MONDRAGON UNIBERTSITATEA

GOI ESKOLA POLITEKNIKOA

ESCUELA POLITÉCNICA SUPERIOR

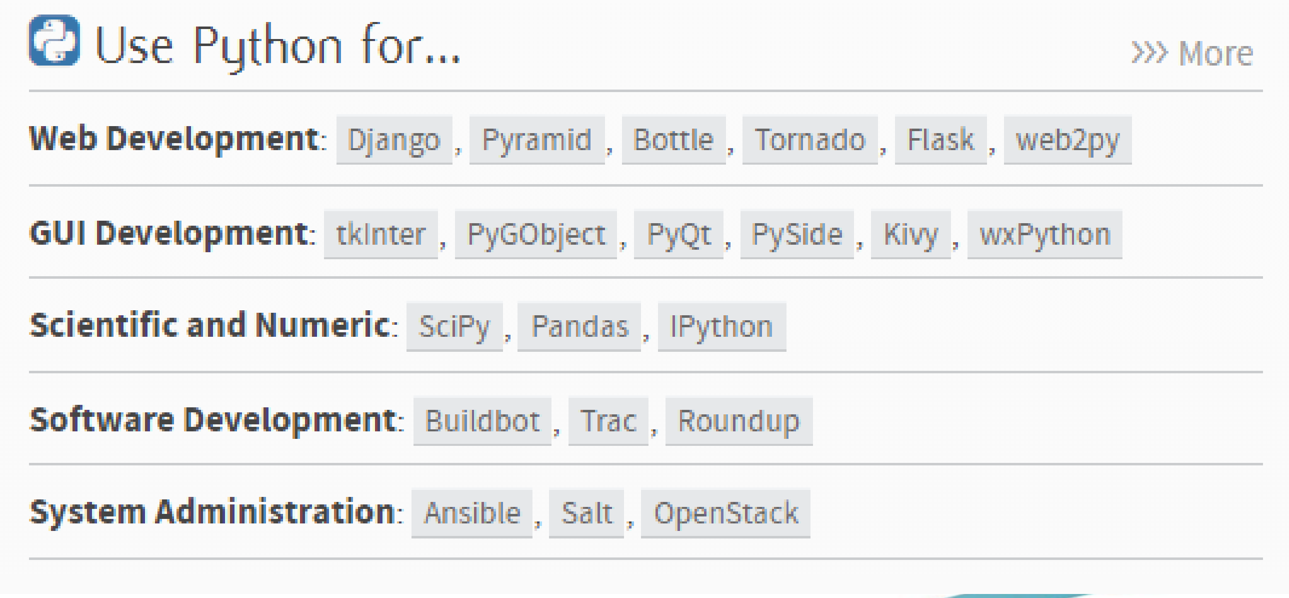
2) Minería de Datos con Python

- Python
  - Instalación (Anaconda python)
  - Terminal
  - Spyder, (eclipse)
  - Ipython, jupyter
- Programación Python & Librerías típicas
  - Python Básico
  - Matplotlib
  - Numpy
  - Scipy
  - Pandas
  - scikit-learn
  - Word2vec, gensim

MONDRAGON UNIBERTSITATEA
GOI ESKOLA POLITEKNIKOA
ESCUELA POLITÉCNICA SUPERIOR

Python

- Rápido & Fácil de Aprender
- Interpretación intuitiva
- Entorno más utilizado para Data Science (junto a R)
- Utilizado para multitud de aplicaciones



Use Python for...

Web Development: Django, Pyramid, Bottle, Tornado, Flask, web2py

GUI Development: tkInter, PyGObject, PyQt, PySide, Kivy, wxPython

Scientific and Numeric: SciPy, Pandas, IPython

Software Development: Buildbot, Trac, Roundup

System Administration: Ansible, Salt, OpenStack

Python

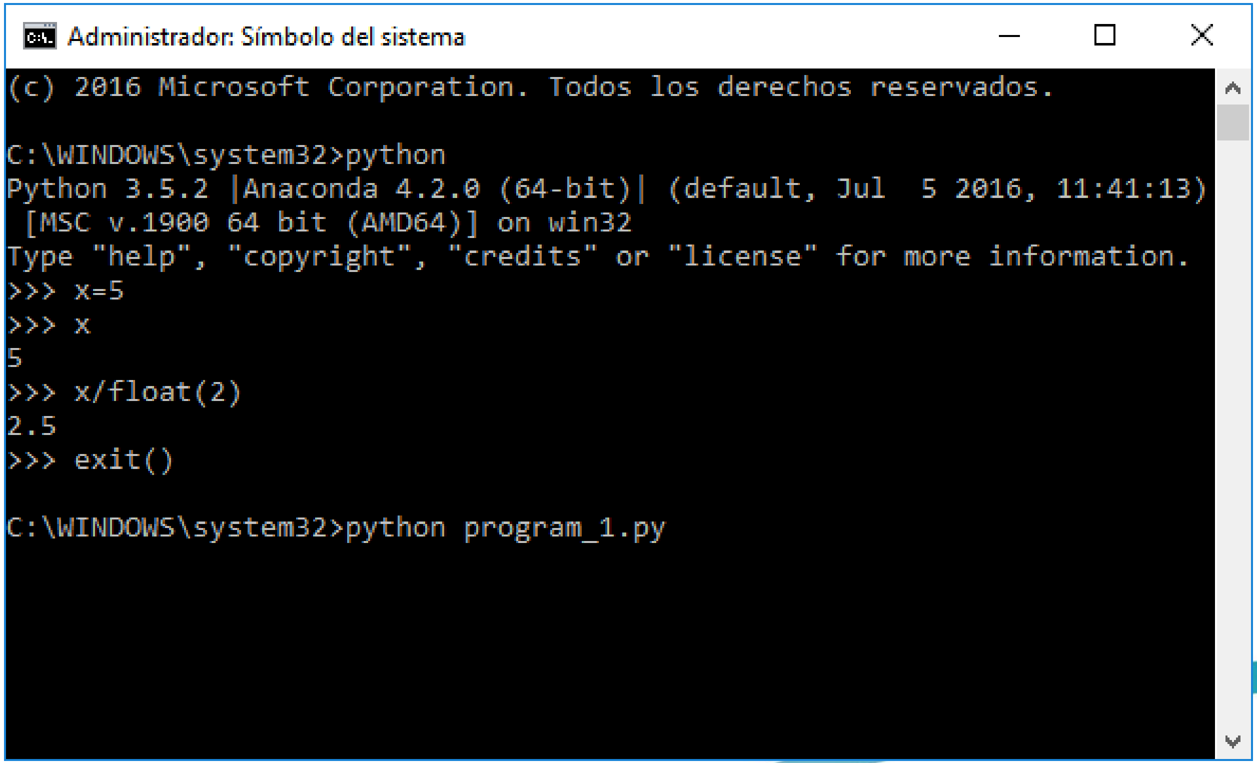
Instalación: Anaconda Python

- Plataforma Data Science basada en Python
- Paquete completo de instalación con multitud de paquetes/librerías utilizadas
- https://www.continuum.io/downloads

Python

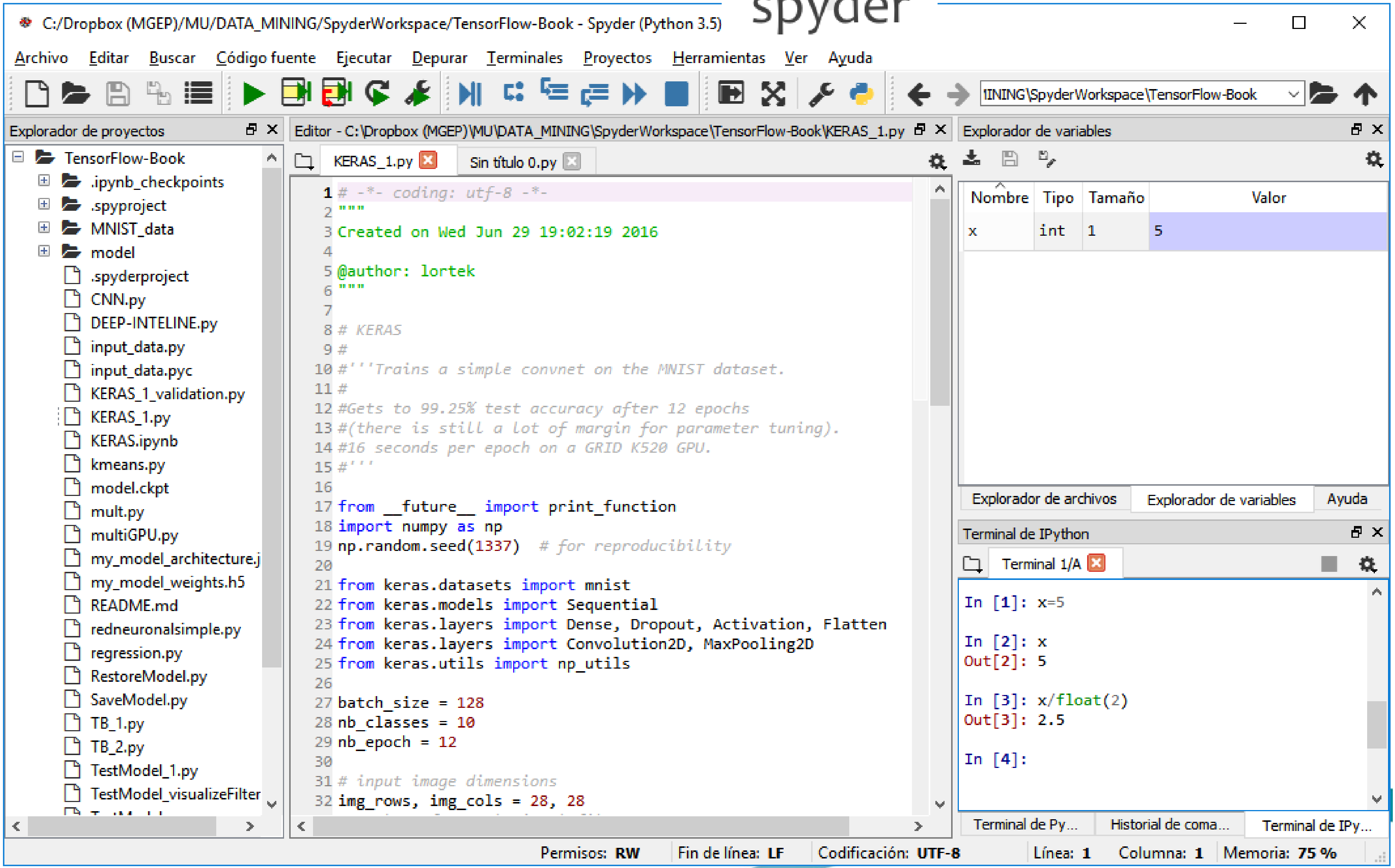
Instalación: Anaconda

Interacción: Terminal, Spyder, Jupyter



```
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>python
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul  5 2016, 11:41:13)
 [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> x=5
>>> x
5
>>> x/float(2)
2.5
>>> exit()

C:\WINDOWS\system32>python program_1.py
```

