

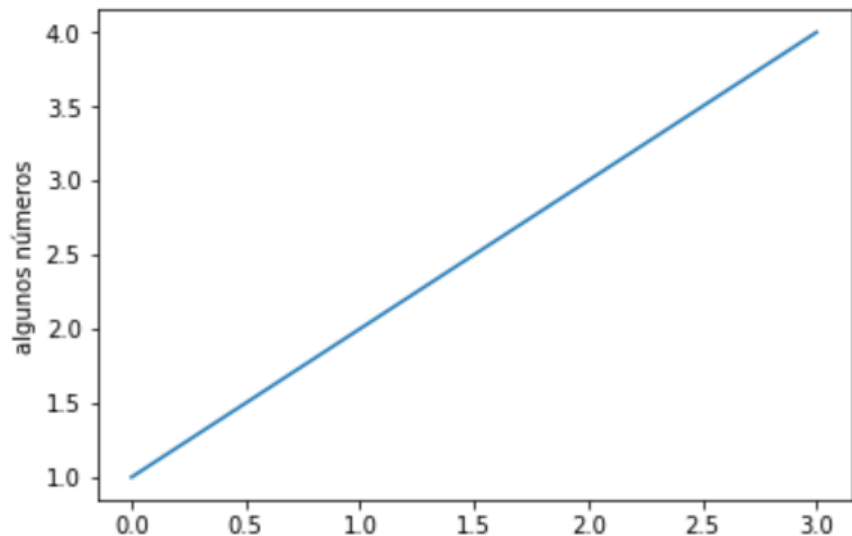


# Programación. Python

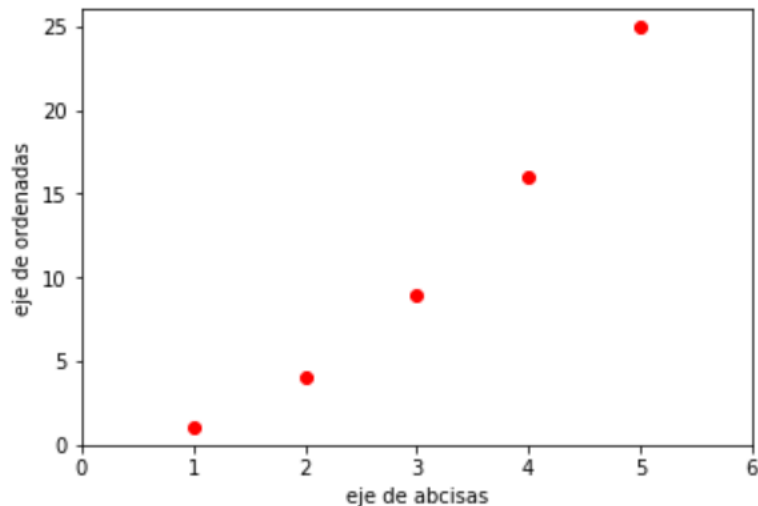
## Librería matplotlib

# matplotlib. Introducción

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

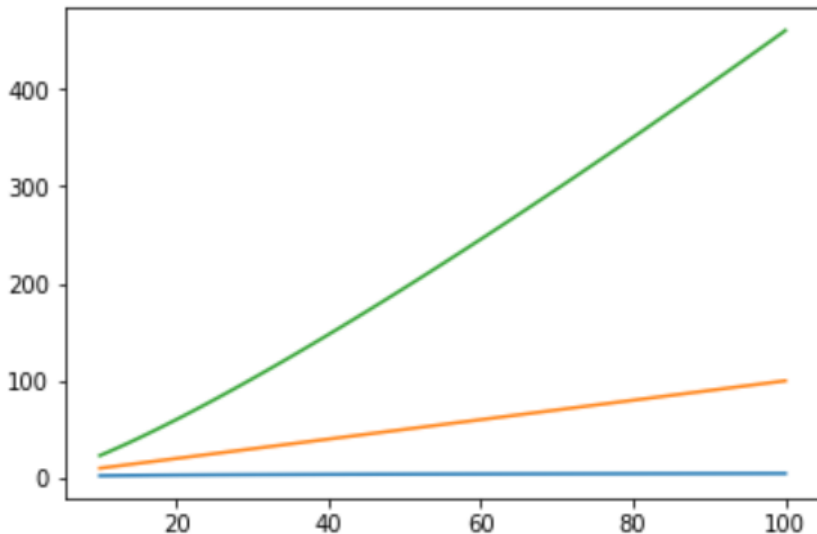


```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4, 5], [1, 4, 9, 16, 25], 'ro')
plt.axis([0, 6, 0, 26]) # x <- [0, 6], y <- [0, 26]
plt.xlabel('eje de abcisas')
plt.ylabel('eje de ordenadas')
plt.show()
```



# Varias gráficas juntas

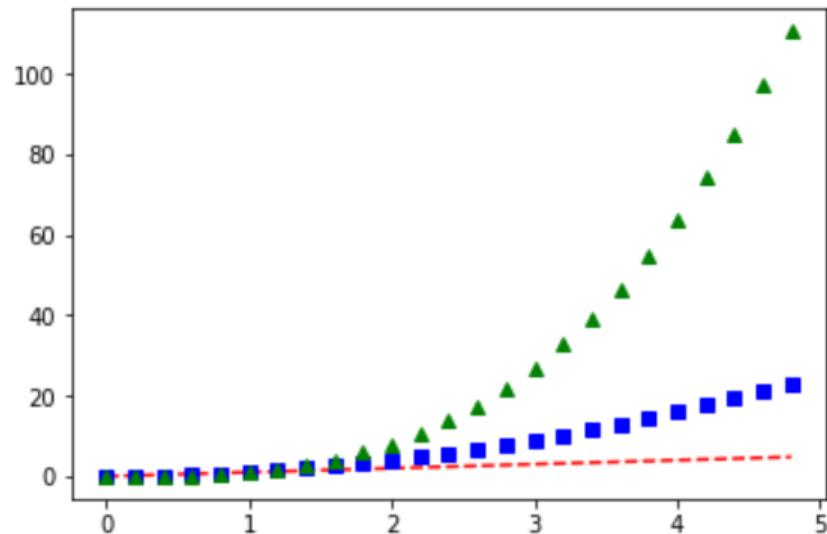
```
import numpy as np # para usar arrays.  
  
def nlogn(x):  
    return np.log(x) * x  
points = np.linspace(10,100,100)  
plt.plot(points, np.log(points), points, points, points, nlogn(points))  
plt.savefig('./figuras/plot.png', dpi=600) # hacer antes que show  
plt.show()
```



