Taller de programación en Python para estadística descriptiva

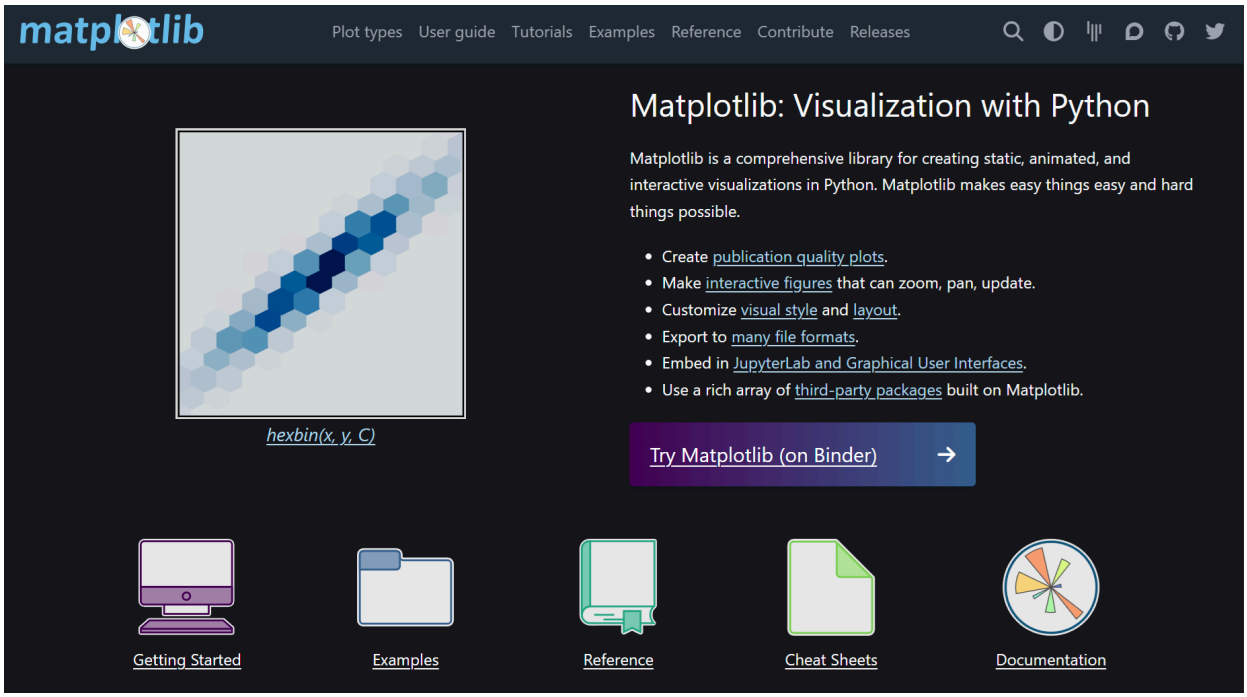
Elaborado por Joshua Martínez Domínguez

24/03/2025

Este documento contiene los temas desarrollados del *Taller de programación en Python para estadística descriptiva* con ejemplos en celdas de código. Este notebook corresponde a la sesion 5.

Matplotlib

Matplotlib es una libreria extensa para crear visualizaciones estadísticas, interactivas y animadas en Python



La instalación mas sencilla y facil de realizar es por medio de la distribución Anaconda, una distribución multiplataforma para analisis de datos y cómputo científico. El administrador del paquete Conda es el método de instalación recomendado para la mayoría de los usuarios.