Spyder



```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 29 19:02:19 2016

@author: lortek
"""

# KERAS
#
#'''Trains a simple convnet on the MNIST dataset.
#
#Gets to 99.25% test accuracy after 12 epochs
#(there is still a lot of margin for parameter tuning).
#16 seconds per epoch on a GRID K520 GPU.
#'''

from __future__ import print_function
import numpy as np
np.random.seed(1337)  # for reproducibility

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.utils import np_utils

batch_size = 128
nb_classes = 10
nb_epoch = 12

# input image dimensions
img_rows, img_cols = 28, 28
```

Nombre	Tipo	Tamaño	Valor
x	int	1	5

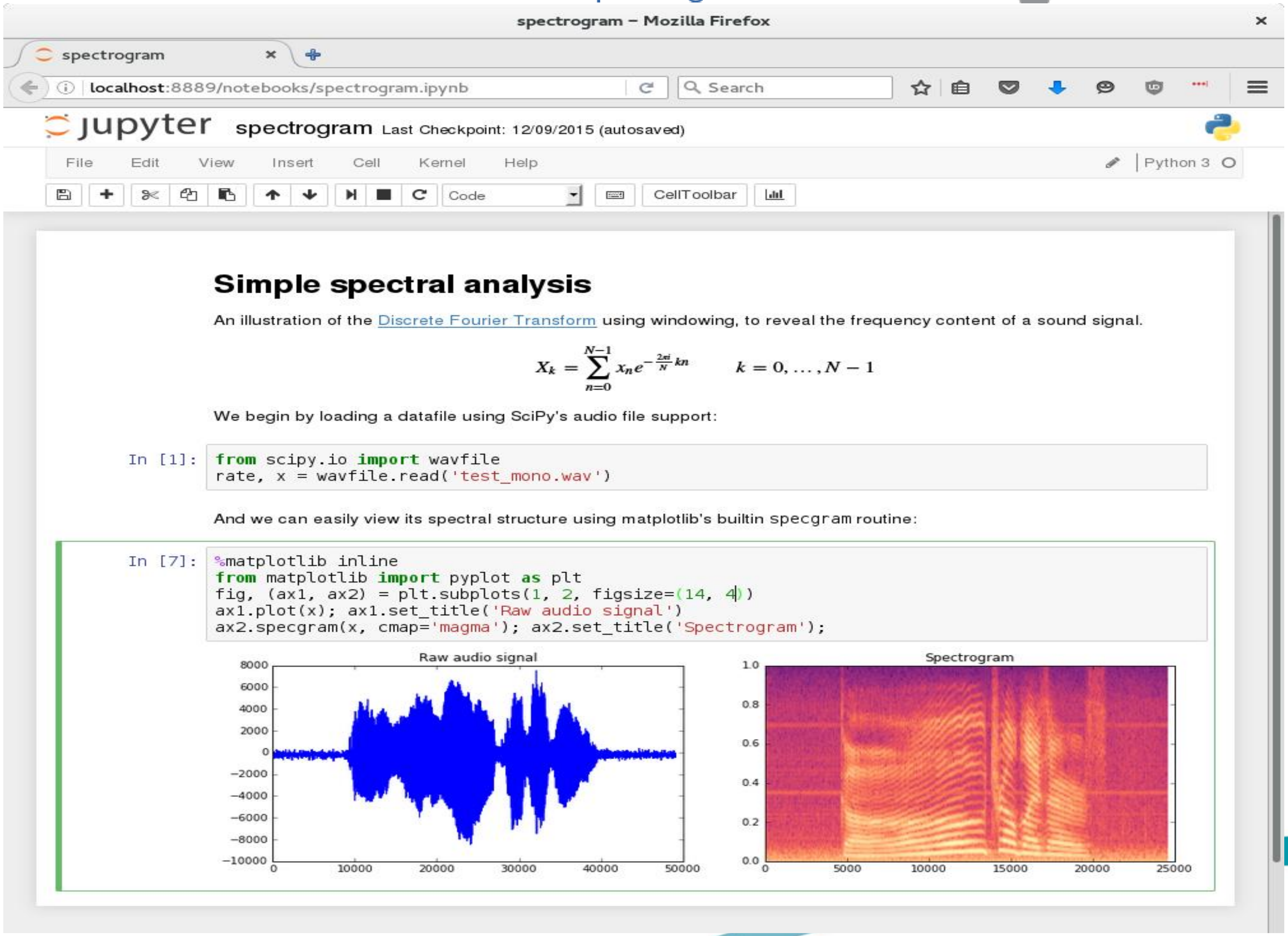
```
In [1]: x=5

In [2]: x
Out[2]: 5

In [3]: x/float(2)
Out[3]: 2.5

In [4]:
```

Permisos: RW   Fin de línea: LF   Codificación: UTF-8   Línea: 1   Columna: 1   Memoria: 75 %

Ipython: Jupyter

IP[y]: IPython
Interactive Computing

jupyter



spectrogram – Mozilla Firefox

localhost:8889/notebooks/spectrogram.ipynb

jupyter spectrogram Last Checkpoint: 12/09/2015 (autosaved)

Simple spectral analysis

An illustration of the Discrete Fourier Transform using windowing, to reveal the frequency content of a sound signal.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

We begin by loading a datafile using SciPy's audio file support:

In [1]:

```
from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin `specgram` routine:

In [7]:

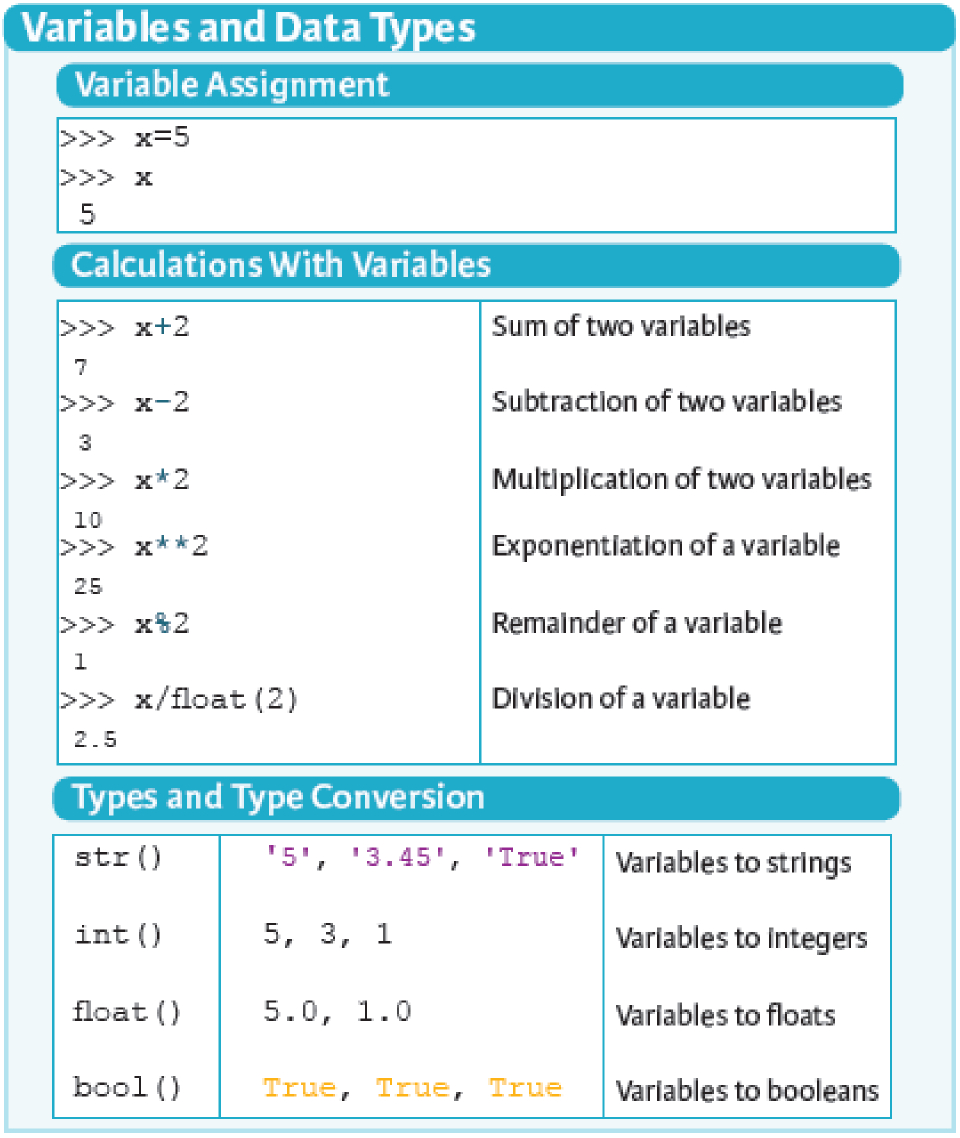
```
%matplotlib inline
from matplotlib import pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x, cmap='magma'); ax2.set_title('Spectrogram');
```

Probar Interacciones

- En terminal:
  - python
  - python program.py (editando en fichero .py)
  - ipython
- Spyder
  - Comandos en el ipython
  - Proyectos, getión de ficheros
- Notebooks
  - ipython notebook (desde terminal)
  - jupyter notebook (desde terminal)

Programación Python & Librerías típicas:

- Python Básico



Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
 5
```

Calculations With Variables

`>>> x+2`
`7`
	Sum of two variables

`>>> x-2`
`3`
	Subtraction of two variables

`>>> x*2`
`10`
	Multiplication of two variables

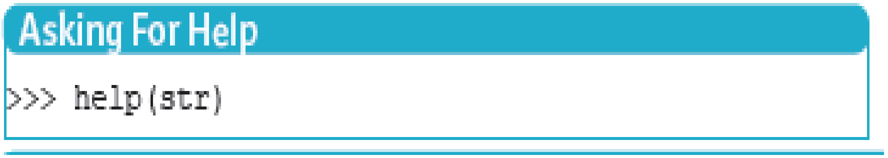
`>>> x**2`
`25`
	Exponentiation of a variable