# Con formatos específicos

```
# 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()
```



# Encapsuado en una función

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

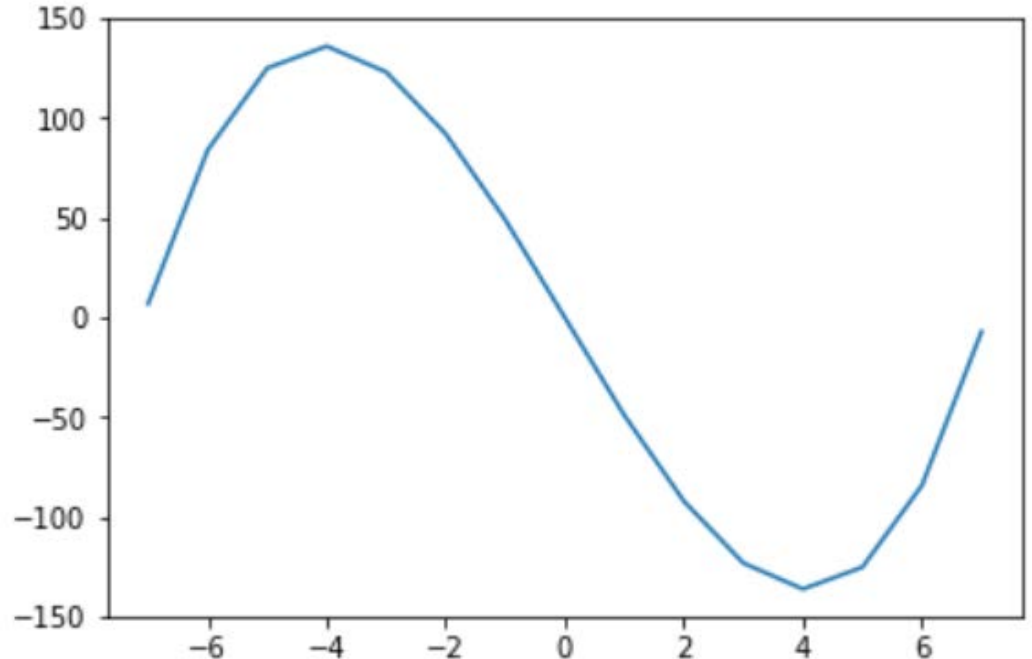
```
import numpy as np
import matplotlib.pyplot as plt

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()
```