También es posible instalarlo usando el comando

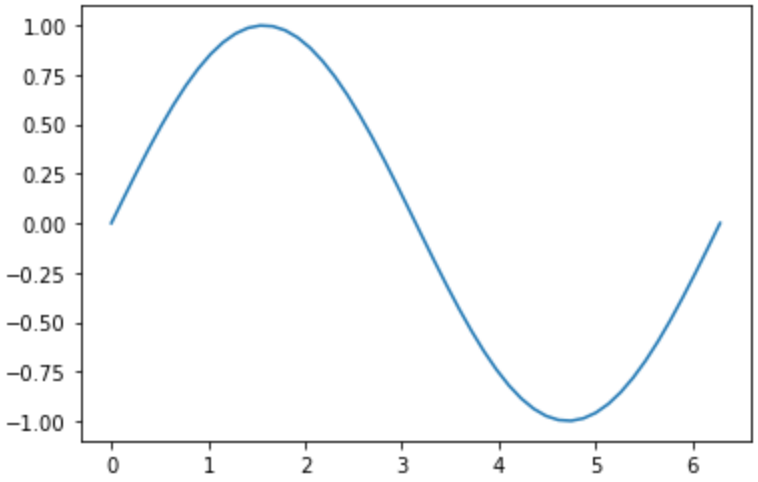
```
pip install matplotlib
```

Para comenzar a usarlo utilizamos el comando

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

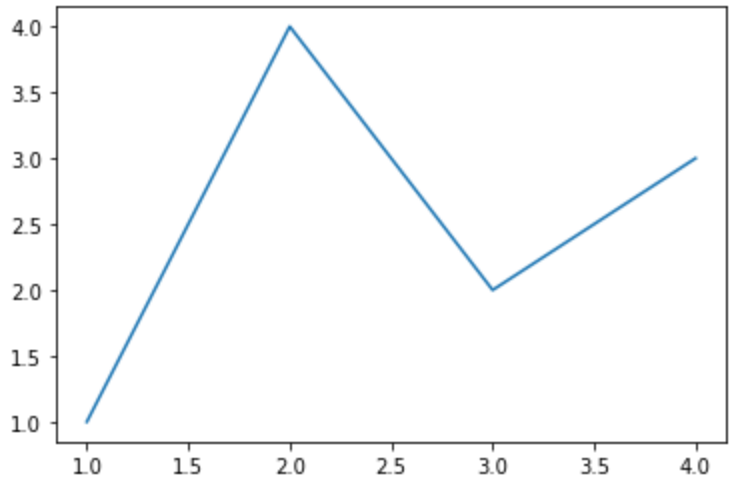
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [21]:
```
# ejemplo
x = np.linspace(0, 2 * np.pi, 50)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
print(x)
print(y)
```