`>>> x%2`
`1`
	Remainder of a variable

`>>> x/float(2)`
`2.5`
	Division of a variable

Types and Type Conversion

`str()`	`'5', '3.45', 'True'`	Variables to strings
`int()`	`5, 3, 1`	Variables to integers
`float()`	`5.0, 1.0`	Variables to floats
`bool()`	`True, True, True`	Variables to booleans

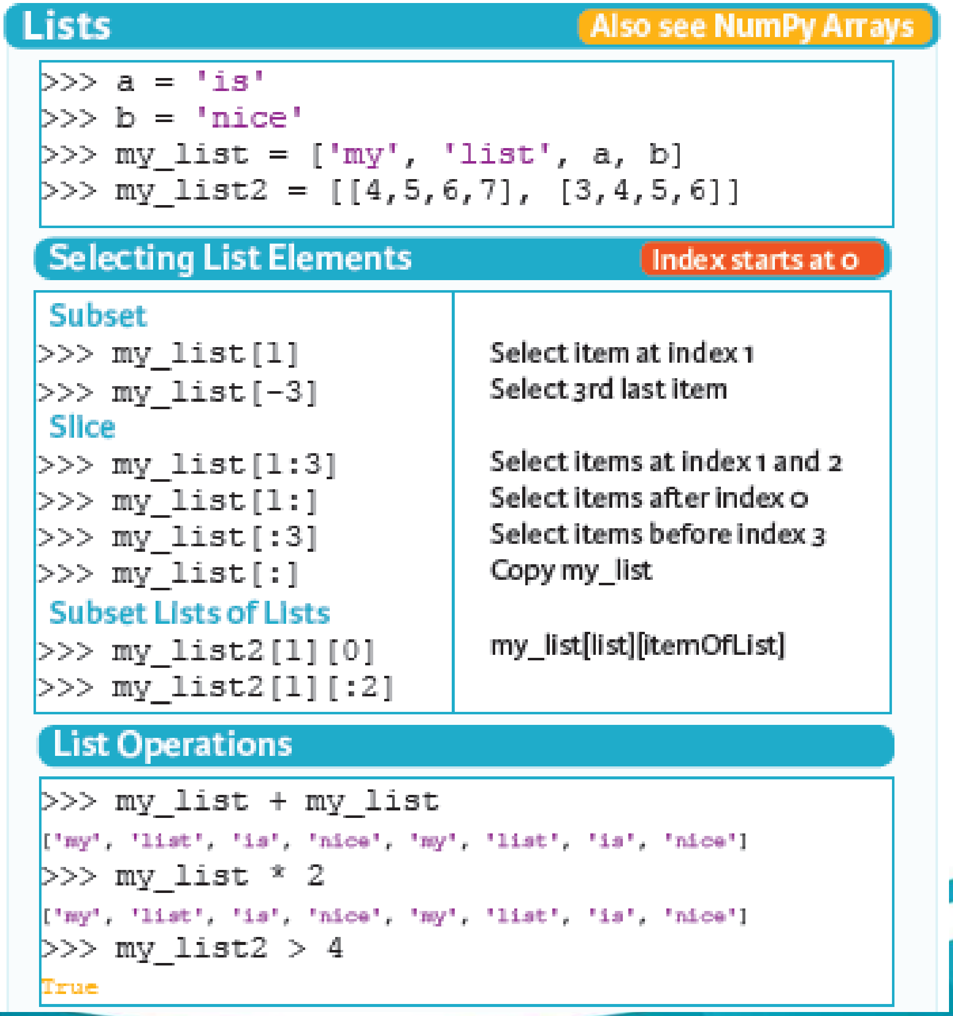


Asking For Help

```
>>> help(str)
```

Programación Python & Librerías típicas:

- Python Básico



Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

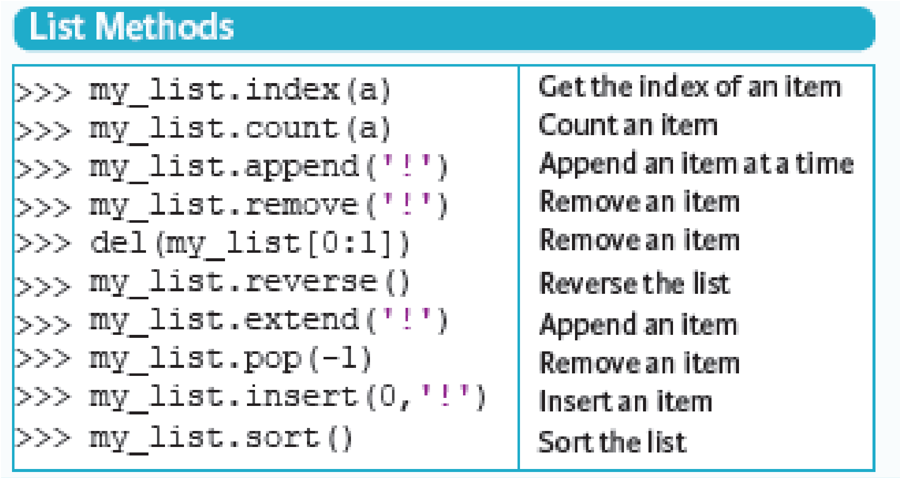
Index starts at 0

Subset	
`>>> my_list[1]`	Select item at index 1
`>>> my_list[-3]`	Select 3rd last item
Slice	
`>>> my_list[1:3]`	Select items at index 1 and 2
`>>> my_list[1:]`	Select items after index 0
`>>> my_list[:3]`	Select items before index 3
`>>> my_list[:]`	Copy my_list
Subset Lists of Lists	

`>>> my_list2[1][0]`
`>>> my_list2[1][:2]`
	my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

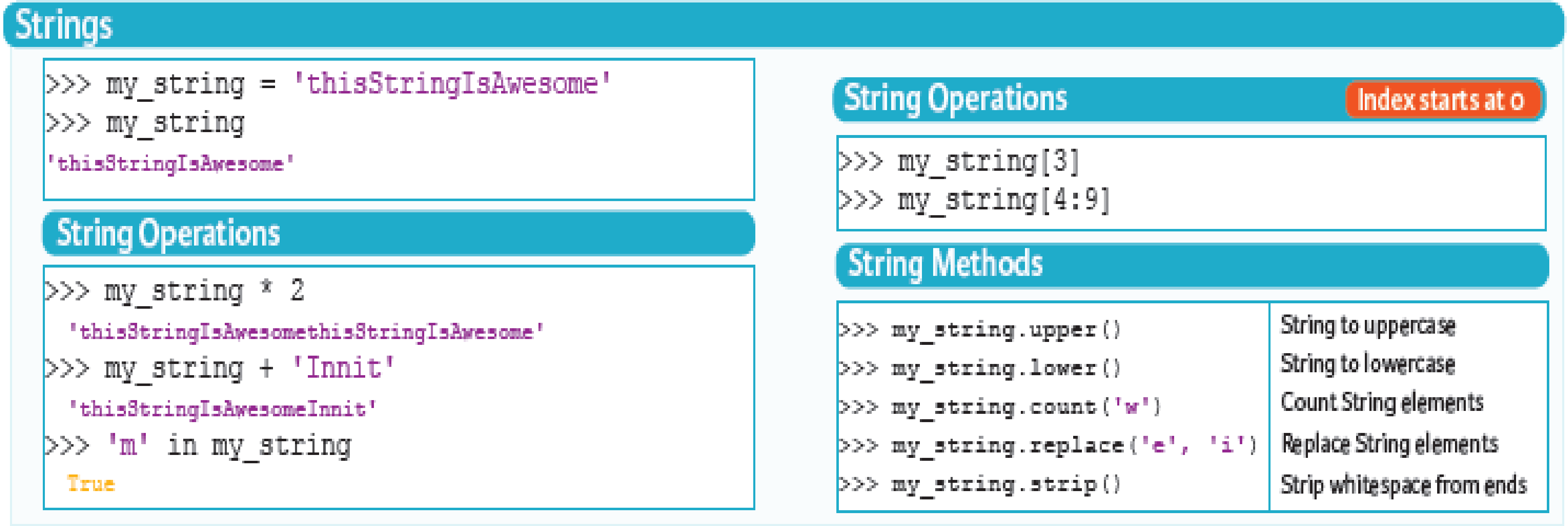


List Methods

`>>> my_list.index(a)`	Get the index of an item
`>>> my_list.count(a)`	Count an item
`>>> my_list.append('!')`	Append an item at a time
`>>> my_list.remove('!')`	Remove an item
`>>> del(my_list[0:1])`	Remove an item
`>>> my_list.reverse()`	Reverse the list
`>>> my_list.extend('!')`	Append an item
`>>> my_list.pop(-1)`	Remove an item
`>>> my_list.insert(0,'!')`	Insert an item
`>>> my_list.sort()`	Sort the list

Programación Python & Librerías típicas:

- Python Básico



Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
 'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
 'thisStringIsAwesomeInnit'
>>> 'm' in my_string
 True
```

String Operations

Index starts at 0

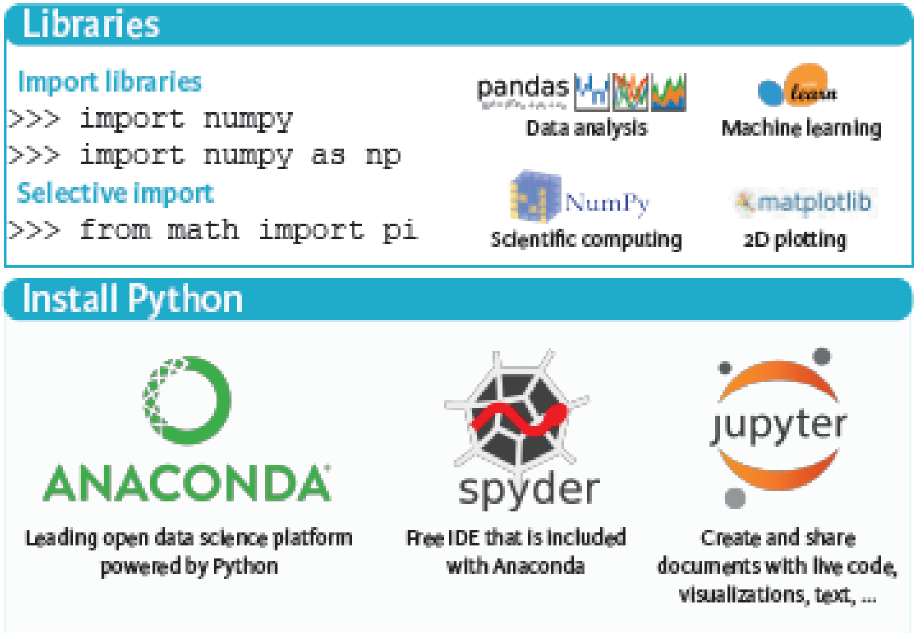
```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

