

Procesamiento Digital de Señales

Introducción a Python

Facultad de Ingeniería, Universidad de Antioquia

Juan Camilo Vásquez Correa

4 de diciembre de 2019

Contenido

① Why Python

② Tools

③ Let's code

Próxima sección

① Why Python

② Tools

③ Let's code



Outline

`143ents[currentsection, currentsubsection, hideothersubsections, sectionsty
show/shaded,]`



UNIVERSIDAD
DE ANTIOQUIA
1803

Why Python



<https://twitter.com/i/status/112145227928913510>

Why Python



Guido Van Rossum

Why Python

Recuperar las palabras de un documento en ABC

```
HOW TO RETURN words document:
  PUT {} IN collection
  FOR line IN document:
    FOR word IN split line:
      IF word not in collection:
        INSERT word IN collection
  RETURN collection
```

Recuperar las palabras de un documento en Python

```
def words(document):
    collection = set()
    for line in document:
        for word in line.split():
            if word not in collection:
                collection.add(word)
    return collection
```

Why Python

IEEE.org | IEEE Xplore Digital Library | IEEE Standards | IEEE Spectrum |



The world's largest technical professional organization for the advancement of technology

IEEE
SPECTRUM

Follow on: [f](#) [t](#) [in](#) [+](#)

Engineering Topics •

Special Reports •

Blogs •

Multimedia •

The Magazine •

Professional Resources •

Search •

Careers | Computing | Software

18 Jul 2017 19:00 GMT

The 2017 Top Programming Languages

Python jumps to No. 1, and Swift enters the Top Ten

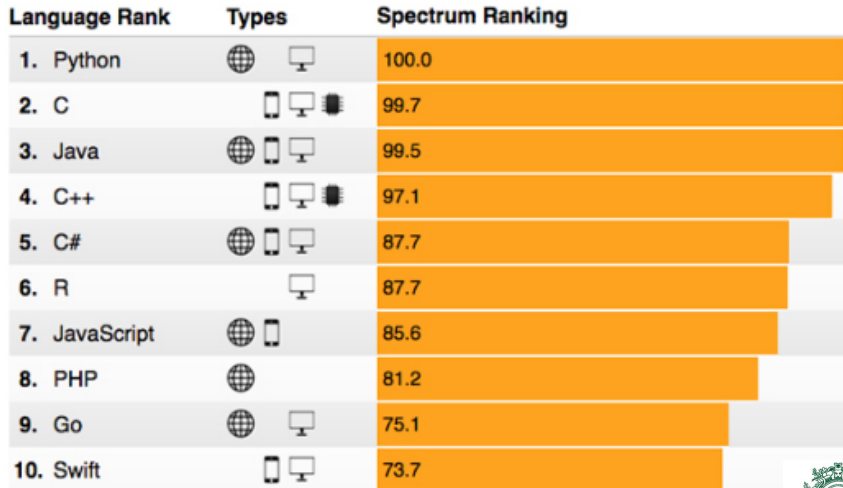
By [Stephen Cass](#)

It's summertime here at *IEEE Spectrum*, and that means it's time for our [fourth interactive ranking](#) of the top programming languages. As with all attempts to rank the use of different languages, we have to rely on various proxies for popularity. In our case, this means having data journalist [Nick Diakopoulos](#) mine and combine 12 metrics from 10 carefully chosen online sources to rank 48 languages. But where we really differ from other rankings is that our interactive allows you choose how those metrics are weighted when they are combined, letting you personalize the rankings to your needs.