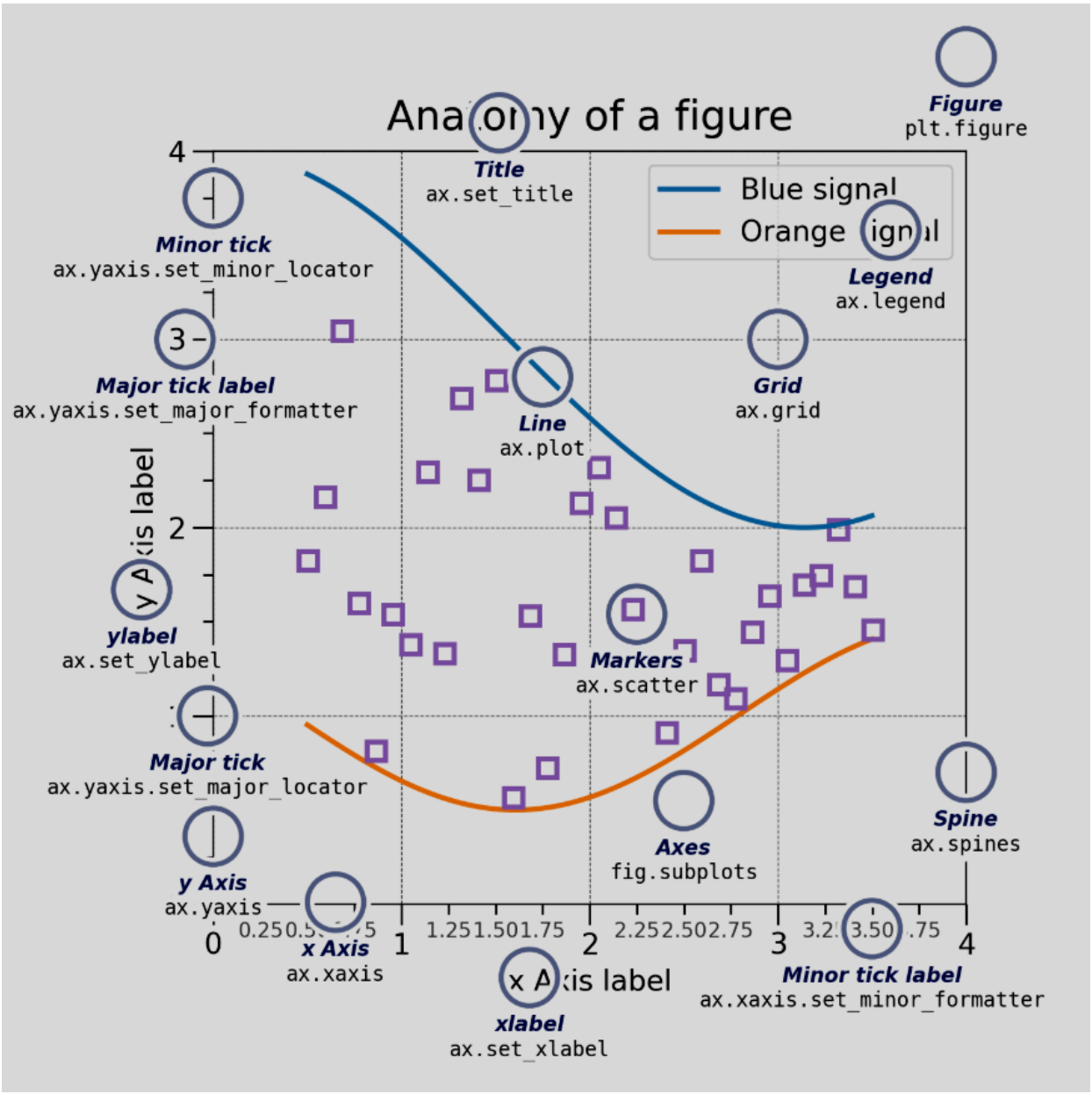

```
[0.         0.12822827 0.25645654 0.38468481 0.51291309 0.64114136
 0.76936963 0.8975979  1.02582617 1.15405444 1.28228272 1.41051099
 1.53873926 1.66696753 1.7951958  1.92342407 2.05165235 2.17988062
 2.30810889 2.43633716 2.56456543 2.6927937  2.82102197 2.94925025
 3.07747852 3.20570679 3.33393506 3.46216333 3.5903916  3.71861988
 3.84684815 3.97507642 4.10330469 4.23153296 4.35976123 4.48798951
 4.61621778 4.74444605 4.87267432 5.00090259 5.12913086 5.25735913
 5.38558741 5.51381568 5.64204395 5.77027222 5.89850049 6.02672876
 6.15495704 6.28318531]
[ 0.00000000e+00  1.27877162e-01  2.53654584e-01  3.75267005e-01
  4.90717552e-01  5.98110530e-01  6.95682551e-01  7.81831482e-01
  8.55142763e-01  9.14412623e-01  9.58667853e-01  9.87181783e-01
  9.99486216e-01  9.95379113e-01  9.74927912e-01  9.38468422e-01
  8.86599306e-01  8.20172255e-01  7.40277997e-01  6.48228395e-01
  5.45534901e-01  4.33883739e-01  3.15108218e-01  1.91158629e-01
  6.40702200e-02 -6.40702200e-02 -1.91158629e-01 -3.15108218e-01
 -4.33883739e-01 -5.45534901e-01 -6.48228395e-01 -7.40277997e-01
 -8.20172255e-01 -8.86599306e-01 -9.38468422e-01 -9.74927912e-01
 -9.95379113e-01 -9.99486216e-01 -9.87181783e-01 -9.58667853e-01
 -9.14412623e-01 -8.55142763e-01 -7.81831482e-01 -6.95682551e-01
 -5.98110530e-01 -4.90717552e-01 -3.75267005e-01 -2.53654584e-01
 -1.27877162e-01 -2.44929360e-16]
```