`>>> my_string.upper()`	String to uppercase
`>>> my_string.lower()`	String to lowercase
`>>> my_string.count('w')`	Count String elements
`>>> my_string.replace('e', 'i')`	Replace String elements
`>>> my_string.strip()`	Strip whitespace from ends

Programación Python & Librerías típicas:

- Python Básico



Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
```

Selective import

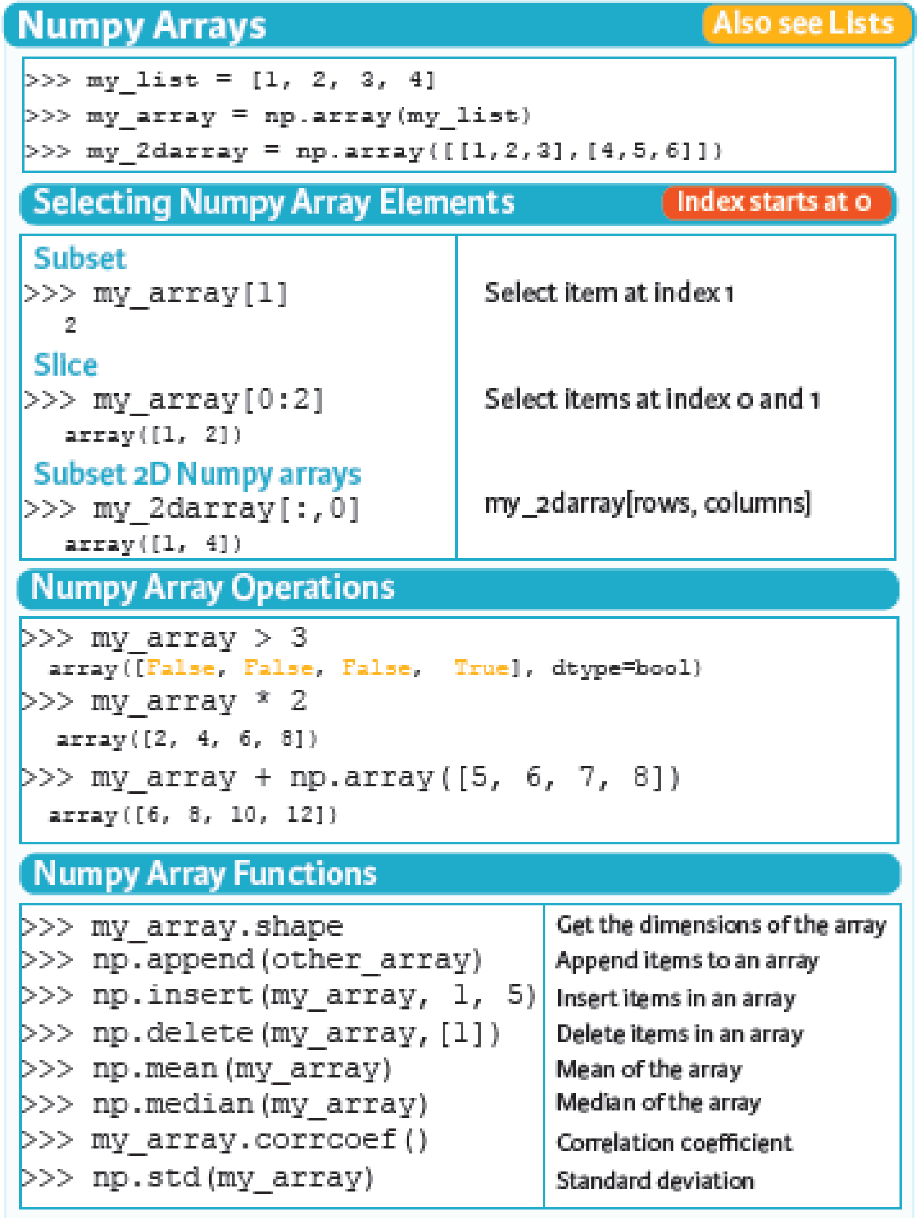
```
>>> from math import pi
```

Install Python

Leading open data science platform powered by Python

Free IDE that is included with Anaconda

Create and share documents with live code, visualizations, text, ...



Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset
`>>> my_array[1]`
`2`
	Select item at index 1

Slice
`>>> my_array[0:2]`
`array([1, 2])`
	Select items at index 0 and 1

Subset 2D Numpy arrays
`>>> my_2darray[:,0]`
`array([1, 4])`
	my_2darray[rows, columns]

Numpy Array Operations

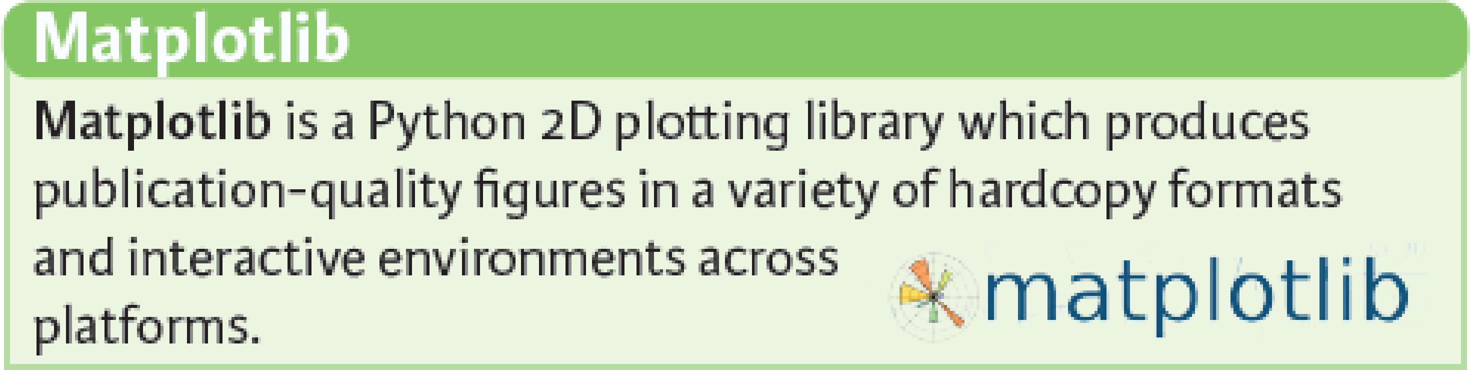
```
>>> my_array > 3
  array([False, False, False,  True], dtype=bool)
>>> my_array * 2
  array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
  array([6, 8, 10, 12])
```

Numpy Array Functions

`>>> my_array.shape`	Get the dimensions of the array
`>>> np.append(other_array)`	Append items to an array
`>>> np.insert(my_array, 1, 5)`	Insert items in an array
`>>> np.delete(my_array, [1])`	Delete items in an array
`>>> np.mean(my_array)`	Mean of the array
`>>> np.median(my_array)`	Median of the array
`>>> my_array.corrcoef()`	Correlation coefficient
`>>> np.std(my_array)`	Standard deviation

Programación Python & Librerías típicas:

- MATPLOTLIB



Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

matplotlib

Programación Python & Librerías típicas:

- MATPLOTLIB

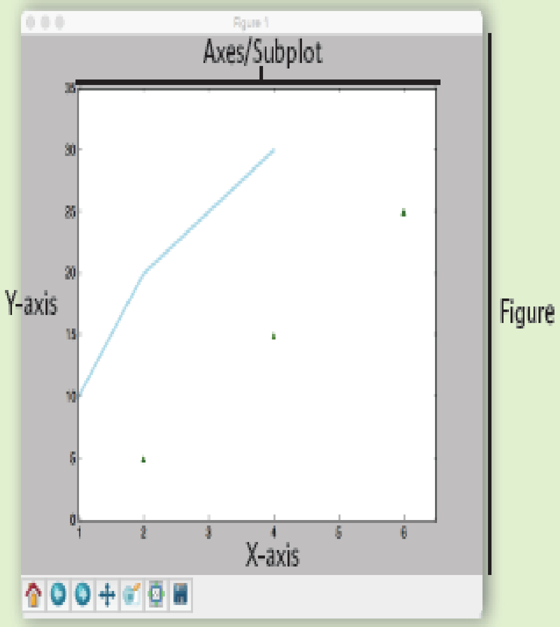
Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

matplotlib

Plot Anatomy & Workflow

Plot Anatomy



Workflow

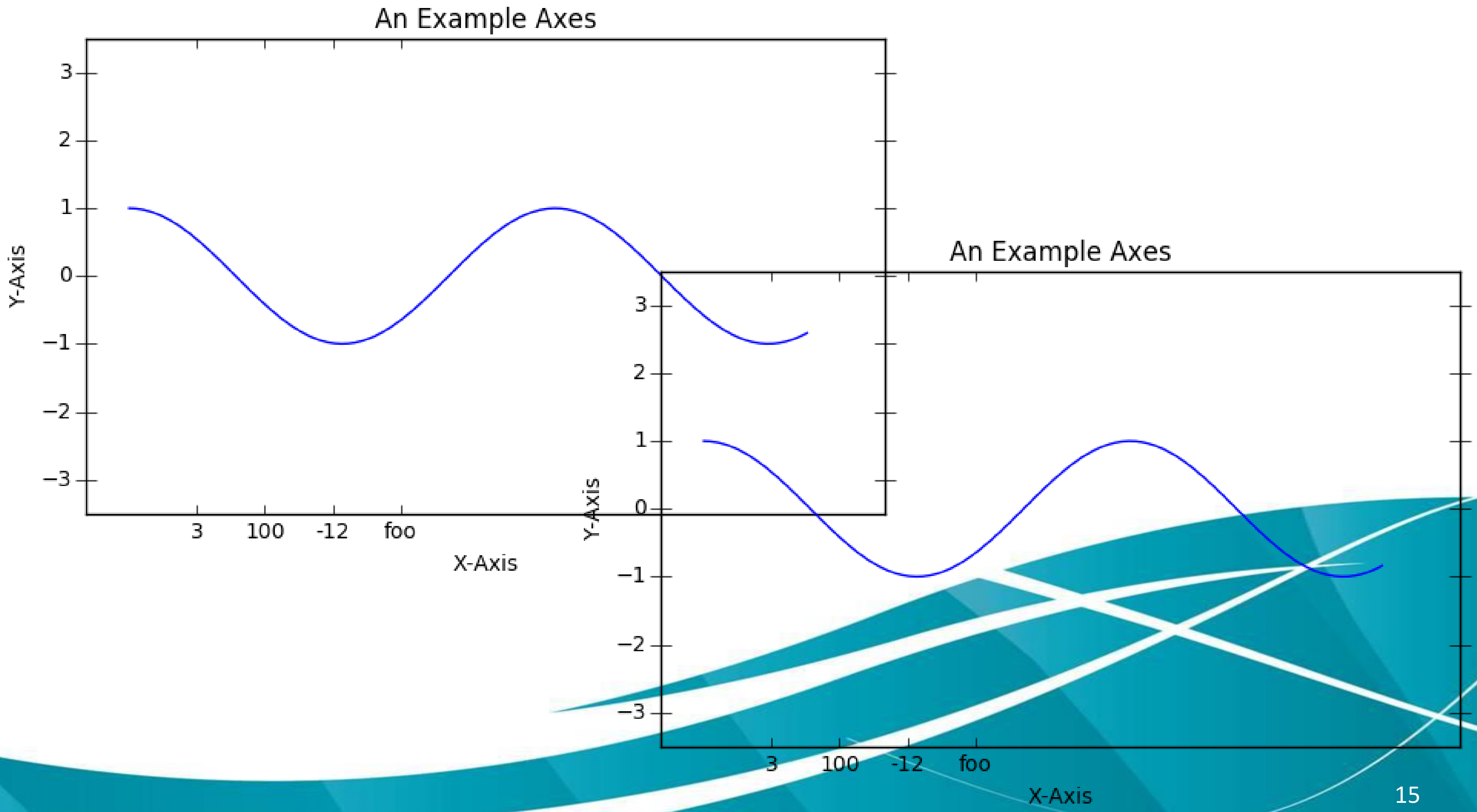
The basic steps to creating plots with matplotlib are:

1 Prepare data 2 Create plot 3 Plot 4 Customize plot 5 Save plot 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4]                    Step 1
>>> y = [10,20,25,30]
>>> fig = plt.figure()  Step 2
>>> ax = fig.add_subplot(111)  Step 3
>>> ax.plot(x, y, color='lightblue', linewidth=3)  Step 3, 4
>>> ax.scatter([2,4,6],
               [5,15,25],
               color='darkgreen',
               marker='^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png')
>>> plt.show()           Step 6
```