UNIVERSIDAD
DE ANTIOQUIA
1803

Why Python



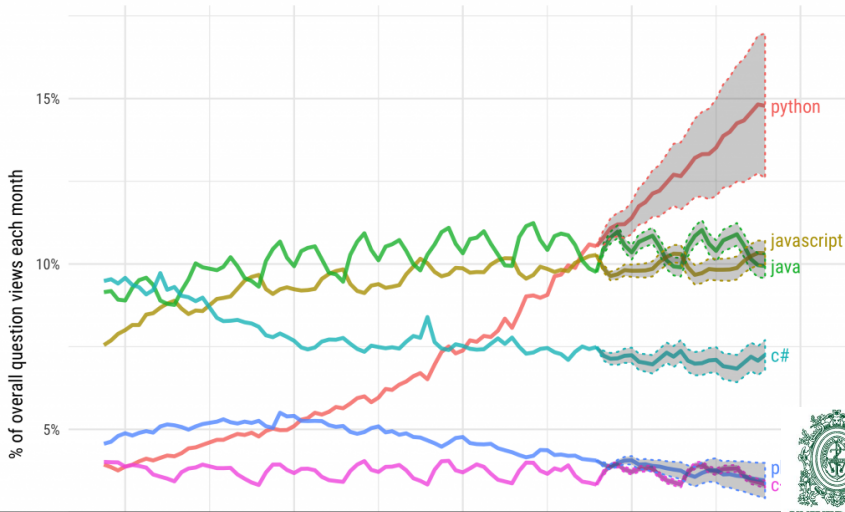
¹Source: IEEE Spectrum 2017:
<http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>



Why python

Projections of future traffic for major programming languages

Future traffic is predicted with an STL model, along with an 80% prediction interval.



UNIVERSIDAD
DE ANTIOQUIA
1803

Why python



```
print("Hello, world!")
```



ANACONDA®



UNIVERSIDAD
DE ANTIOQUIA
1803

Python3

A (very incomplete) list of properties Python

- Multi-paradigm programming language (e.g. object-oriented)
- Garbage collection for memory management
- The intention is to have highly readable code
- Indentation instead of (curly) braces
- Dynamic typing
- Data types / structures

Lists `a = []`

Dictionaries `d = {'one':1, 'two':2}`

Integer `i = 3`

Tuple `b = ()`

Strings `str = "hi"`

Float `j = 1.5`

Type verification: `type(i)`

- Due to Python's popularity there are numerous libraries available for everything one can think of.



Why python

IDE

Integrated Development Environment

Why python

Spyder

The screenshot displays the Spyder Python IDE interface. The main editor window shows a file named `Interpolation.py` with the following Python code:

```
1 *****
2 Interpolation of an N-D curve
3 From the SciPy Cookbook
4 *****
5
6 from numpy import arange, cos, linspace, pi, sin, random
7 from scipy.interpolate import splprep, splev
8
9 # make ascending spiral in 3-space
10 t=linspace(0,1.75*2*pi,100)
11
12 x = sin(t)
13 y = cos(t)
14 z = t
15
```

The Variable explorer on the right shows the following variables:

Name	Type	Size	Value
e	float	1	2.7182818284590451
pi	float	1	3.1415926535897931

The Console at the bottom shows the IPython 0.10.1 startup message:

```
IPython 0.10.1 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object'. ?object also works, ?? prints more.

Welcome to pylab, a matplotlib-based Python environment.
For more information, type 'help(pylab)'.

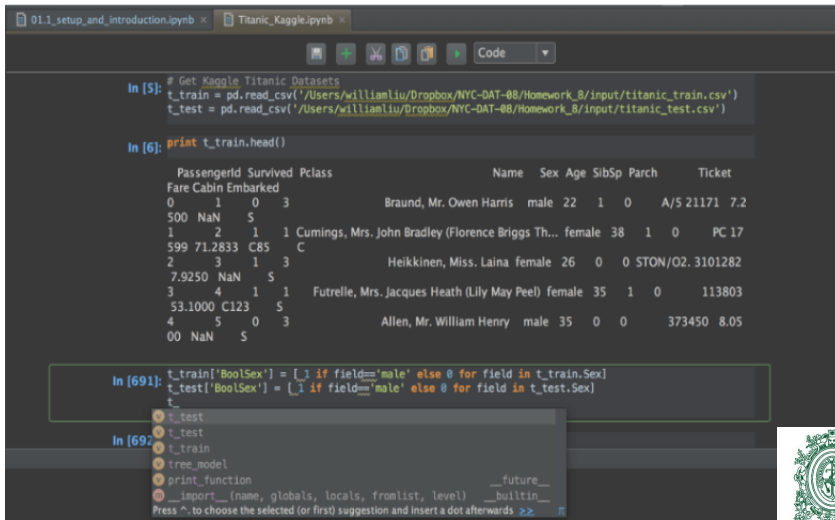
In [1]:
```



