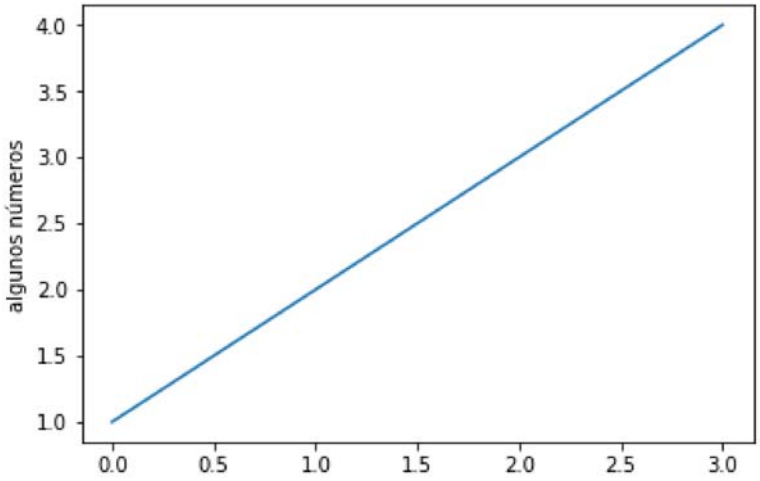
Matplotlib

Matplotlib es una librería de trazado de gráficos que genera figuras de diversos tipos y formatos con gran calidad. La mejor manera de introducirse en las posibilidades de esta librería es mediante ejemplos.

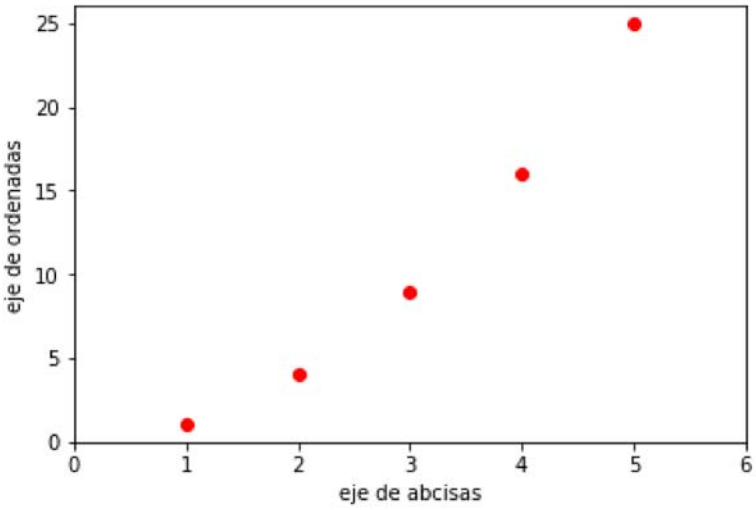
In [2]:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('algunos números')
plt.show()
```



In [2]:

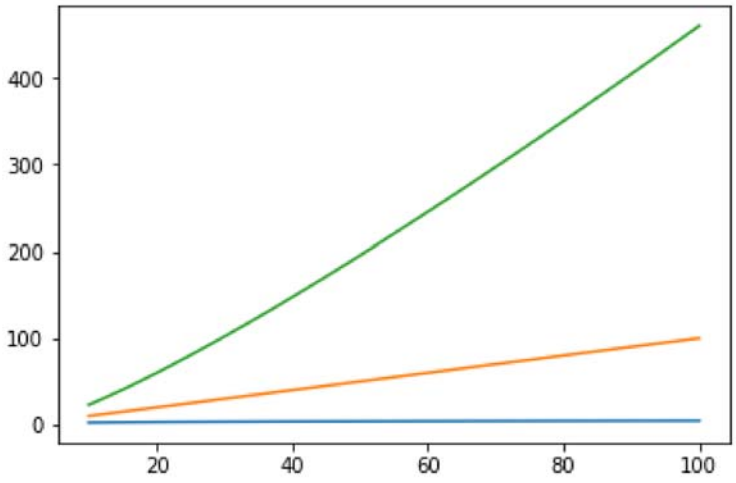
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4, 5], [1, 4, 9, 16, 25], 'ro')
plt.axis([0, 6, 0, 26])      # x <- [0. 6], y <- [0, 26]
plt.xlabel('eje de abcisas')
plt.ylabel('eje de ordenadas')
plt.show()
```



