



Technical University of Lodz  
Institute of Electronics

# Algorithms and Data Structures

Python scripts

Łódź 2012





# General information

1. Script (<http://searchenterprise-linux.techtarget.com/definition/script>)

*In computer programming, a script is a program or sequence of instructions that is interpreted or carried out by another program rather than by the computer processor (as a compiled program is).*

2. Script is a **text file** and can be written within any Text Editor e.g.

- in Windows Notepad++ (<http://notepad-plus-plus.org/>)
- In MacOS TextWrangler (<http://www.barebones.com/products/TextWrangler/>)
- It is convenient to install e.g. Eclipse classic (with PyDev) for big projects



# Modules and Packages

**A module** is a file with extension `.py` that includes definitions of classes and functions related to its application.

**A package** is a set of modules or sub-packages. A package is a directory containing either `.py` files or sub-directories defining other packages.

The most useful packages in biomedical signal/image processing:

1. **numpy** – fundamental package for scientific computing with Python. It contains among other things:
  - i. A powerful N-dimensional array object
  - ii. Sophisticated (broadcasting) functions
  - iii. Tools for intergraing C/C++ and Fortran code
  - iv. Useful linear algebra, Fourier transform and random number capabilities
2. **scipy.signal** – signal processing toolbox
3. **scipy.ndimage** – n-d image processing and analysis toolbox
4. **scipy.linalg** – very fast linear algebra capabilities



# Standard Modules

Python has a large library of standard modules:  
This is the philosophy of "batteries included."

**re** - regular expressions

**datetime** - time and date objects

**math, cmath** - real and complex math

**decimal, fraction** - arbitrary precision  
decimal and rational number objects

**os, os.path, shutil** - filesystem operations

**sqlite3** - internal SQLite database

**gzip, bz2, zipfile, tarfile** – compression and  
archiving formats

**csv, netrc** – file format handling

**xml** – various modules for handling XML

**htmllib** – an HTML parser

**httplib, ftplib, poplib, socket, etc.** –  
modules for standard internet protocols

**cmd** – support for command interpreters

**pdb** – Python interactive debugger

**profile, cProfile, timeit** – Python profilers

**collections, heapq, bisect** – standard CS  
algorithms and data structures

**mmap** – memory-mapped files

**threading, Queue** – threading support

**subprocess** – executing external commands

**pickle, cPickle** – object serialization

**and many more... To see the content of one:**

```
>>> dir(module_name)
```



# Continuing Scripts

1. Inside the script you **have to import** all used modules!!!
2. All **Python scripts** should have extension “.py”, e.g: **script\_to\_run.py**
3. Run from a command line: **python script\_to\_run.py**
4. Run from IPython: **run script\_to\_run.py**
5. Edit (default text editor) and run (when closing file) from IPython: **edit script\_to\_run.py**



# Import Variations – Basic Import

#create script *sinus1.py* in **d://Biomed2012** folder

```
import pylab
```

```
import numpy
```

```
x = numpy.arange(100)
```

```
y = numpy.sin(x*numpy.pi/50.)
```

```
pylab.plot(x,y)
```

```
pylab.show()
```



# Import Variations – Aliasing Name

#create script *sinus2.py* in **d://Biomed2012** folder

```
import pylab as pl
```

```
import numpy as np
```

```
x = np.arange(100)
```

```
y = np.sin(x*np.pi/50.)
```

```
pl.plot(x,y)
```

```
pl.show()
```



# Import Variations – Specific Symbols

```
#create script sinus3.py in d://Biomed2012 folder
```

```
from pylab import plot, show, title  
from numpy import arange, sin, pi
```

```
x = arange(100)  
y = sin(x*pi/50)
```

```
plot(x,y)  
show()
```





# Import Variations – \*EVERYTHING\*

```
#create script sinus4.py in d://Biomed2012 folder
```

```
from pylab import *  
from numpy import *
```

```
x = arange(100)  
y = sin(x*pi/50)
```

```
plot(x,y)  
show()
```

***#NOT RECCOMENDED!!!***



# Imports in Scripts Console (I)Python

ENTHOUGHT

## Getting Started

### IMPORT NUMPY

```
In [1]: from numpy import *
```

```
In [2]: __version__
```

```
Out[2]: 1.6.0
```

or

```
In [1]: from numpy import \  
        array, ...
```

Often at the command line, it is handy to import everything from NumPy into the command shell.

However, if you are writing scripts, it is easier for others to read and debug in the future if you use explicit imports.

### USING IPYTHON -PYLAB

```
C:\> ipython -pylab
```

```
In [1]: array((1,2,3))
```

```
Out[1]: array([1, 2, 3])
```

IPython has a 'pylab' mode where it imports all of NumPy, Matplotlib, and SciPy into the namespace for you as a convenience. It also enables threading for showing plots.



While IPython is used for all the demos, '>>>' is used on future slides instead of 'In [1]:' to save space.

81