UNIVERSIDAD
DE ANTIOQUIA
1803

Why python

Pycharm



The screenshot shows the PyCharm IDE with two tabs: '01.1_setup_and_introduction.ipynb' and 'Titanic_Kaggle.ipynb'. The active tab contains the following code:

```
In [5]: # Get Kaggle Titanic Datasets
t_train = pd.read_csv('/Users/williamliu/Dropbox/NYC-DAT-08/Homework_8/input/titanic_train.csv')
t_test = pd.read_csv('/Users/williamliu/Dropbox/NYC-DAT-08/Homework_8/input/titanic_test.csv')

In [6]: print t_train.head()
```

The output of the code is a table showing the first few rows of the training dataset:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171 7.2
500	NaN	S						
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17
599	71.2833	C85	C					
2	3	1	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282
7.9250	NaN	S						
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
53.1000	C123	S						
4	5	0	Allen, Mr. William Henry	male	35	0	0	373450 8.05
00	NaN	S						

Below the code, a variable inspection window is open for 't_train'. It shows the following code snippet:

```
In [691]: t_train['BoolSex'] = [_1 if field=='male' else 0 for field in t_train.Sex]
t_test['BoolSex'] = [_1 if field=='male' else 0 for field in t_test.Sex]
t_
```

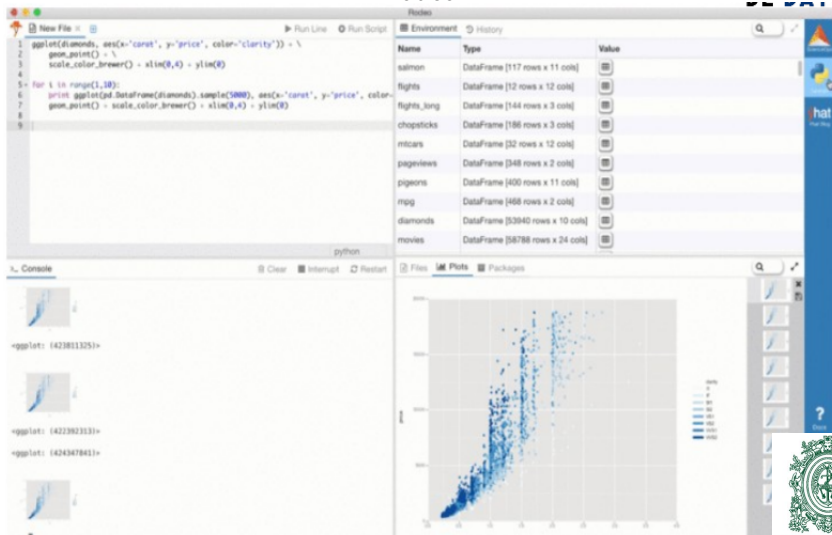
The variable inspection window shows the following variables:

- t_test
- t_train
- tree_model
- print_function
- __import__(name, globals, locals, fromlist, level)
- __future__
- __builtin__

Press ^ to choose the selected (or first) suggestion and insert a dot afterwards >>>

Why python

Rodeo



Why python

Atom

```
demo.py — /Users/lukasgeiger/Desktop
demo.py
19 # %% Display a data frame
20 df = pd.DataFrame({'A': 1.,
21                    'B': pd.Timestamp('20130102'),
22                    'C': pd.Series(1, index=list(range(4)), dtype='float32'),
23                    'D': np.array([3] * 4, dtype='int32'),
24                    'E': pd.Categorical(["test", "train", "test", "train"]),
25                    'F': 'foo'})
26 df
27
28 # %% Render Latex
29 x, y, z = sp.symbols('x, y, z')
30 f = sp.sin(x * y) + sp.cos(y * z)
31
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo



Why python

IP[y]: Notebook

spectrogram Last Checkpoint: a few seconds ago (autosaved)

IPython (Python 3)

File Edit View Insert Cell Kernel Help



Code

Cell Toolbar: None

Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#) using windowing, to reveal the frequency content of a sound signal.

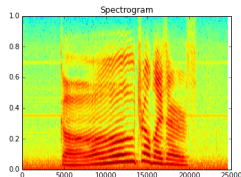
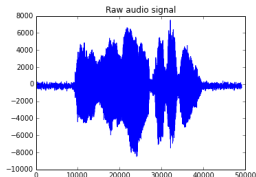
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1$$

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 spectrogram routine:

```
In [2]: %matplotlib inline
from matplotlib import pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.spectrogram(x); ax2.set_title('Spectrogram');
```



Jupyter



UNIVERSIDAD
DE ANTIOQUIA
1803

Próxima sección

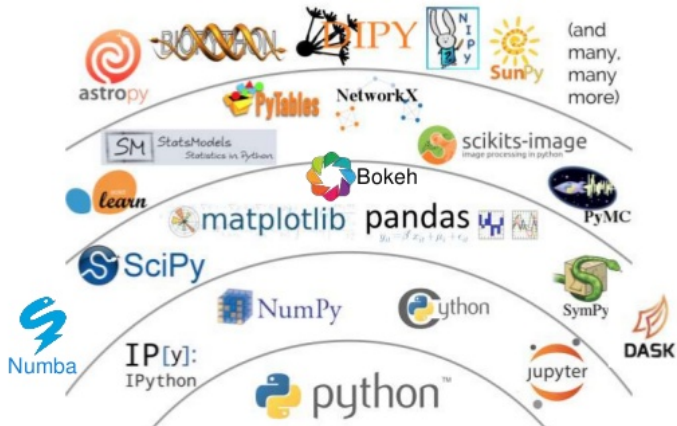
① Why Python

② Tools

③ Let's code



Tools



ANACONDA

© 2016 Continuum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools

- Jupyter
- Numpy
- Pandas
- Matplotlib
- Scipy



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Jupyter

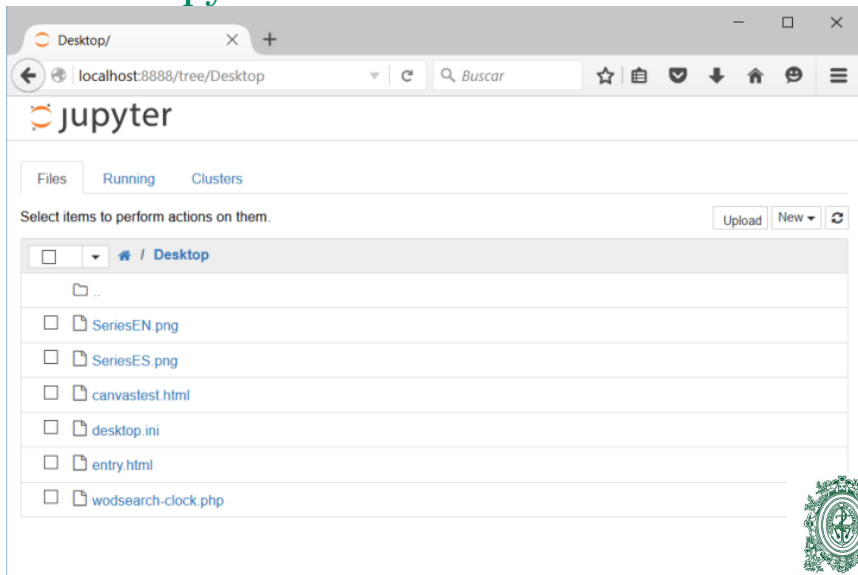
Interactive environment to create documents, code, interactive widgets, graphs, texts and equations.



2

²<http://jupyter.org/>

Tools: Jupyter



The screenshot shows a web browser window with the address bar displaying 'localhost:8888/tree/Desktop'. The page title is 'jupyter'. Below the title, there are three tabs: 'Files', 'Running', and 'Clusters'. The 'Files' tab is active. The main content area shows a file tree for the 'Desktop' directory. The tree includes a '..' folder and several files: 'SeriesEN.png', 'SeriesES.png', 'canvastest.html', 'desktop.ini', 'entry.html', and 'wodsearch-clock.php'. Each file has a checkbox to its left. At the top right of the file list, there are buttons for 'Upload', 'New', and a refresh icon. The browser's address bar also shows a search icon and the text 'Buscar'.

Tools: Jupyter

Desktop/ IPython Notebook (Jupyter) demo (unsaved changes) Title

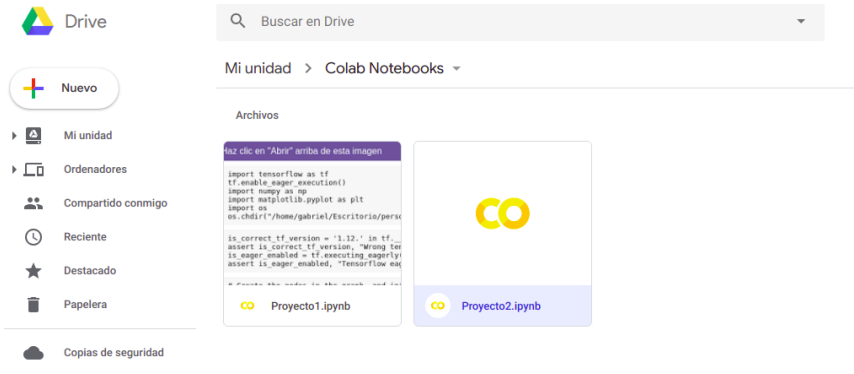
File Edit View Insert Cell Kernel Help Menu bar

Cell Toolbar: None Tool bar

Cells

In this entry, we are going to get to know one of the most usefull Python tools ever created:

Tools: Google Colab



The screenshot shows the Google Drive web interface. At the top left is the Google Drive logo. Below it is a sidebar with a 'Nuevo' button and a list of items: 'Mi unidad', 'Ordenadores', 'Compartido conmigo', 'Reciente', 'Destacado', 'Papelería', and 'Copias de seguridad'. The main area shows 'Mi unidad' selected, displaying a list of files under the heading 'Archivos'. Two files are visible: 'Proyecto1.ipynb' and 'Proyecto2.ipynb'. The 'Proyecto1.ipynb' file is expanded, showing a code editor with Python code for TensorFlow and NumPy. The code includes imports, a path definition, and version checks. The 'Proyecto2.ipynb' file is also visible but not expanded.

Drive

Buscar en Drive

Mi unidad > Colab Notebooks ▾

Archivos

haz clic en "Abrir" arriba de esta imagen