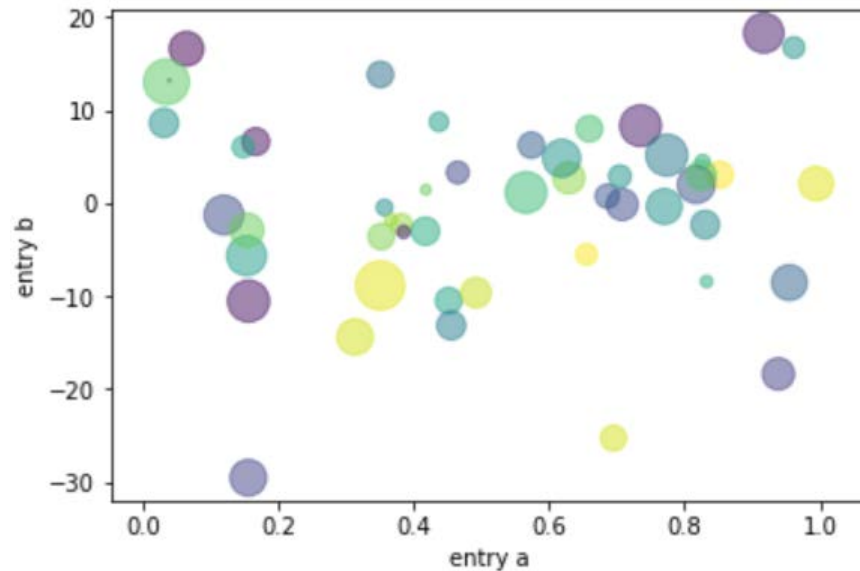


# Gráfico de dispersión

```
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()
```



```
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=None,
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``.

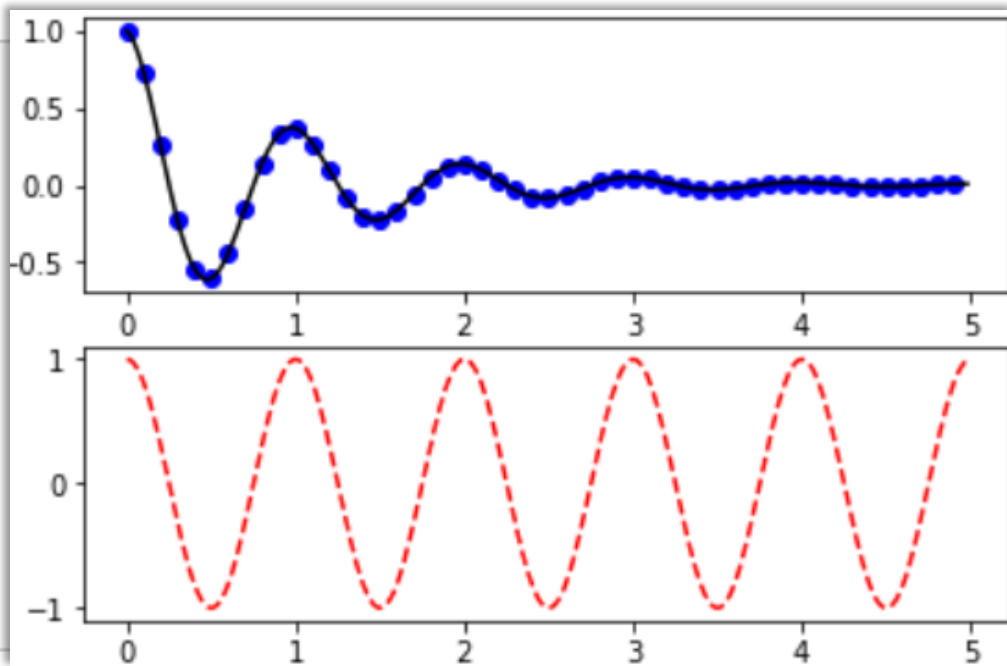
*c* : color, sequence, or sequence of color, optional

The marker color. Possible values:

- A single color format string.

# Varias gráficas separadas

```
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()
```





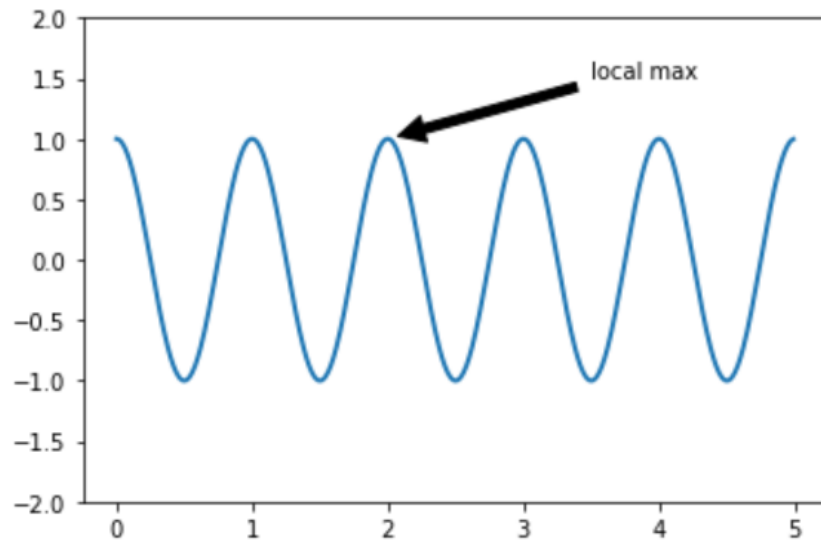
# Con anotaciones

```
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)