In [22]:
```
# Elementos de las instrucciones básicas

fig, ax = plt.subplots()              # Create a figure containing a single Axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the Axes.
plt.show()                            # Show the figure.
```



Partes de una figura

Se muestran las partes de una figura en matplotlib



La función:

```
matplotlib.pyplot.subplots(nrows=1, ncols=1, *, sharex=False,
sharey=False, squeeze=True, width_ratios=None, height_ratios=None,
subplot_kw=None, gridspec_kw=None, **fig_kw)[source]
```

crea una figura y un conjunto de subgráficas. Es conveniente cuando se usarán subgráficas con configuraciones comunes en una sola instrucción.

Los parámetros mas importantes de esta función son:

`nrows, ncols : int, default: 1` : Número de filas y columnas en el panel de subgráficas.

`sharex, shareybool or {'none', 'all', 'row', 'col'}, default: False` : Controla las propiedades que se comparten en los ejes x y y.

- *True* o "*all*": ejes x o y se compartirán en todos los subgfráficos.
- *False* o "*none*": ejes x o y serán independientes en cada subgráfica.
- 'row': cada fila de subgráficas compartirá un eje x o y
- 'col': cada columna de subgráficas compartirá un eje x o y

`squeezebool, default: True` : Si es *True*, la dimensiones son comprimidas para devolver un arreglo de `Axes`

- Si solo una subgráfica es construida (nrows = ncols = 1), el objeto singular `Axes` sera devuelto como un escalar.
- Para Nx1 or 1xM subgráficas, el objeto devuelto será un arreglo Numpy 1D en el objeto `Axes`
- Para NxM subgráficas, devolverá un arreglo 2D

-Si es *False*, el objeto `Axes` siempre devolverá un arreglo 2D

La función devuelve:

`fig` : Figura

`ax` : `Axes` o arreglo

`Figure`

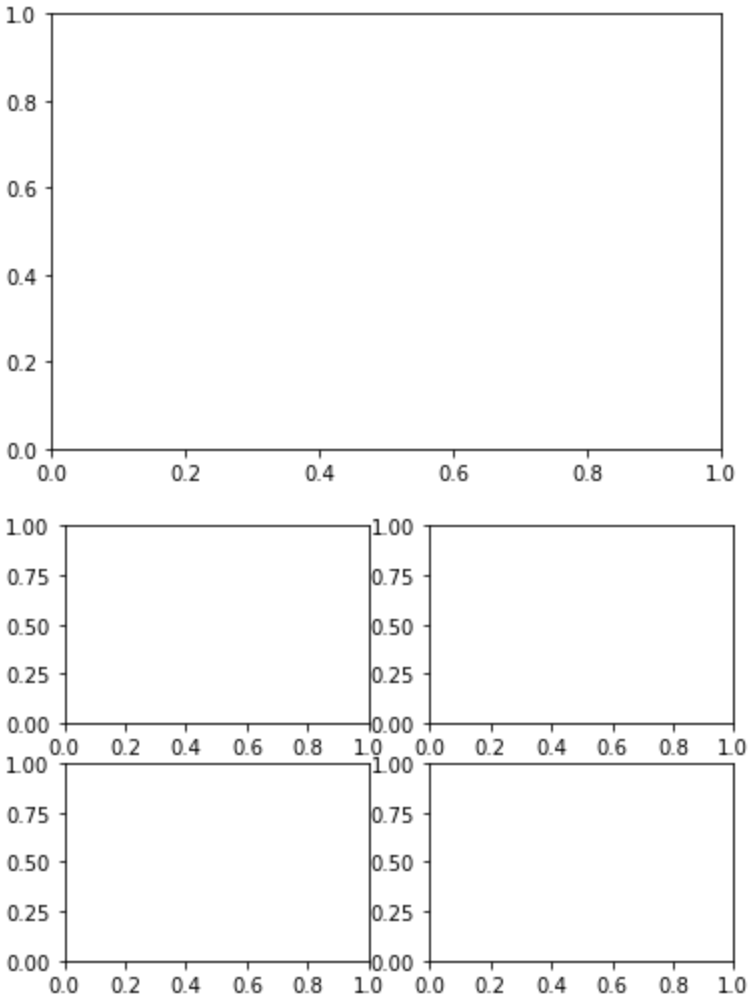
Es la figura completa. La figura almacena a los `Axes` , que son un grupo de arte especial (titulos, legendas de figuras, barras de colores, etc.)