```
import tensorflow as tf
tf.enable_eager_execution()
import numpy as np
import matplotlib.pyplot as plt
import os
os.chdir("/home/gabriel/Escritorio/persa")

is_correct_tf_version = '1.12.' in tf.__version__
assert is_correct_tf_version, "Wrong tensorflow version"
is_eager_enabled = tf.executing_eagerly()
assert is_eager_enabled, "Tensorflow not running in eager mode"
```

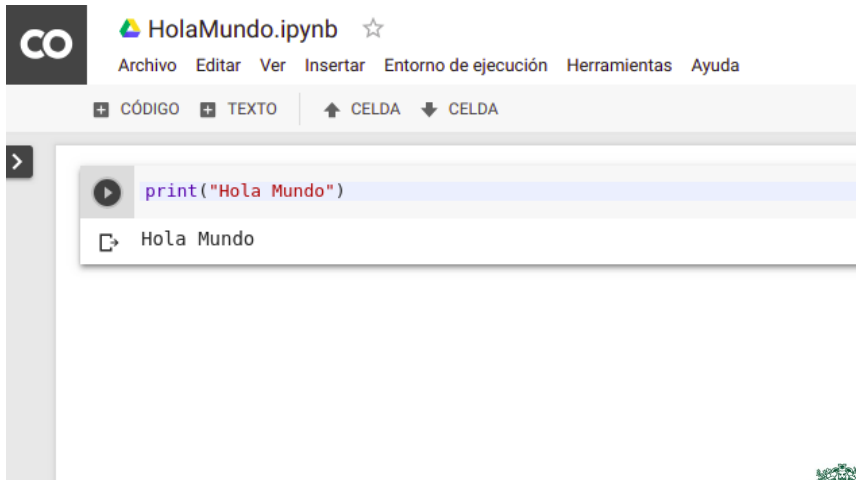
Proyecto1.ipynb

Proyecto2.ipynb

5

⁵<https://colab.research.google.com/notebooks/welcome.ipynb>

Tools: Google Colab



6

⁶<https://colab.research.google.com/notebooks/welcome.ipynb>

Tools: List processing

- Lists are just like the arrays, declared in other languages.
- Lists need not be homogeneous always which makes it a most powerful tool in Python.
- A single list may contain DataTypes like Integers, Strings, as well as Objects.
- Lists are also very useful for implementing stacks and queues.

Tools: List processing