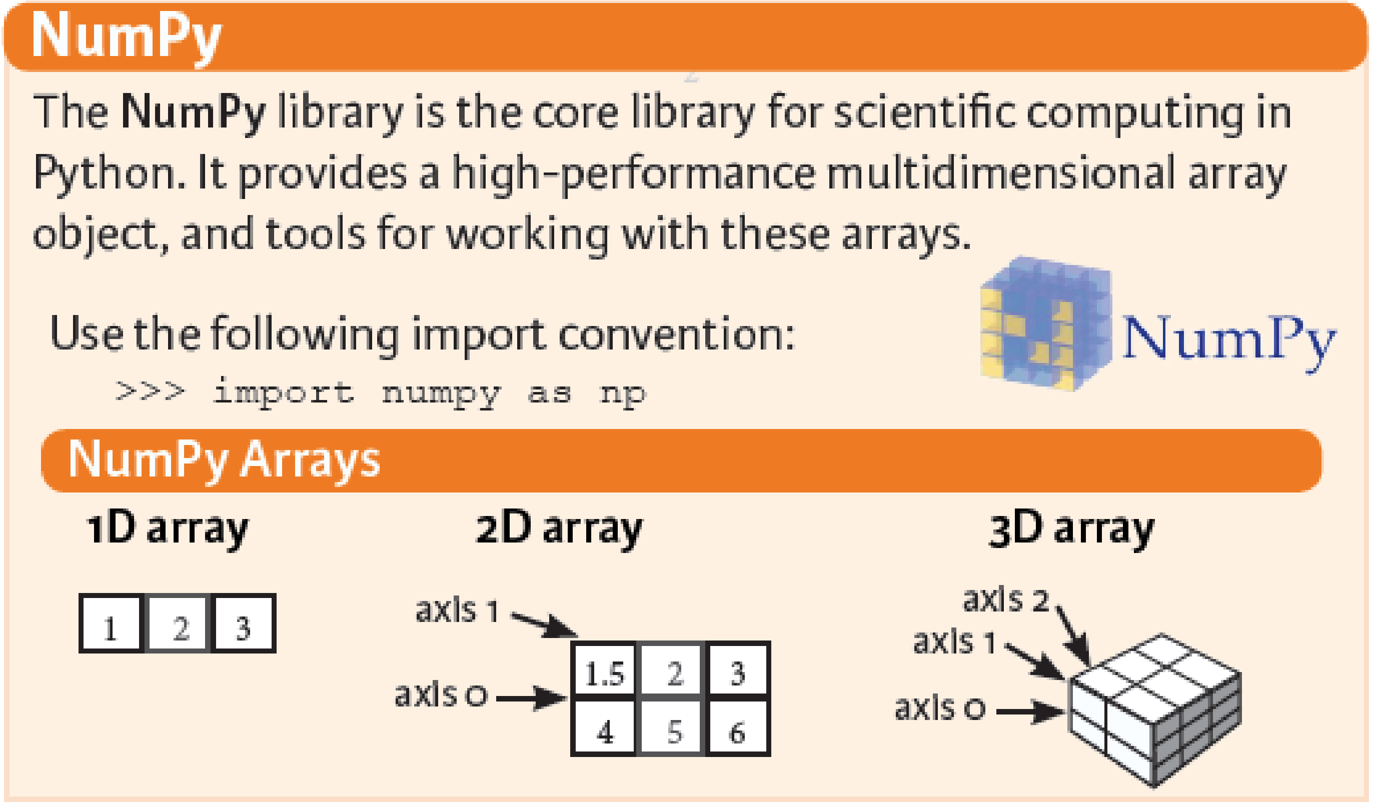
Programación Python & Librerías típicas:

- MATPLOTLIB



Programación Python & Librerías típicas:

- NUMPY



NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

1D array

1	2	3

2D array

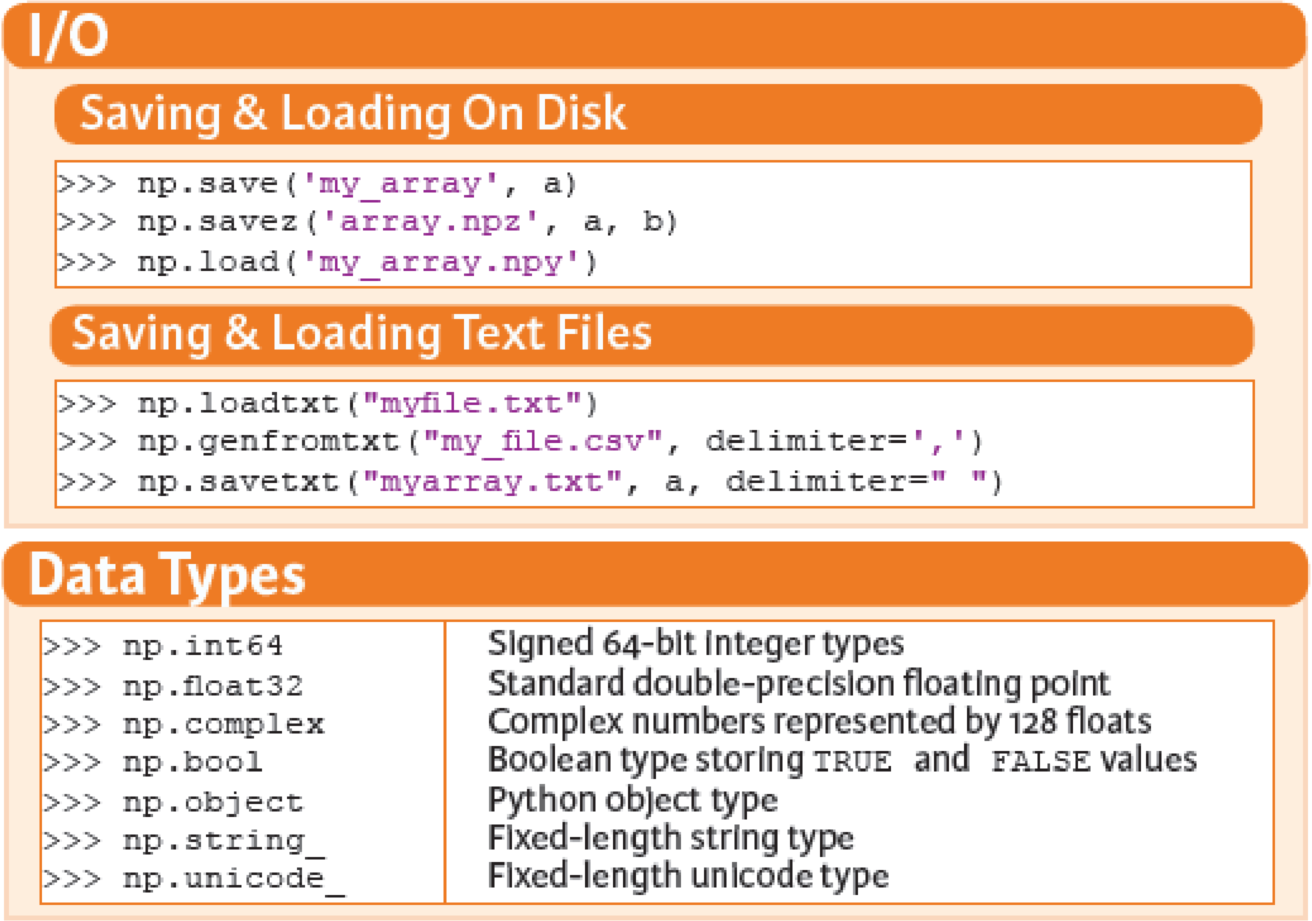
1.5	2	3
4	5	6

3D array

NumPy
axis 1
axis 0
axis 2
axis 1
axis 0

Programación Python & Librerías típicas:

- NUMPY



I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

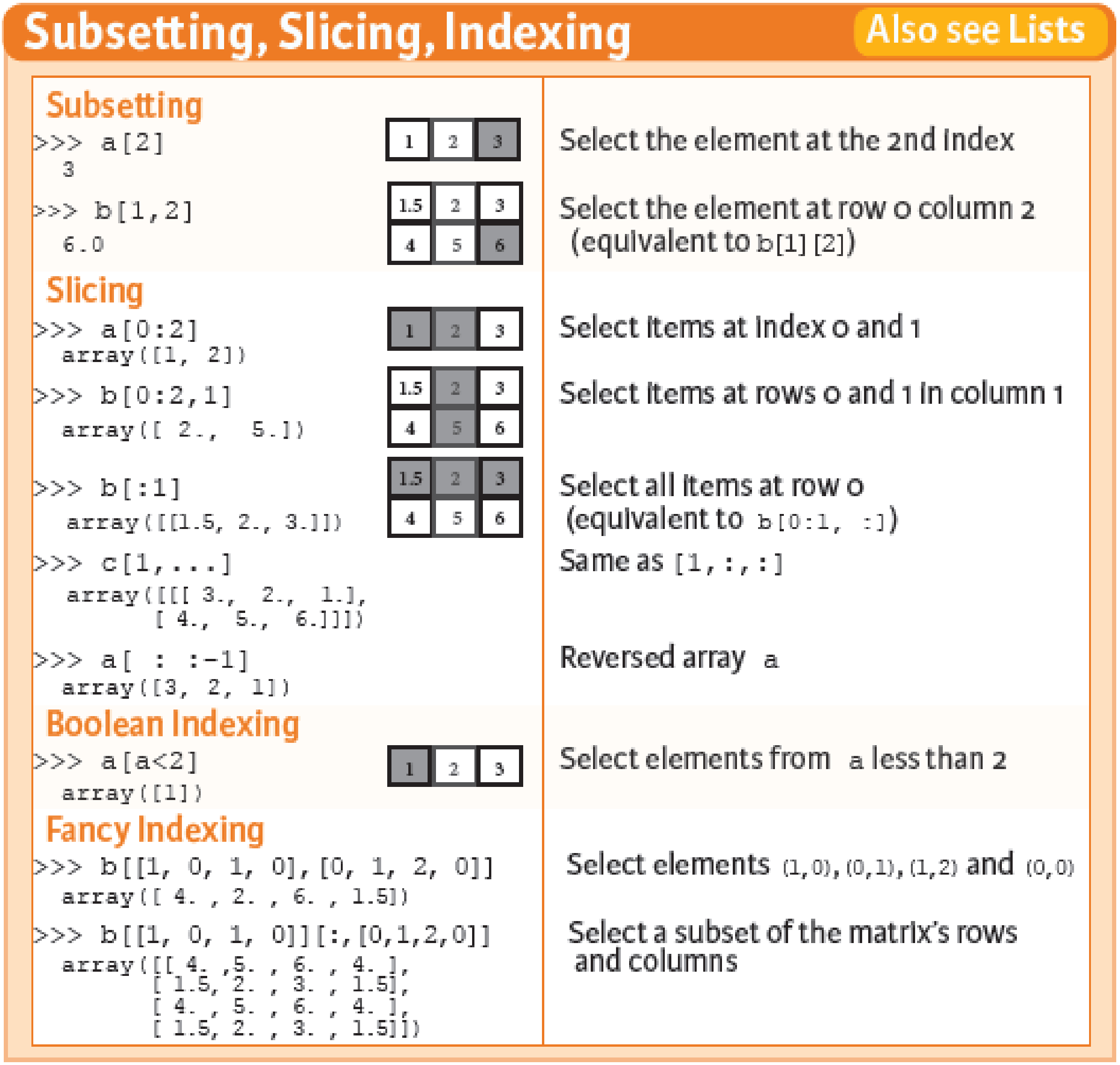
```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

`>>> np.int64`	Signed 64-bit integer types
`>>> np.float32`	Standard double-precision floating point
`>>> np.complex`	Complex numbers represented by 128 floats
`>>> np.bool`	Boolean type storing `TRUE` and `FALSE` values
`>>> np.object`	Python object type
`>>> np.string_`	Fixed-length string type
`>>> np.unicode_`	Fixed-length unicode type

Programación Python & Librerías típicas:

- NUMPY



Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]
 3
```

Select the element at the 2nd index

```
>>> b[1,2]
 6.0
```

Select the element at row 0 column 2 (equivalent to b[1][2])

Slicing

```
>>> a[0:2]
 array([1, 2])
```

Select items at index 0 and 1

```
>>> b[0:2,1]
 array([ 2.,  5.])
```

Select items at rows 0 and 1 in column 1

```
>>> b[:1]
 array([[1.5, 2., 3.]])
```

Select all items at row 0 (equivalent to b[0:1, :])

```
>>> c[1,...]
 array([[[ 3.,  2.,  1.],
         [ 4.,  5.,  6.]]])
```

Same as [1,:,:]

```
>>> a[ : :-1]
 array([3, 2, 1])
```

Reversed array a

Boolean Indexing

```
>>> a[a<2]
 array([1])
```

Select elements from a less than 2

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]
 array([ 4. , 2. , 6. , 1.5])
```

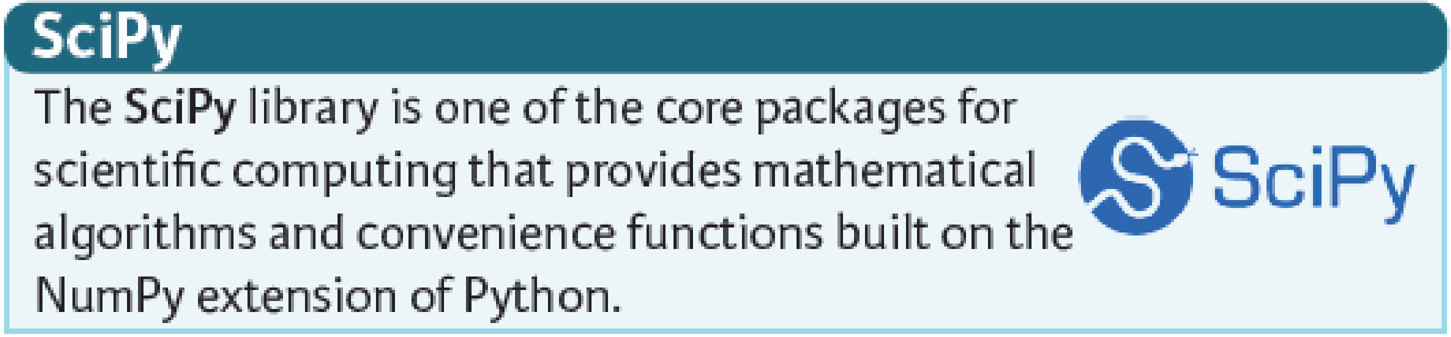
Select elements (1,0),(0,1),(1,2) and (0,0)

