ndim(A)`

Shape: `A.shape` -or-  
`np.shape(A)`

Assignment and access of data via Indexing and Slicing

## Jupyter notebook s04

# Differences: indexing and slices

- **One** indexed vs **zero** indexing for Numpy
- Syntax: **round** vs. **square** brackets
- **Closed** vs. **open** ranges (  $[k,j]$  vs  $[k,j)$  )

<code>a(end)</code>	<code>a[-1]</code>	last element
<code>a(2,5)</code>	<code>a[1,4]</code>	second row, fifth column
<code>a(2,:)</code>	<code>a[1]</code> or <code>a[1,:]</code>	entire 2 <sup>nd</sup> row
<code>a(1:5,:)</code>	<code>a[0:5]</code> or <code>a[:5]</code> or <code>a[0:5,:]</code>	the first five rows of a
<code>a(end-4:end,:)</code>	<code>a[-5:]</code>	the last five rows of a

# Copying Numpy arrays

Create a reference:

```
B = A
```

Create a full copy:

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

Create a copy in existing array:

```
np.copyto(B, A)  
B[:] = a
```

## Jupyter notebook s04



# Differences: Pass by reference

- In Numpy values are passed by reference!
- Slices in an Array give views on the data
- MATLAB does copy on write
- Use `copy.deepcopy` for the same behavior in Numpy

# Matrix versus array

- Matlab: Default type is a multidimensional array. 2d operations default to linear matrix algebra.
- Numpy: Default type is a multidimensional array. Operations are element wise.
- Numpy has a matrix subclass that mirrors some Matlab functionality (More on this the next slides)
  - External packages might return Array with Matrix input
  - Not a complete reimplementatation

# Why use NUMPY.Matrix?

- When you know that all your operations will be linear algebra
- Construction can be more convenient: `matrix("[1 2 3; 4 5 6]")`
- Matrix forces 2d shape: `[0][x]`
- matrix also has `.H`, `.I`, and `.A` attributes

# Why not use NUMPY.Matrix?

- Matrix is always 2d
- External packages might return Array with Matrix input
- Inconsistent operator overloading
- Code might be harder to read for non MATLAB coders.  
Try to write code with least amount of surprise.

# numpy.matlib

Versions of numpy functions that return Matrix

- empty
- zeros
- ones
- eye
- identity
- repmat
- rand
- randn

# Linear algebra

## scipy.linalg module

Matlab	Python <code>scipy.linalg</code>
<code>norm(x)</code>	<code>sqrt(dot(v,v))</code> or <code>linalg.norm(v)</code>
<code>inv(v)</code>	<code>linalg.inv(a)</code>
<code>a\b</code>	<code>linalg.solve(a,b)</code> if a is square; <code>linalg.lstsq(a,b)</code> else
<code>[U,S,V]=svd(a)</code>	<code>U, S, Vh = linalg.svd(a)</code> <code>V = Vh.T</code>
<code>regress(y,X)</code>	<code>linalg.lstsq(X,y)</code>

<https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html>

# Linear regression

Ipython example session

<https://glowingpython.blogspot.de/2012/03/solving-overdetermined-systems-with-qr.html>  
<https://glowingpython.blogspot.de/2012/03/linear-regression-with-numpy.html>  
<https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.lstsq.html>

09/10/2017

# References and resources used

<https://glowingpython.blogspot.de/2012/03/solving-overdetermined-systems-with-qr.html>  
<https://glowingpython.blogspot.de/2012/03/linear-regression-with-numpy.html>  
<https://www.python-course.eu/numpy.php>

<https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html>  
<https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.lstsq.html>



## The people



# Thank you for your attention

References and further reading:

<https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html>

<https://github.com/jrjohansson/scientific-python-lectures/blob/master/Lecture-2-Numpy.ipynb>

<https://www.python-course.eu/numpy.php>