```
1 # Creating a List
2 List = []
3 print("Intial blank List: ")
4 print(List)
5
6 # Creating a List with
7 # the use of a String
8 List = ['GeeksForGeeks']
9 print("\nList with the use of String: ")
10 print(List)
11
12 # Creating a List with
13 # the use of multiple values
14 List = ["Geeks", "For", "Geeks"]
15 print("\nList containing multiple values: ")
16 print(List[0])
17 print(List[2])
```



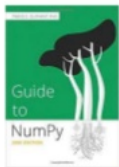
Tools: List processing

```
1 # Creating a List with
2 # the use of Numbers
3 # (Having duplicate values)
4 List = [1, 2, 4, 4, 3, 3, 3, 6, 5]
5 print("\nList with the use of Numbers: ")
6 print(List)
7
8 # Creating a List with
9 # mixed type of values
10 # (Having numbers and strings)
11 List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
12 print("\nList with the use of Mixed Values: ")
13 print(List)
```

Tools: Numpy

NumPy: an Array Extension of Python

- Data: the array object
 - slicing and shaping
 - data-type map to Bytes
- Fast Math (ufuncs):
 - vectorization
 - broadcasting
 - aggregations



© 2016 Continuum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Numpy

- Array, matrix, vectorial operations.
- **Import** numpy **as** np

[0]	[1]	[2]	[3]	[4]
2	5	1	3	4



NumPy

7

⁷<http://www.numpy.org/>

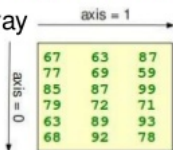


UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Numpy

NumPy Examples

2d array

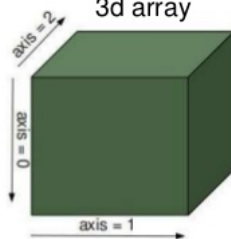


```
1 import numpy as np
2
3 x = np.array([[67, 63, 87],
4               [77, 69, 59],
5               [85, 87, 89],
6               [79, 72, 71],
7               [63, 89, 93],
8               [68, 92, 78]])
9 print x.sum(axis=0), x.sum(axis=1)
```

[439 472 477]

[217 205 261 222 245 238]

3d array



```
12 y = 3*np.random.randn(10,20,30)+10
13 print y.mean(), y.std()
```

9.98330639789 2.96677717122



© 2016 Continuum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Numpy

NumPy Slicing (Selection)

```
>>> a[0,3:5]
array([3, 4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2, 12, 22, 32, 42, 52])

>>> a[2::2,::2]
array([[20, 22, 24],
       [40, 42, 44]])
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55



© 2016 Continuum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Numpy

```
1 # manejo de arreglos
2
3 import numpy as np #libreria estandar para manejo de datos y ←
   senales en Python
4
5 a = np.zeros((2,2)) # Crea un arreglo tipo matriz de ceros
6 print(a)           # Prints "[[ 0.  0.]
7                     #           [ 0.  0.]]"
8 b = np.ones((1,2)) # Crea un arreglo tipo vector fila de ←
   unos
9 print(b)           # Prints "[[ 1.  1.]]"
10
11 c = np.full((2,2), 7) # Crea un arreglo tipo matriz de un ←
   valor constante
12 print(c)           # Prints "[[ 7.  7.]
13                     #           [ 7.  7.]]"
14 c = 7*np.ones((2,2)) # Otra forma
15 print (c)
16
17 d = np.eye(2)      # Crea un arreglo con la matriz de
```



Tools: Numpy

```
1 e = np.random.random((2,2)) # Crea un arreglo tipo matriz de ↵
    numeros aleatorios entre 0 y 1
2 print (e)                  # print "[[ 0.91940167  0.08143941]
3                             #          [ 0.68744134  0.87236687]]"
4 print (e[0,1])              # Seleccionar un elemento
5 print (e[0,:])              # Seleccionar la primera fila
6 print (e[:,1])              # Seleccionar la segunda ↵
    columna
7
8 xrange=np.arange(20) # Crea un arreglo de numeros consecutivos↵
    entre 0 y 19
9 print (xrange)              # print "[0 1 2 3 4 5... 19]"
10 # Para seleccionar partes del arreglo use los siguientes ↵
    comandos
11 print (xrange[0:5])        # print "[0 1 2 3 4]"
12 print (xrange[10])         # print [10]
13
14 A=np.random.random(100)
15 media=np.mean(A) # calcula la media de un arreglo A
16 maxA=np.max(A) # calcula el maximo valor de un arreglo
```



Tools: Pandas

Python has long been great for data munging and preparation, but less so for data analysis and modeling. pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python

pandas

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



⁸

⁸<https://pandas.pydata.org/>

Tools: Pandas

Data structure and tools for data analysis.