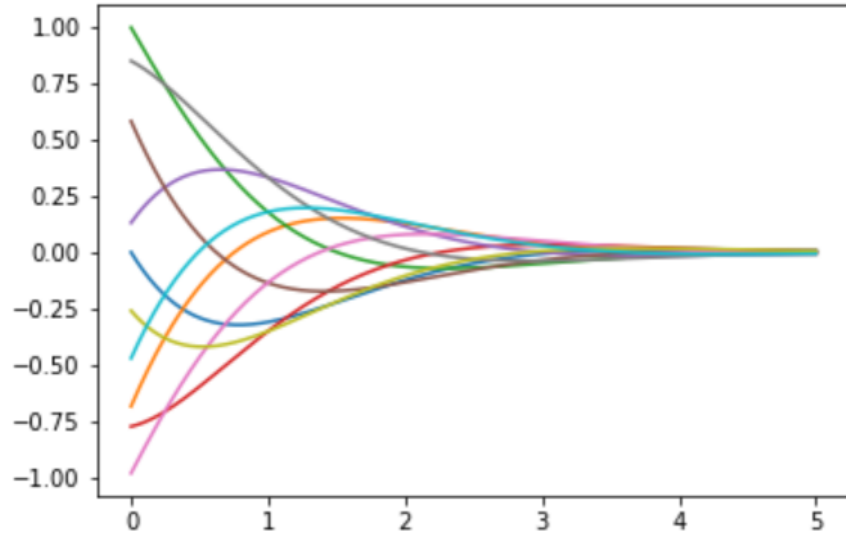
plt.annotate('local max', xy=(2, 1), # La punta,
             xytext=(3.5, 1.5), #Posición del texto,
             arrowprops=dict(facecolor='black', shrink=0.05),
             )