Varias gráficas juntas usando arrays

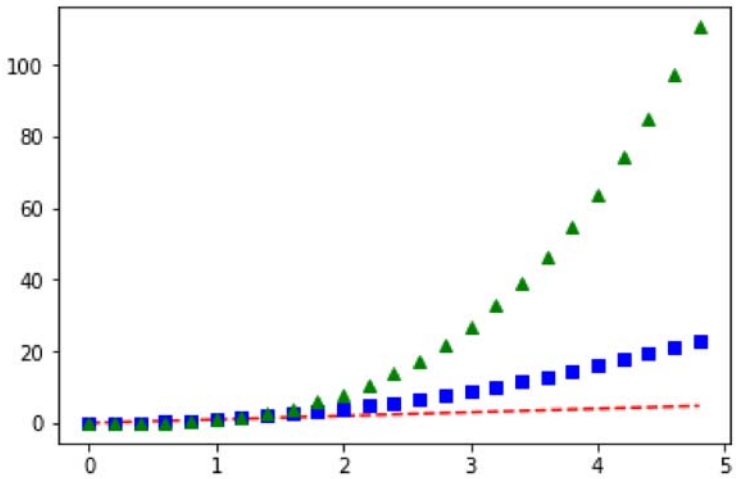
In [3]:

```
import numpy as np  # para usar arrays.

def nlogn(x):
    return np.log(x) * x
points = np.linspace(10,100,100)                                    # x1 x2 delta_x
plt.plot(points, np.log(points), points, points, points, nlogn(points))      # Las tres grá
plt.savefig('./figuras/plot.png', dpi=600) # hacer antes que show
plt.show()
```



In [4]:

```
# Tres gráficas juntas a pasos discretos: [0.0, 5.0] a pasos de 0.2
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')  # x1, y1, (color1, forma1), ...
plt.show()
```



Una figura a trozos

In [5]:

```
# https://stackoverflow.com/questions/14000595/graphing-an-equation-with-matplotlib

import numpy as np
import matplotlib.pyplot as plt

def graph(formula, x_range):
    x = np.array(x_range)
    y = formula(x)
    plt.plot(x, y)
    plt.show()

def my_formula(x):
    return x**3 - 50*x

def my_graph():
    graph(my_formula, range(-7, 8))

my_graph()
```

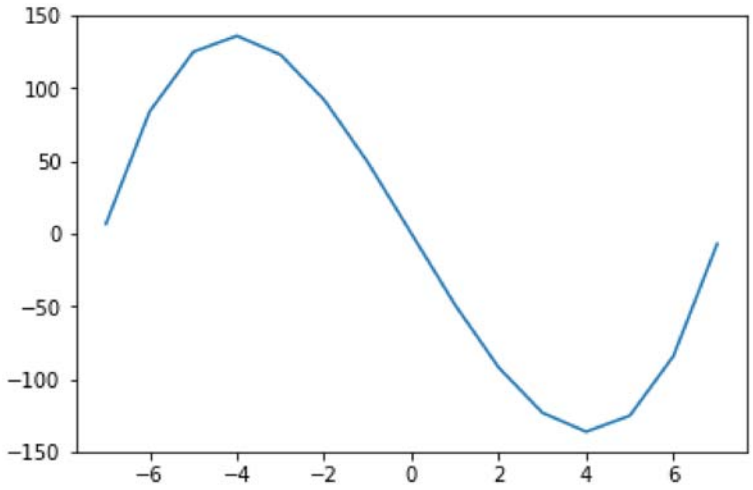
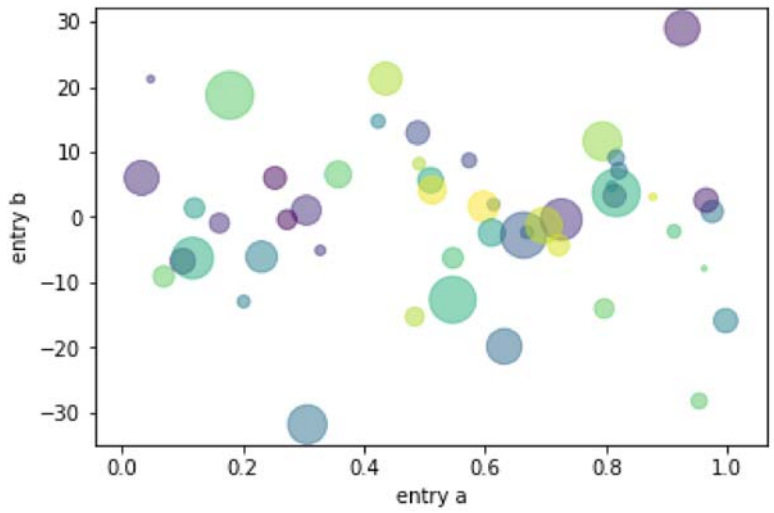


Gráfico de dispersión

In [6]:

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

N = 50
x = np.random.rand(N)
y = x + 10 * np.random.randn(N)
color = np.random.randint(0, N, N)
tamanno = np.abs(np.random.randn(N)) * 250

plt.scatter(x, y, c=color, s=tamanno, alpha=0.5)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```



In [7]:

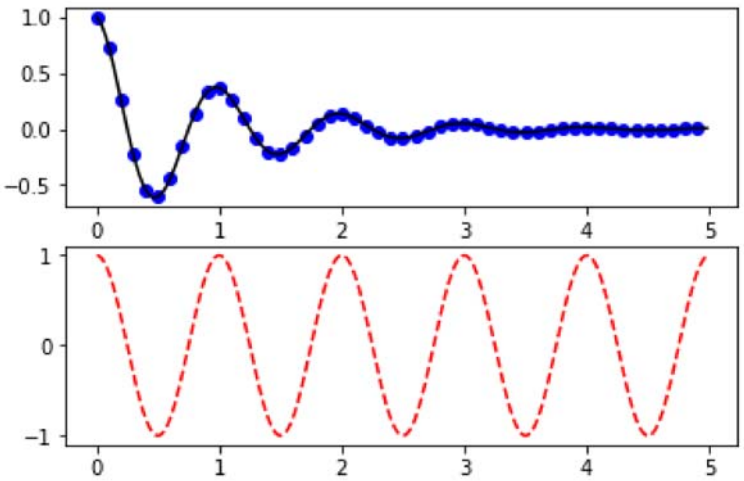
```
help(plt.scatter)
```

```
Help on function scatter in module matplotlib.pyplot:

scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=Non
e, vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, *,
data=None, **kwargs)
    A scatter plot of *y* vs *x* with varying marker size and/or color.

    Parameters
    ----------
    x, y : array_like, shape (n, )
        The data positions.

    s : scalar or array_like, shape (n, ), optional
        The marker size in points**2.
        Default is ``rcParams['lines.markersize'] ** 2``.

    c : color, sequence, or sequence of color, optional
        The marker color. Possible values:
```

Representación de varias gráficas separadas

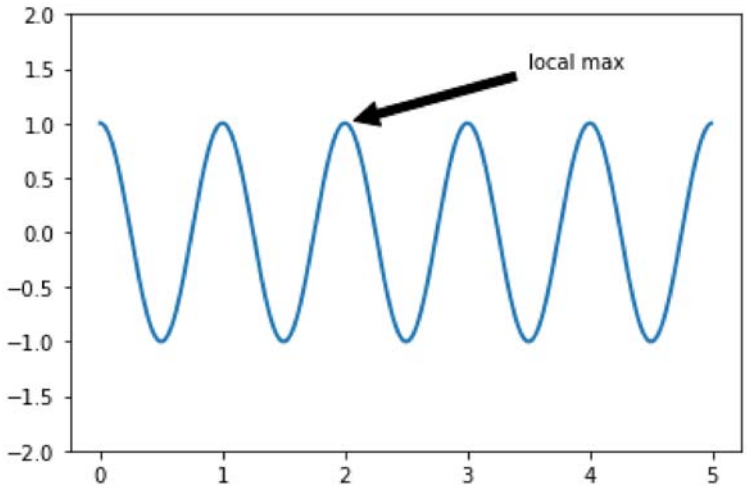
In [8]:

```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```



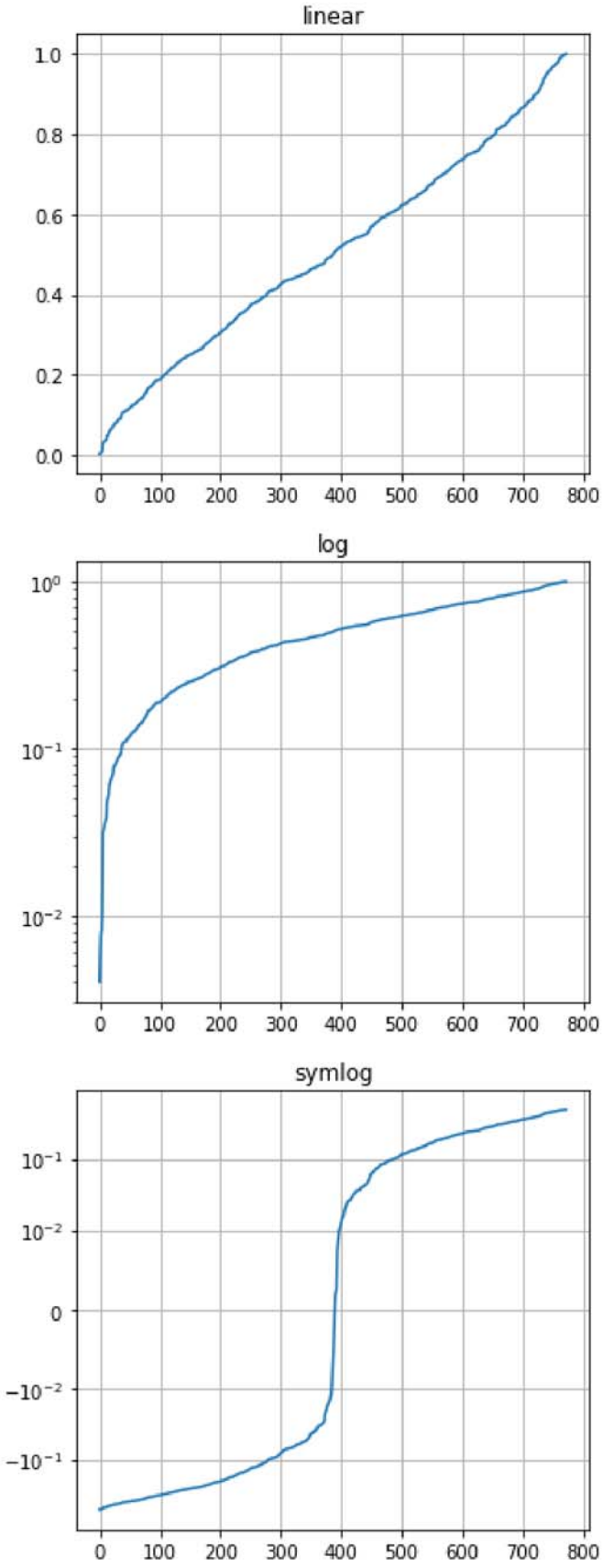
In [9]:

```
ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max', xy=(2, 1), # La punta,
             xytext=(3.5, 1.5), #Posición del texto,
             arrowprops=dict(facecolor='black', shrink=0.05),
             )

plt.ylim(-2, 2)
plt.show()
```



In [10]:

```
# make up some data in the interval ]0, 1[
y = np.random.normal(loc=0.5, scale=0.4, size=1000)
y = y[(y > 0) & (y < 1)]
y.sort()
x = np.arange(len(y))

plt.figure(figsize=(5,15)) # tamaño en pulgadas
# linear
plt.subplot(311)
plt.plot(x, y)
plt.yscale('linear')
plt.title('linear')
plt.grid(True)

# log
plt.subplot(312)
plt.plot(x, y)
plt.yscale('log')
plt.title('log')
plt.grid(True)

# symmetric log
plt.subplot(313)
plt.plot(x, y - y.mean())
plt.yscale('symlog', linthreshy=0.01)
plt.title('symlog')

plt.grid(True)
plt.show()
```


linear
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700
800
log
10^{0}
10^{-1}
10^{-2}
0
100
200
300
400
500
600
700
800
symlog
10^{-1}
10^{-2}
0
-10^{-2}
-10^{-1}
0
100
200
300
400
500
600
700
800

In [11]:

```
help(plt.figure)
```

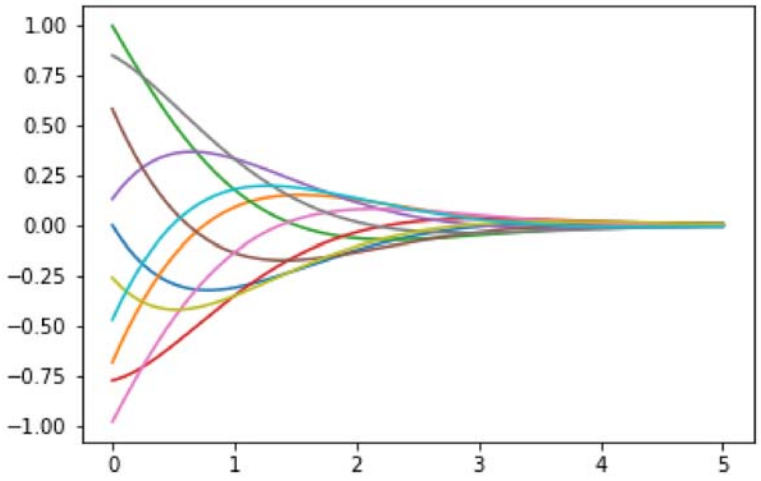
```
Help on function figure in module matplotlib.pyplot:

figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, fra
meon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **kw
args)
    Create a new figure.

    Parameters
    ----------

    num : integer or string, optional, default: None
        If not provided, a new figure will be created, and the figure number
        will be incremented. The figure objects holds this number in a `numb
er`
        attribute.
        If num is provided, and a figure with this id already exists, make
        it active, and returns a reference to it. If this figure does not
        exists, create it and returns it.
        If num is a string, the window title will be set to this figure's
        `num`.

    figsize : (float, float), optional, default: None
        width, height in inches. If not provided, defaults to
        :rc:`figure.figsize` = ``[6.4, 4.8]``.

    dpi : integer, optional, default: None
        resolution of the figure. If not provided, defaults to
        :rc:`figure.dpi` = ``100``.

    facecolor :
        the background color. If not provided, defaults to
        :rc:`figure.facecolor` = ``'w'``.

    edgecolor :
        the border color. If not provided, defaults to
        :rc:`figure.edgecolor` = ``'w'``.

    frameon : bool, optional, default: True
        If False, suppress drawing the figure frame.

    FigureClass : subclass of `~matplotlib.figure.Figure`
        Optionally use a custom `.Figure` instance.

    clear : bool, optional, default: False
        If True and the figure already exists, then it is cleared.

    Returns
    -------
    figure : `~matplotlib.figure.Figure`
        The `.Figure` instance returned will also be passed to new_figure_ma
nager
        in the backends, which allows to hook custom `.Figure` classes into
the
        pyplot interface. Additional kwargs will be passed to the `.Figure`
        init function.
```

```
    Notes
    -----
    If you are creating many figures, make sure you explicitly call
    :func:`.pyplot.close` on the figures you are not using, because this wil
l
    enable pyplot to properly clean up the memory.