- DataFrames
- GroupBy, merge, join
- Data reading and writing
- **Import** pandas **as** pd

pandas

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



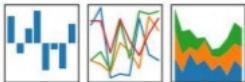
⁹

⁹<https://pandas.pydata.org/>

Tools: Pandas

pandas

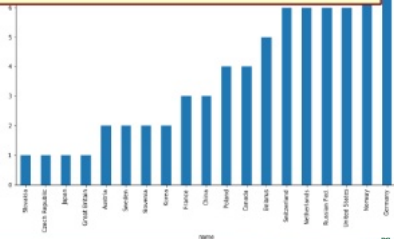
$$y_{it} = \beta^T x_{it} + \mu_i + \epsilon_{it}$$



Easy Data Wrangling

```
medals = pd.read_csv('data/medals.csv', index_col='name')
medals.head()
gold = medals['medal'] == 'gold'
won = medals['count'] > 0
medals.loc[gold & won, 'count'].sort_values().plot(kind='bar', figsize=(12,8))
```

	count	medal	country
name			
Australia	1	bronze	AUS
Australia	2	silver	AUS
Australia	0	gold	AUS
Austria	1	bronze	AUT
Austria	6	silver	AUT



ANACONDA

© 2016 Continuum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Matplotlib

Graphics in Python

- Frequency diagram
- Histograms
- Scatter-plots
- 3D graphics
- Surface

Import matplotlib

matplotlib



10

¹⁰<https://matplotlib.org/>

Tools: Matplotlib

matplotlib

a powerful plotting engine

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt

np.random.seed(0)

# example data
mu = 100 # mean of distribution
sigma = 15 # standard deviation of distribution
x = mu + sigma * np.random.randn(437)

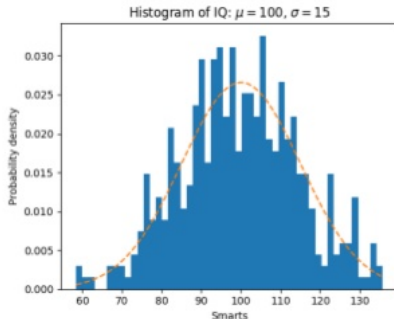
num_bins = 50

fig, ax = plt.subplots()

# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, normed=1)

# add a 'best fit' line
y = mlab.normpdf(bins, mu, sigma)
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
ax.set_title(r'Histogram of IQ:  $\mu=100$ ,  $\sigma=15$ ')

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```



© 2016 Corenum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Matplotlib



a powerful plotting engine

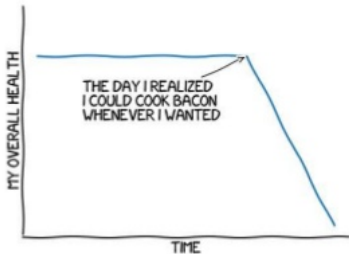
```
import matplotlib.pyplot as plt
import numpy as np

with plt.xkcd():
    fig = plt.figure()
    ax = fig.add_axes([0.1, 0.2, 0.8, 0.7])
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    plt.xticks([])
    plt.yticks([])
    ax.set_ylim([-30, 10])
    data = np.ones(100)
    data[70:] -= np.arange(30)

    plt.annotate('THE DAY I REALIZED\nI COULD\nCOOK BACON\nWHENEVER I WANTED',
                xy=(70, 1),
                arrowprops=dict(arrowstyle='->'),
                xytext=(15, -10))

    plt.plot(data)

    plt.xlabel('time')
    plt.ylabel('my overall health')
    fig.text(0.5, 0.05, "Stove Ownership" from
    xkcd by Randall Monroe',
            ha='center')
```