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



*# La segunda variable se está simulando a base de elegir una decena de valores*

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



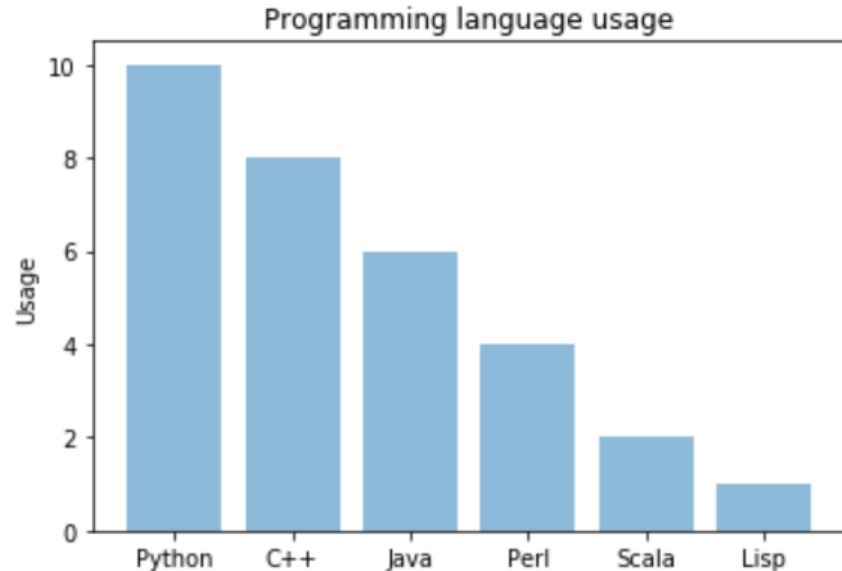
Dos  
variables

# Diagrama de barras

```
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()
```

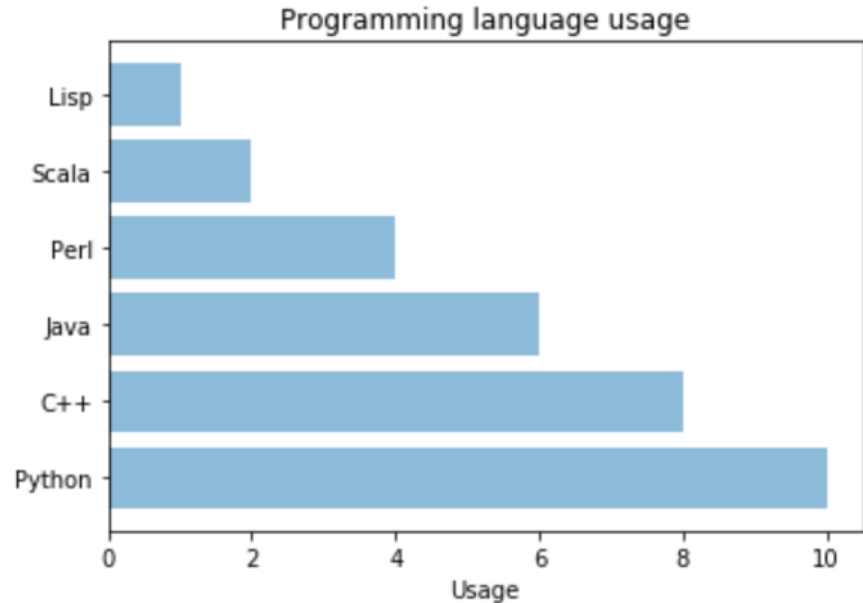


# Diagrama de barras

```
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()
```



# Diagrama de barras

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