```
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]
 array([[ 4. ,5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5],
        [ 4. , 5. , 6. , 4. ],
        [ 1.5, 2. , 3. , 1.5]])
```

Select a subset of the matrix's rows and columns

Programación Python & Librerías típicas:

- Scipy

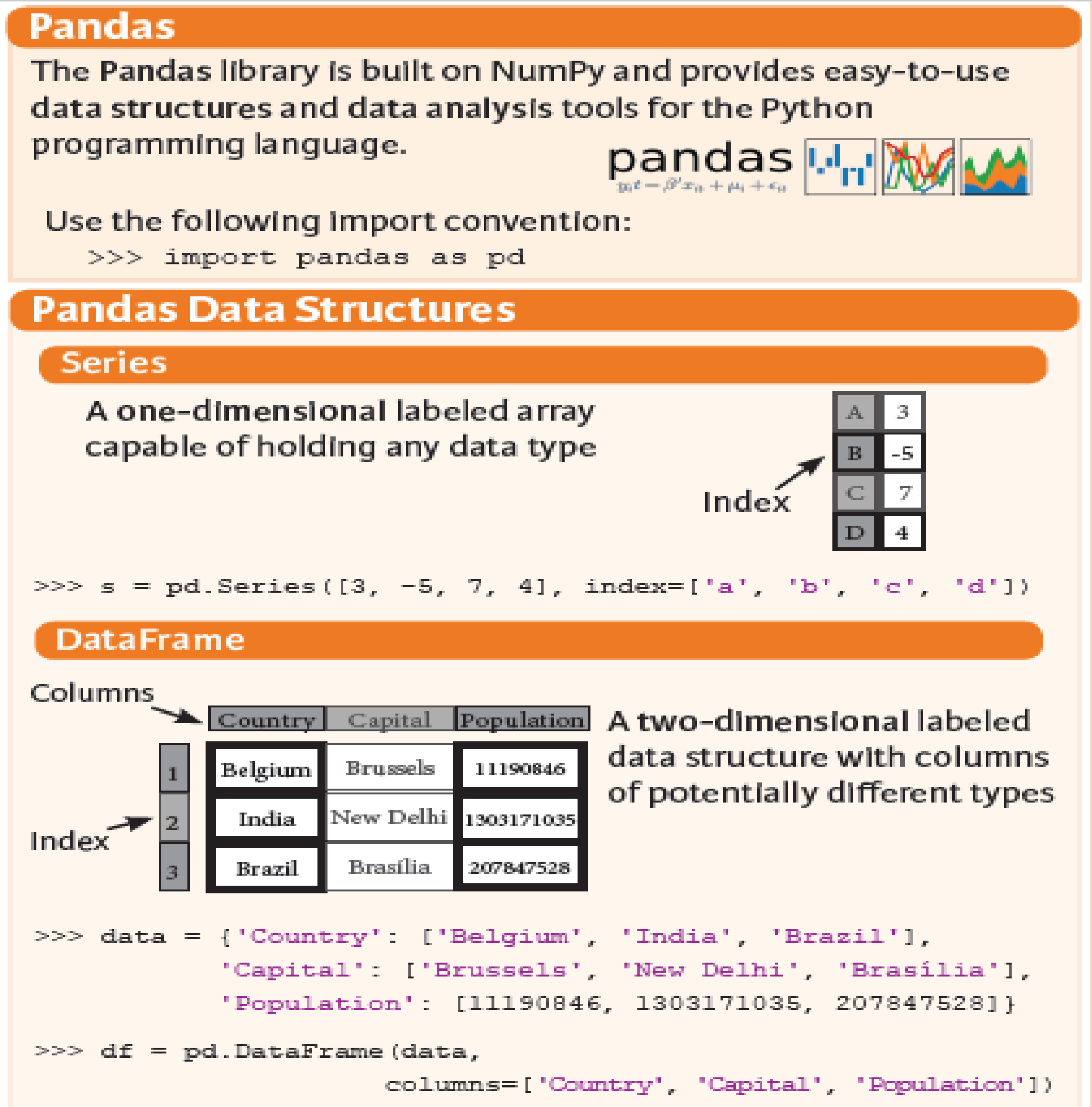


SciPy

The **SciPy** library is one of the core packages for scientific computing that provides mathematical algorithms and convenience functions built on the NumPy extension of Python.

Programación Python & Librerías típicas:

- Pandas



Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A one-dimensional labeled array capable of holding any data type

Index

A	3
B	-5
C	7
D	4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns

Index

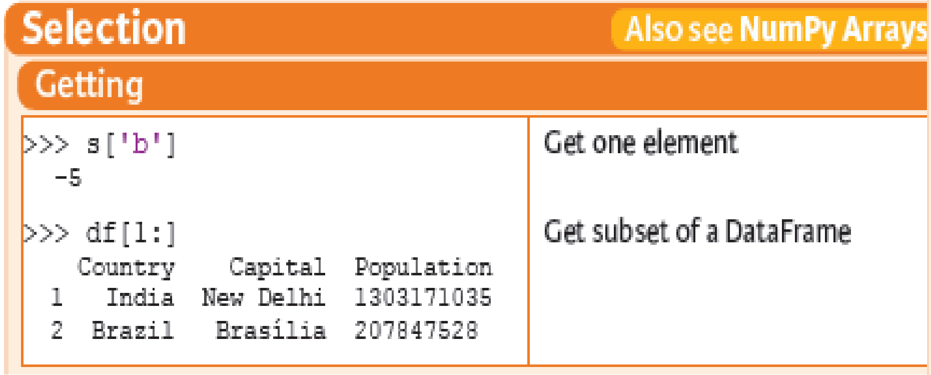
	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasília	207847528

A two-dimensional labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Programación Python & Librerías típicas:

- Pandas



Selection

Also see NumPy Arrays

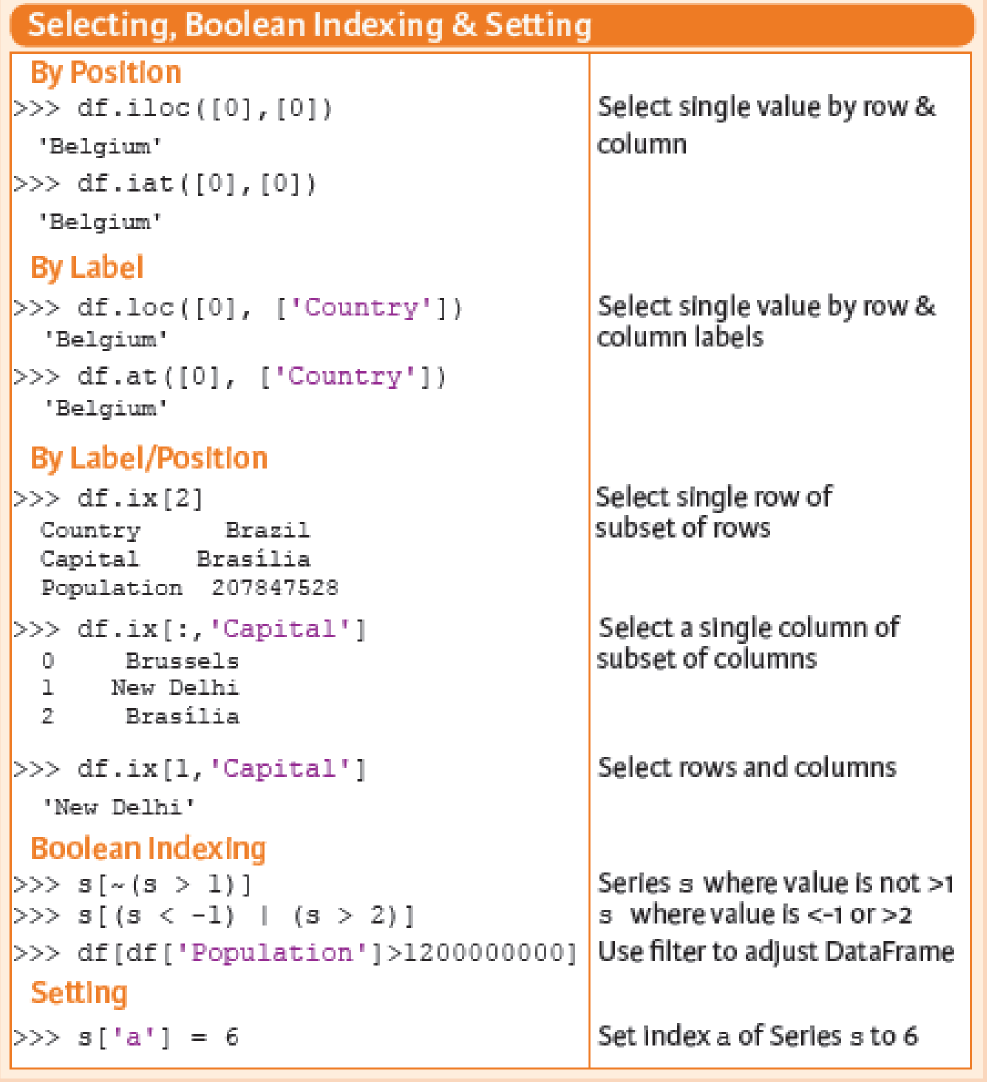
Getting

```
>>> s['b']
  -5
```

Get one element

```
>>> df[1:]
   Country     Capital   Population
 1   India   New Delhi   1303171035
 2  Brazil    Brasília    207847528
```

Get subset of a DataFrame



Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc([0],[0])
 'Belgium'
>>> df.iat([0],[0])
 'Belgium'
```

Select single value by row & column

By Label

```
>>> df.loc([0], ['Country'])
  'Belgium'
>>> df.at([0], ['Country'])
  'Belgium'
```

Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
 Country        Brazil
 Capital      Brasília
 Population  207847528
```

Select single row of subset of rows

```
>>> df.ix[:,'Capital']
 0      Brussels
 1     New Delhi
 2      Brasília
```

Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
 'New Delhi'
```

Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population']>1200000000]
```

Series s where value is not >1
s where value is <-1 or >2
Use filter to adjust DataFrame

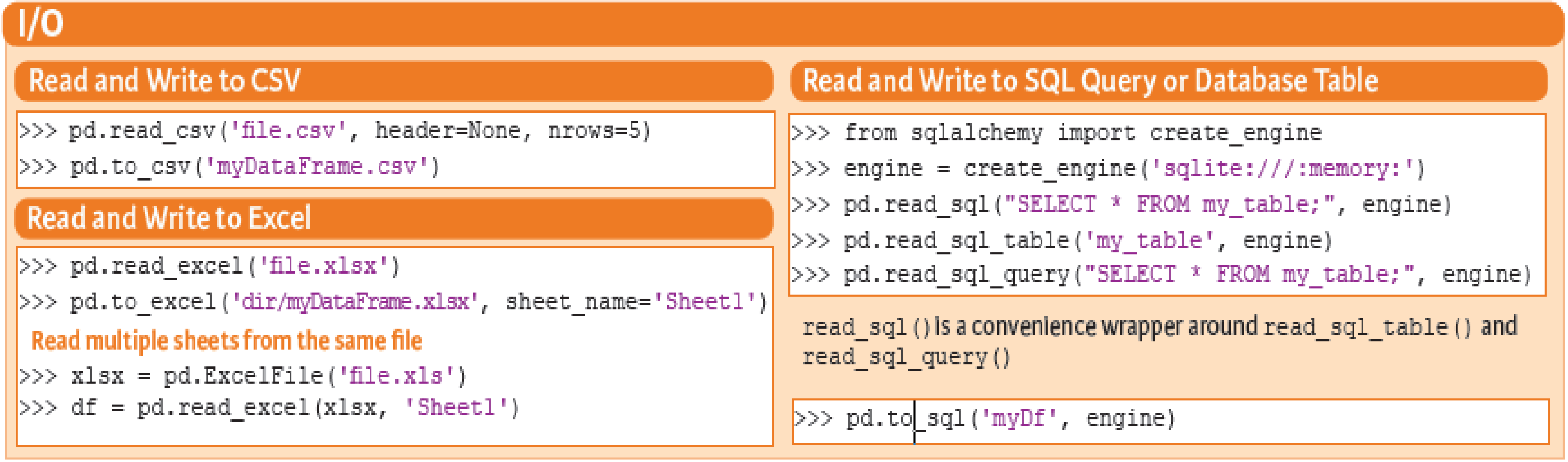
Setting

```
>>> s['a'] = 6
```

Set index a of Series s to 6

Programación Python & Librerías típicas:

- Pandas



I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

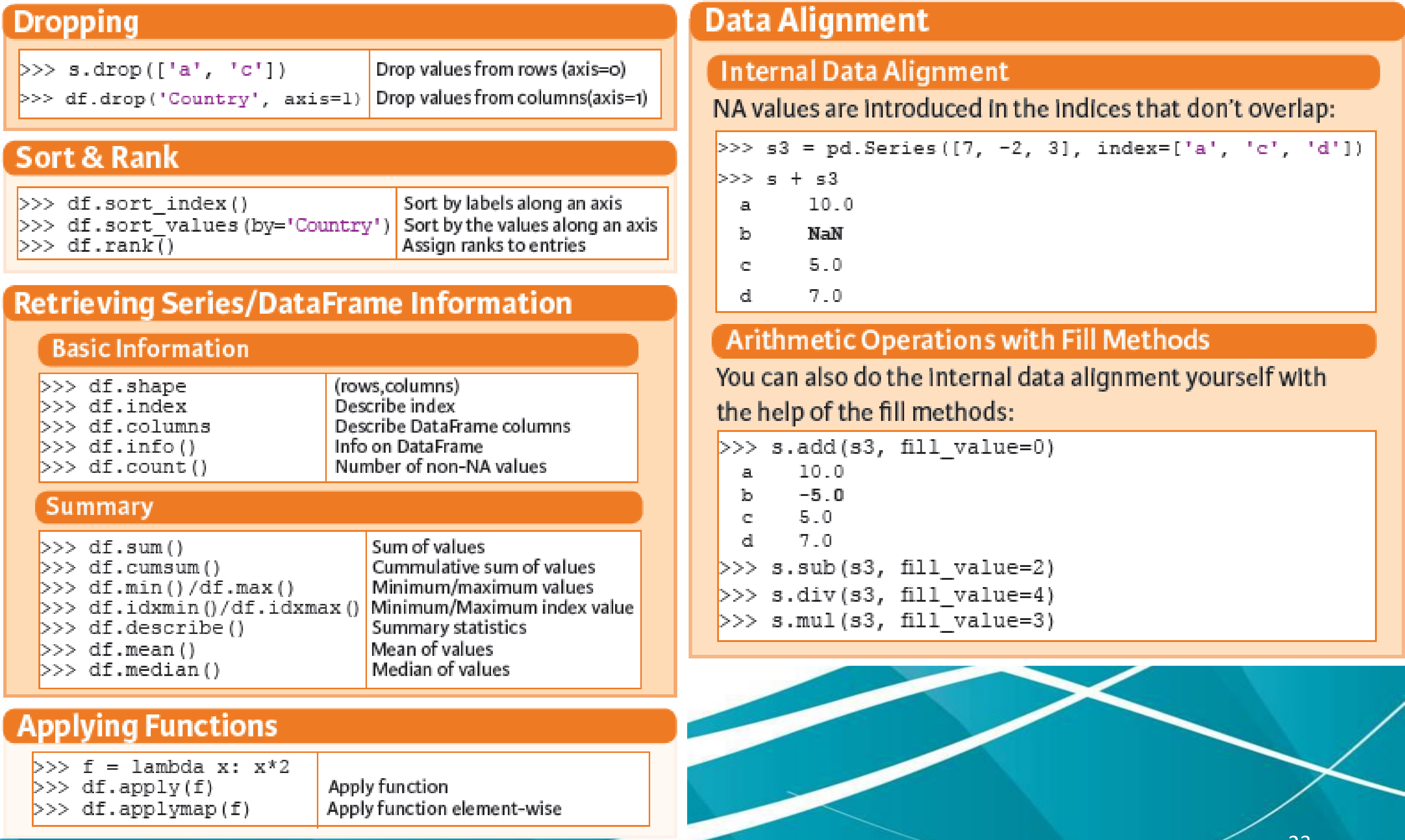
```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