    `~matplotlib.rcParams` defines the default values, which can be modified
    in the matplotlibrc file.
```

Representación de una función de dos variables

In [12]:

```
# La segunda variable se está simulando a base de elegir una decena de valores de la misma:

def f(z,t):
    return np.exp(-z)*np.sin(t-z)

z = np.linspace(0,5,3001)
t = np.arange(0,40000,4000)     # array([0,  4000,  8000, 12000, 16000, 20000, 24000, 28000

for tval in t:
    plt.plot(z, f(z, tval))
plt.show()
```

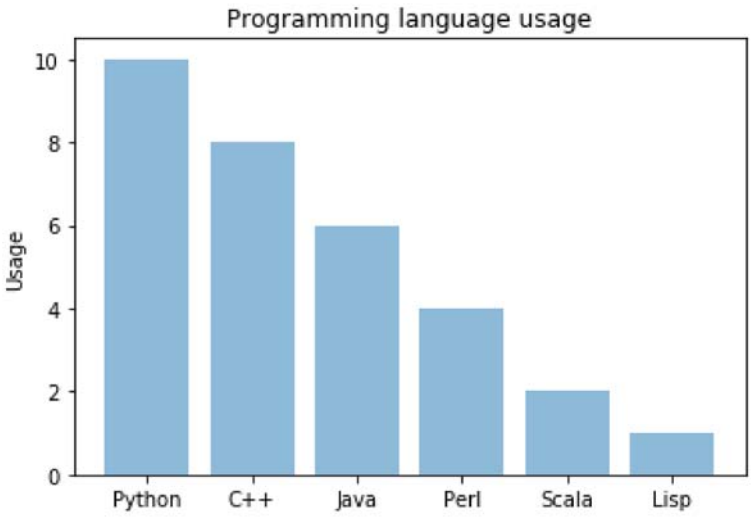


Diagramas de barras

https://pythonspot.com/matplotlib-bar-chart/ (https://pythonspot.com/matplotlib-bar-chart/)

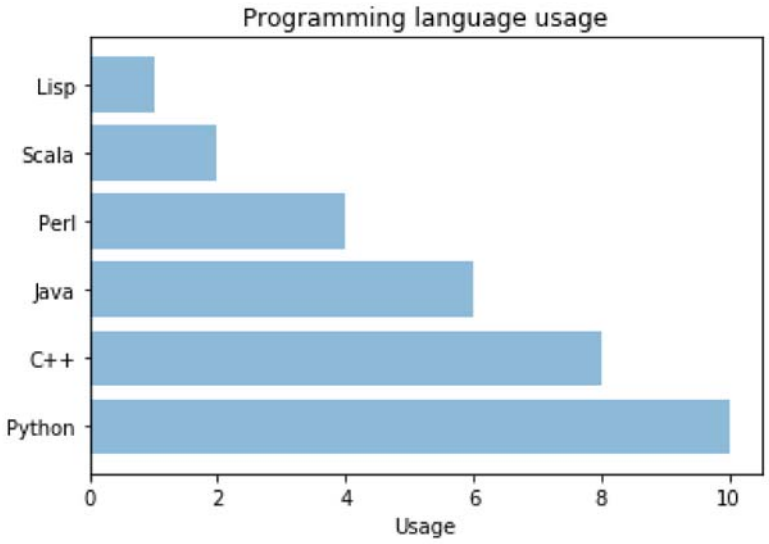
In [13]:

```
objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Usage')
plt.title('Programming language usage')

plt.show()
```



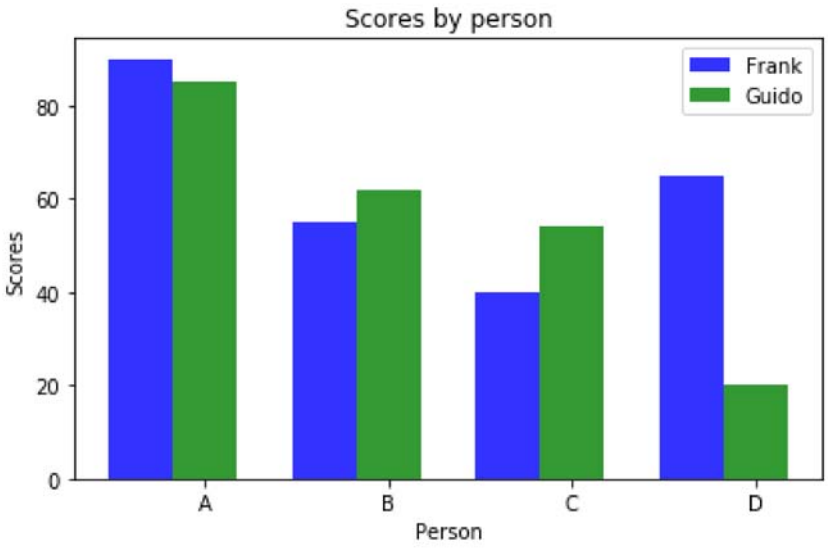
In [14]:

```
objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]

plt.barh(y_pos, performance, align='center', alpha=0.5)
plt.yticks(y_pos, objects)
plt.xlabel('Usage')
plt.title('Programming language usage')

plt.show()
```



In [15]:

```
# data to plot
n_groups = 4
means_frank = (90, 55, 40, 65)
means_guido = (85, 62, 54, 20)

# create plot
fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8

rects1 = plt.bar(index, means_frank, bar_width,
alpha=opacity,
color='b',
label='Frank')

rects2 = plt.bar(index + bar_width, means_guido, bar_width,
alpha=opacity,
color='g',
label='Guido')

plt.xlabel('Person')
plt.ylabel('Scores')
plt.title('Scores by person')
plt.xticks(index + bar_width, ('A', 'B', 'C', 'D'))
plt.legend()

plt.tight_layout()
plt.show()
```



Matplotlib

Matplotlib es una librería con muchas posibilidades, muchas más que las vistas en esta breve colección de ejemplos. Es imposible verlas todas, pero podemos tener una idea de dichas posibilidades y una especie de índice viendo las siguientes figuras:

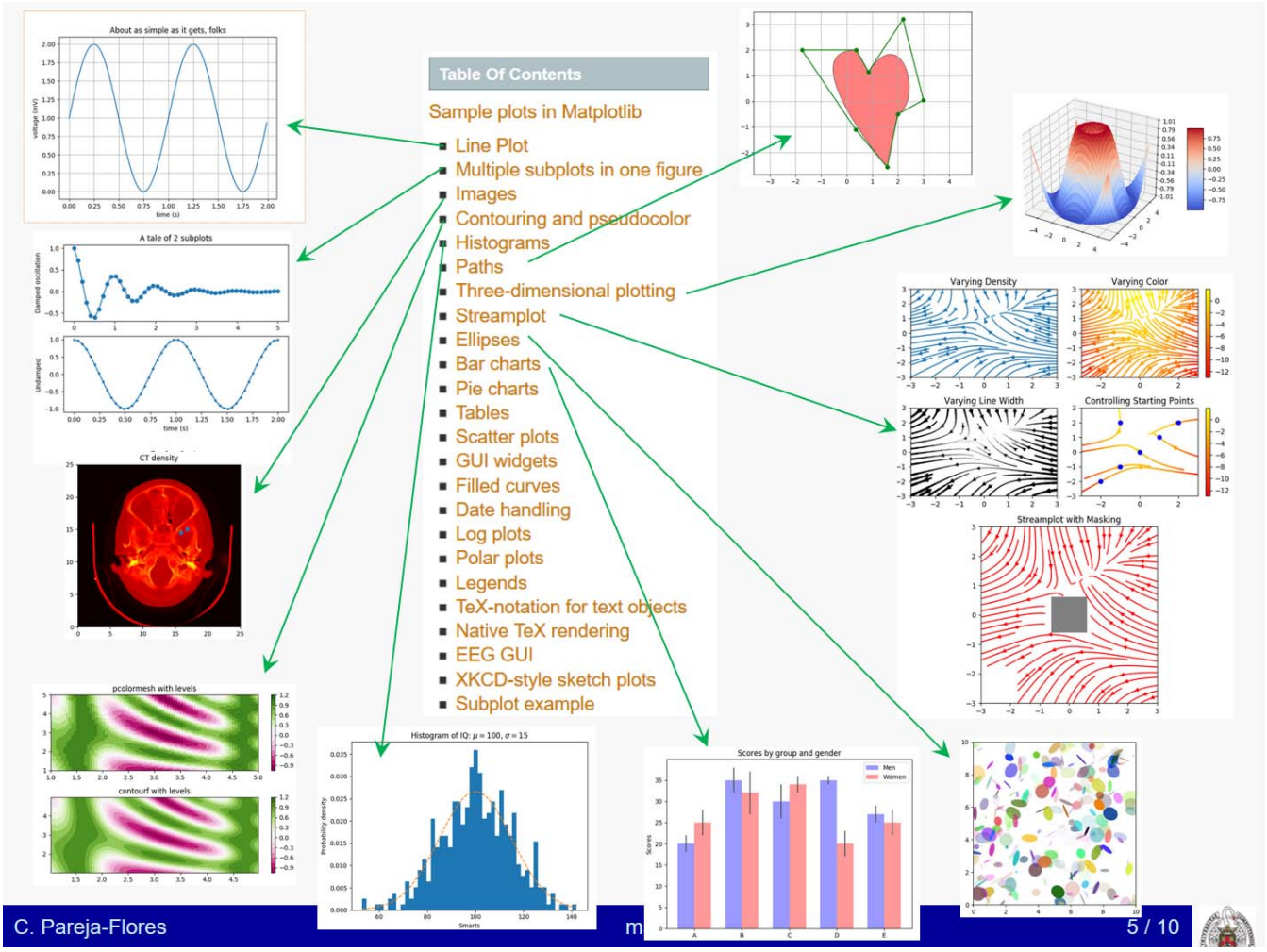


Table Of Contents
Sample plots in Matplotlib
Line Plot
Multiple subplots in one figure
Images
Contouring and pseudocolor
Histograms
Paths
Three-dimensional plotting
Streamplot
Ellipses
Bar charts
Pie charts
Tables
Scatter plots
GUI widgets
Filled curves
Date handling
Log plots
Polar plots
Legends
TeX-notation for text objects
Native TeX rendering
EEG GUI
XKCD-style sketch plots
Subplot example
About as simple as it gets, folks
A tale of 2 subplots
CT density
pcolormesh with levels
contourf with levels
Varying Density
Varying Color
Varying Line Width
Controlling Starting Points
Streamplot with Masking
Histogram of IQ: $\mu = 100$, $\sigma = 15$
Scores by group and gender
Men
Women
C. Pareja-Flores
5 / 10

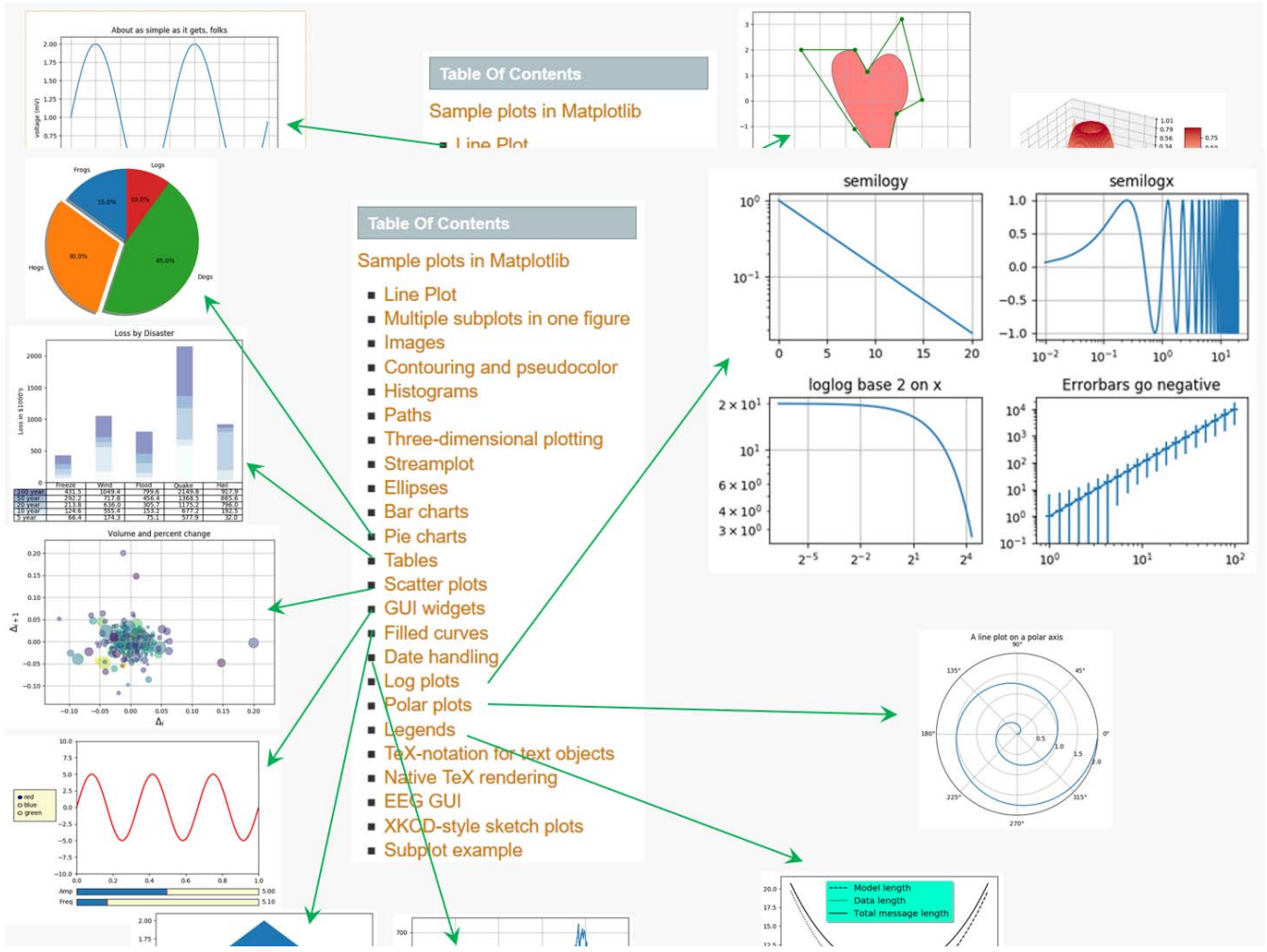
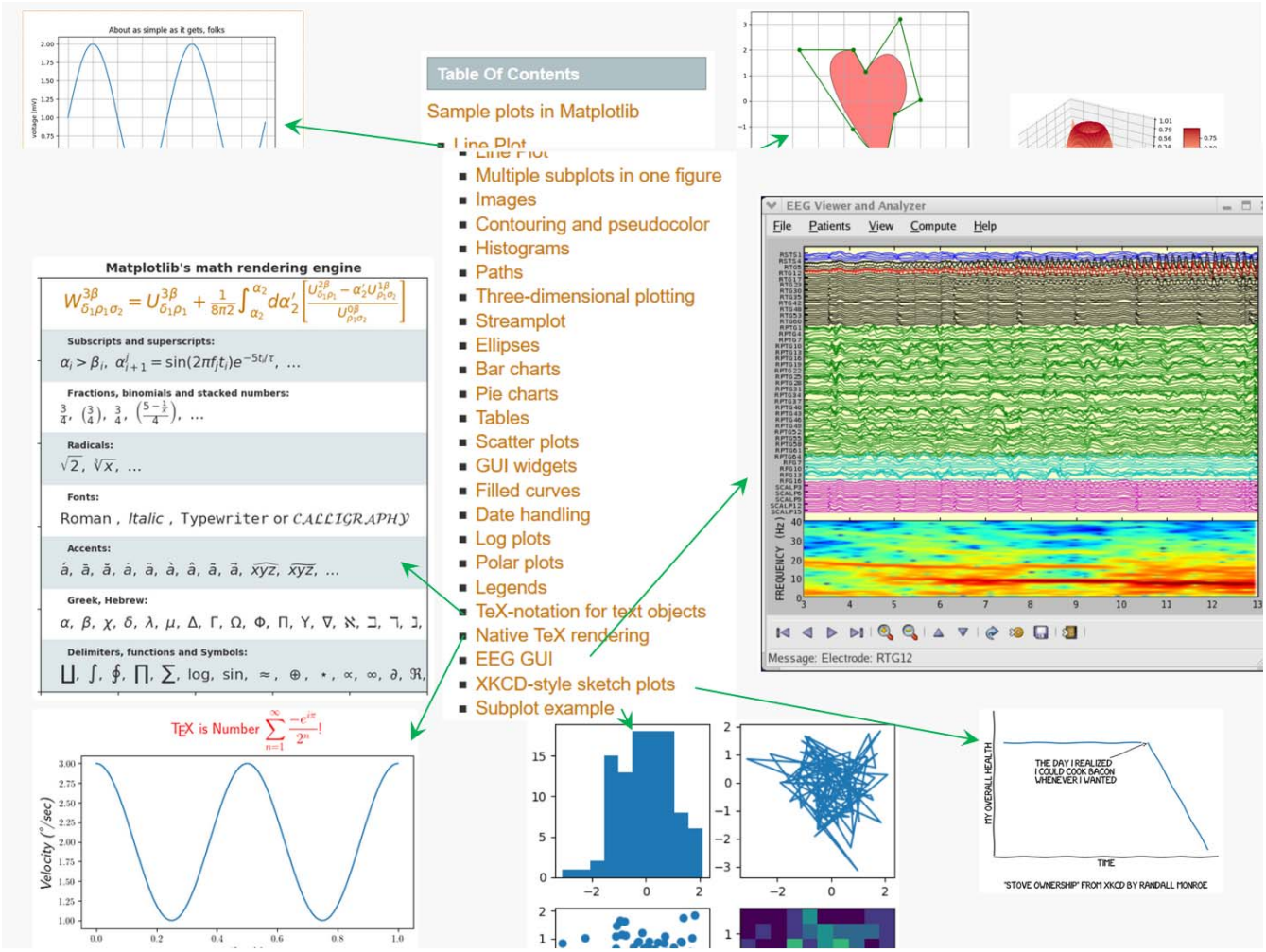


Table Of Contents
Sample plots in Matplotlib
Line Plot
Multiple subplots in one figure
Images
Contouring and pseudocolor
Histograms
Paths
Three-dimensional plotting
Streamplot
Ellipses
Bar charts
Pie charts
Tables
Scatter plots
GUI widgets
Filled curves
Date handling
Log plots
Polar plots
Legends
TeX-notation for text objects
Native TeX rendering
EEG GUI
XKCD-style sketch plots
Subplot example
About as simple as it gets, folks
Frogs
Logs
Hogs
Dogs
Loss by Disaster
Volume and percent change
semilogy
semilogx
loglog base 2 on x
Errorbars go negative
A line plot on a polar axis
Model length
Data length
Total message length



Brevísimo comentario final

Las dos referencias siguientes son obligadas, pero en Internet pueden encontrarse multitud de tutoriales y ejemplos hechos.

https://matplotlib.org/ (https://matplotlib.org/) --> home --> examples --> tutorials --> API --> docs
https://matplotlib.org/users/pyplot_tutorial.html (https://matplotlib.org/users/pyplot_tutorial.html)