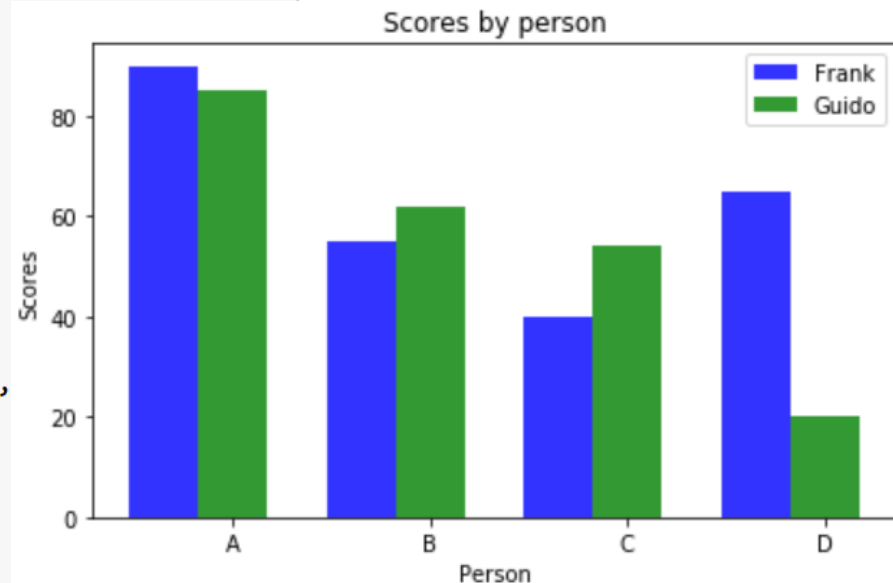
# create plot
fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8

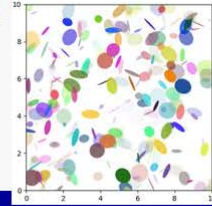
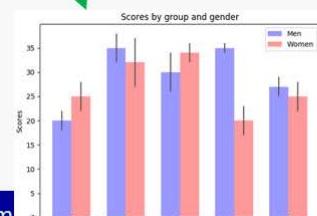
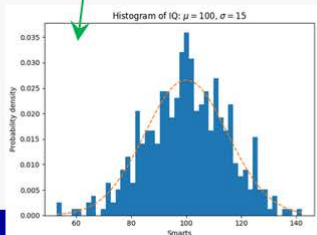
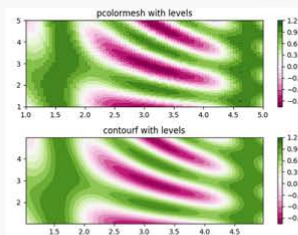
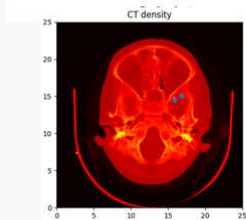
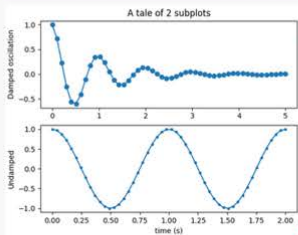
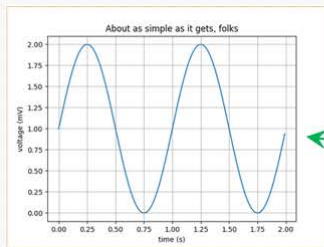
rects1 = plt.bar(index, means_frank, bar_width,
alpha=opacity,
color='b',
label='Frank')