read_sql() is a convenience wrapper around read_sql_table() and read_sql_query()

```
>>> pd.to_sql('myDf', engine)
```

Programación Python & Librerías típicas:

- Pandas



Dropping

>>> s.drop(['a', 'c'])	Drop values from rows (axis=0)
>>> df.drop('Country', axis=1)	Drop values from columns(axis=1)

Sort & Rank

>>> df.sort_index()	Sort by labels along an axis
>>> df.sort_values(by='Country')	Sort by the values along an axis
>>> df.rank()	Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

>>> df.shape	(rows,columns)
>>> df.index	Describe index
>>> df.columns	Describe DataFrame columns
>>> df.info()	Info on DataFrame
>>> df.count()	Number of non-NA values

Summary

>>> df.sum()	Sum of values
>>> df.cumsum()	Cummulative sum of values
>>> df.min()/df.max()	Minimum/maximum values
>>> df.idxmin()/df.idxmax()	Minimum/Maximum index value
>>> df.describe()	Summary statistics
>>> df.mean()	Mean of values
>>> df.median()	Median of values

Applying Functions

>>> f = lambda x: x*2	
>>> df.apply(f)	Apply function
>>> df.applymap(f)	Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
  a      10.0
  b      NaN
  c      5.0
  d      7.0
```

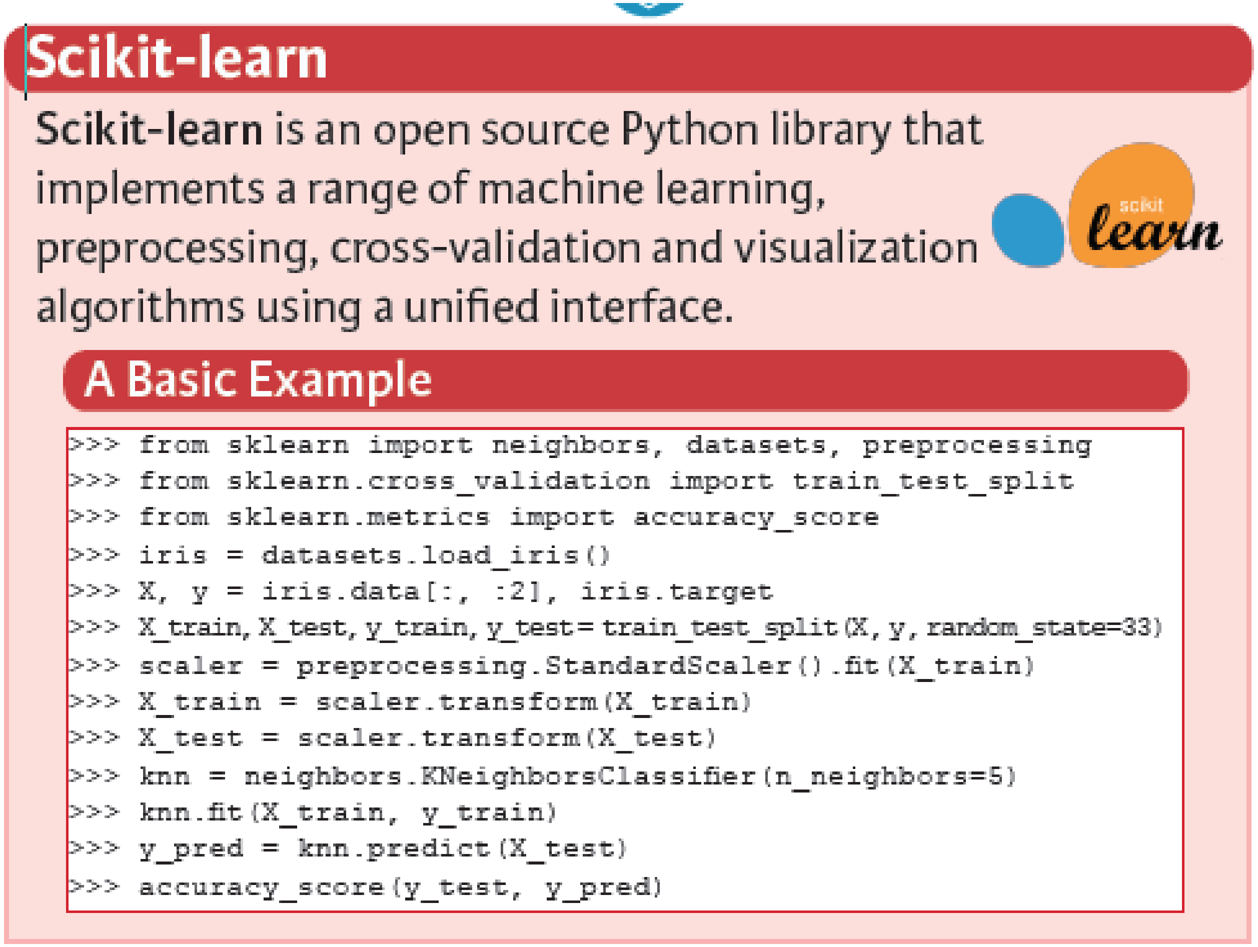
Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
  a    10.0
  b    -5.0
  c    5.0
  d    7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

Programación Python & Librerías típicas:

- Scikit-Learn



Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

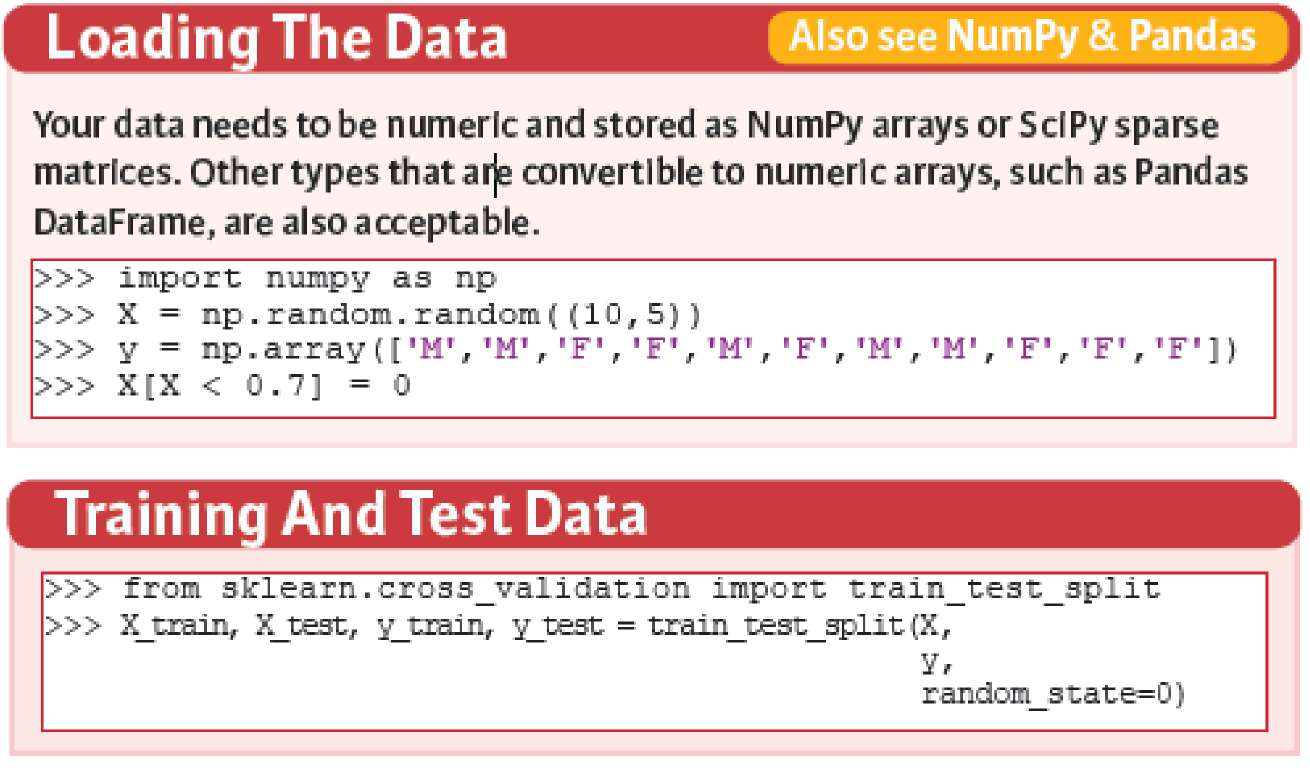
scikit
learn

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test= train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Programación Python & Librerías típicas:

- Scikit-Learn



Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

ESKERRIKASKO!!

Luka Eciolaza Echeverria
leciolaza@mondragon.edu

____

Robotics & Automation Area
Electronics and Computing Department
Mondragon University - Faculty of Engineering