"STOVE OWNERSHIP" FROM XKCD BY RANDALL MONROE



© 2016 Continuum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Matplotlib

matplotlib

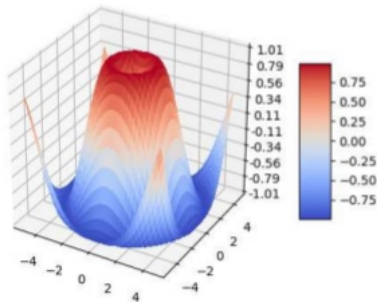
a powerful plotting engine

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator,
FormatStrFormatter
import numpy as np

fig = plt.figure()
ax = fig.gca(projection='3d')
# Make data.
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%
.02f'))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```



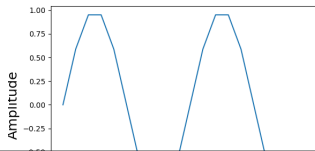
© 2016 Continuum Analytics - Confidential & Proprietary



UNIVERSIDAD
DE ANTIOQUIA
1803

Tools: Matplotlib

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 f=1.0 # Signal frequency
5 fs=10.0 # Sampling frequency
6 t=np.arange(0, 2.0, 1.0/fs) # Vector de tiempo
7 x = np.sin(2*np.pi*f*t)
8 plt.plot(t,x)
9 plt.xlabel('Time',fontsize=18)
10 plt.ylabel('Amplitude',fontsize=18)
11 plt.show()
```



UNIVERSIDAD
DE ANTIOQUIA
1803

File managment

Load Text files containing vector or matrices data.

```
1 import numpy as np
2 data=np.loadtxt("file.txt")
```

Write a Text file with vector or matrices data.

```
1 Data=np.random.rand((100,20))
2 np.savetxt("file.txt", Data)
```

File management

Loading and plotting a Text file that contains a signal.

```
1 data=np.loadtxt("PrecioDolar.txt")
2 signal=data[:,3]
3 t=np.arange(len(signal))
4 plt.plot(t,signal)
5 plt.xlabel("samples")
6 plt.ylabel("TRM COP -US")
7 plt.show()
```

```
1 Ts=1.0/365
2 t=np.arange(1992, 2016+244*Ts+5*Ts, Ts)
3 plt.plot(t,signal)
4 plt.xlabel("Fecha")
5 plt.ylabel("TRM COP-US")
6 plt.show()
```

File managment

Load data from excel files (.csv)

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 data=pd.read_csv("DataSiata.csv")
4 signal=data["pm10"]
5 t=data["index"]
6 plt.plot(t,signal)
7 plt.xlabel("samples")
8 plt.ylabel("Concentracion particulas PM10")
9 plt.show()
```

File management

Load audio signals (.wav)

```
1  from scipy.io.wavfile import read
2  import matplotlib.pyplot as plt
3  fs,signal=read("098_readtext_PCGITA.wav")
4  t=np.arange(0,len(signal)/fs, 1./fs)
5  plt.plot(t,signal)
6  plt.xlabel("Time [s]")
7  plt.ylabel("Amplitude")
8  plt.show()
```

File management

Write signal into .wav file

```
1  from scipy.io.wavfile import write
2  import numpy as np
3
4  fs=44100
5  t=np.arange(0,10,1.0/fs)
6  signal=np.sin(2*np.pi*fs*(t**3))
7
8  signal2=np.asarray(signal*2**15, dtype=np.int16)
9  write("signal.wav", fs, signal2)
```


Functions

If a sequence of commands is used more often, a function is helpful:

```
1 def say_hello():  
2     print("hello")
```

```
1 def say_hello_to(name):  
2     print("hello " + name)
```

Call the function like this:

```
1 say_hello()  
2 say_hello_to("Donald")
```

Functions

```
1 def energy(signal):  
2     s2=signal**2  
3     energy=sum(s2)  
4     return(energy)
```

Call the function like this:

```
1 signal=[2,4,6,0,0,1,2,4,0]  
2 E=energy(signal)  
3 print("Energy="+str(E))
```

Important: Make sure you get the indentation right, especially when you do cut and paste (so your code works).



Próxima sección

① Why Python

② Tools

③ Let's code



Let's code



Terminal: **jupyter notebook**

Thanks!



jcamilo.vasquez@udea.edu.co