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()
```

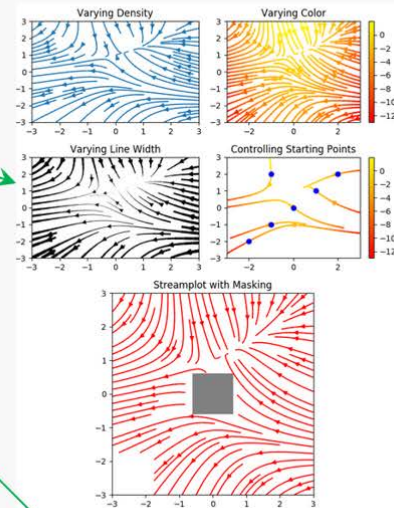
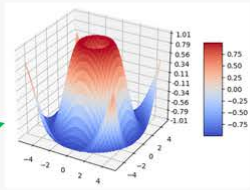




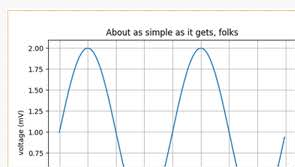
## Table Of Contents

### 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



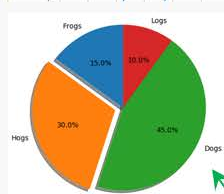
# matplotlib



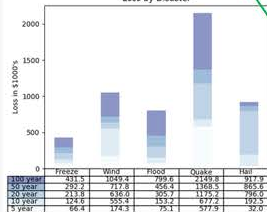
## Table Of Contents

### Sample plots in Matplotlib

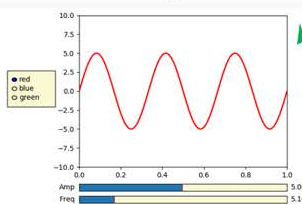
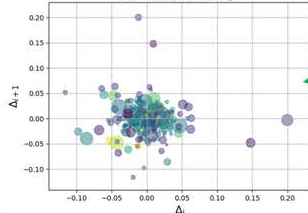
#### Line Plot



Loss by Disaster



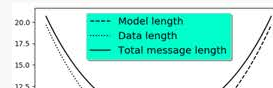
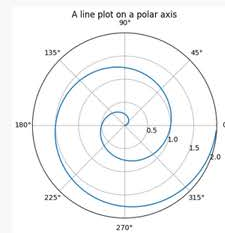
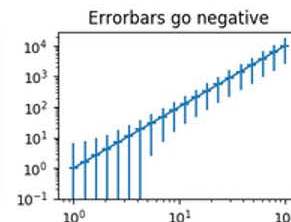
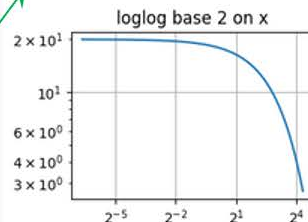
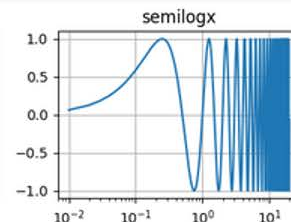
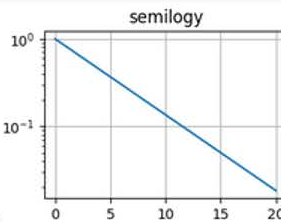
Volume and percent change



## Table Of Contents

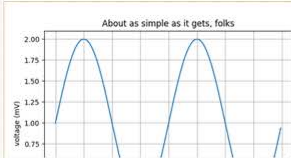
### Sample plots in Matplotlib

- 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





# matplotlib



## 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



#### Matplotlib's math rendering engine

$$W_{\partial_1 \rho_1 \partial_2}^{3\beta} = U_{\partial_1 \rho_1}^{3\beta} + \frac{1}{8\pi^2} \int_{\alpha_2}^{\alpha_2'} d\alpha_2' \left[ \frac{U_{\partial_1 \rho_1}^{2\beta} - \alpha_2' U_{\partial_1 \rho_1}^{1\beta}}{U_{\partial_1 \rho_1}^{0\beta}} \right]$$

#### Subscripts and superscripts:

$$\alpha_i > \beta_i, \alpha'_{i+1} = \sin(2\pi f_i t_i) e^{-5t_i/\tau}, \dots$$

#### Fractions, binomials and stacked numbers:

$$\frac{3}{4}, \binom{3}{4}, \frac{3}{4} + \left(\frac{5}{4} - \frac{1}{4}\right), \dots$$

#### Radicals:

$$\sqrt{2}, \sqrt[3]{x}, \dots$$

#### Fonts:

Roman, *Italic*, Typewriter or *CALLIGRAPHY*

#### Accents:

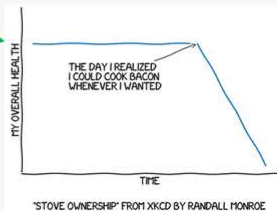
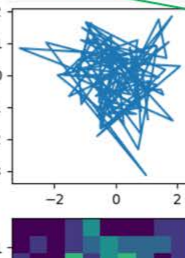
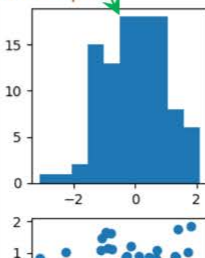
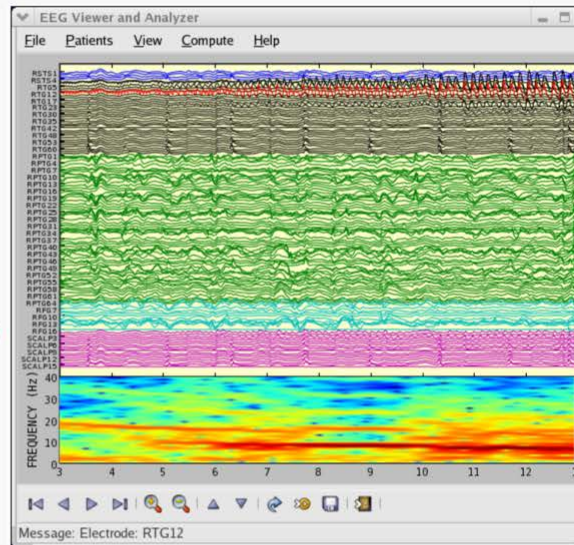
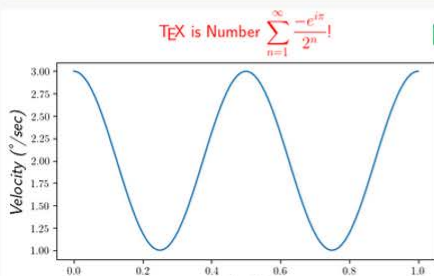
â, ã, ä, å, à, â, ã, ä, ä, xyz, xyz, ...

#### Greek, Hebrew:

$\alpha, \beta, \chi, \delta, \lambda, \mu, \Delta, \Gamma, \Omega, \Phi, \Pi, \Upsilon, \nabla, \aleph, \beth, \gamma, \daleth,$

#### Delimiters, functions and Symbols:

$\sqcup, \int, \oint, \prod, \sum, \log, \sin, \approx, \oplus, \ast, \propto, \infty, \partial, \Re,$





<https://matplotlib.org/>

→ home

→ examples

→ tutorials

→ API

→ docs

→ [https://matplotlib.org/users/pyplot\\_tutorial.html](https://matplotlib.org/users/pyplot_tutorial.html)