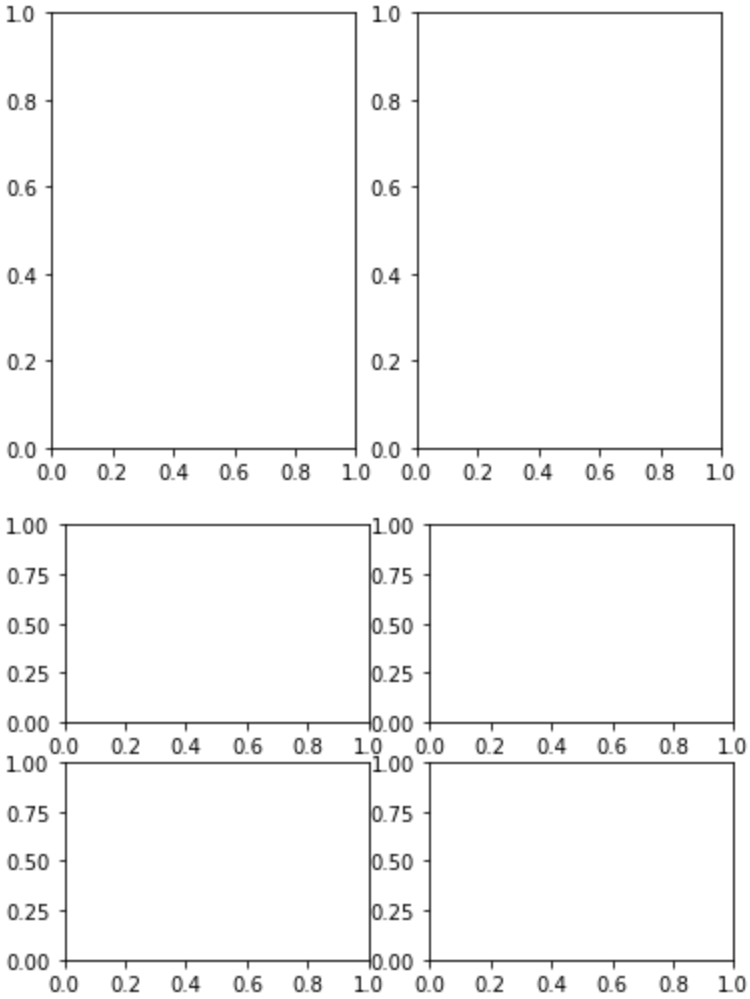
`Axes`

Son un arte anclado a la figura que contiene la región de datos graficados y usualmente incluye dos objetos `Axis` o ejes (hay que ser consciente de la diferencia de `Axes` y `Axis`) que brindan lineas

y marcas en las lineas para incluir escalas para los datos en los "Axes". Cada Axes contiene tambien un título, una etiqueta en x y una etiqueta en y.

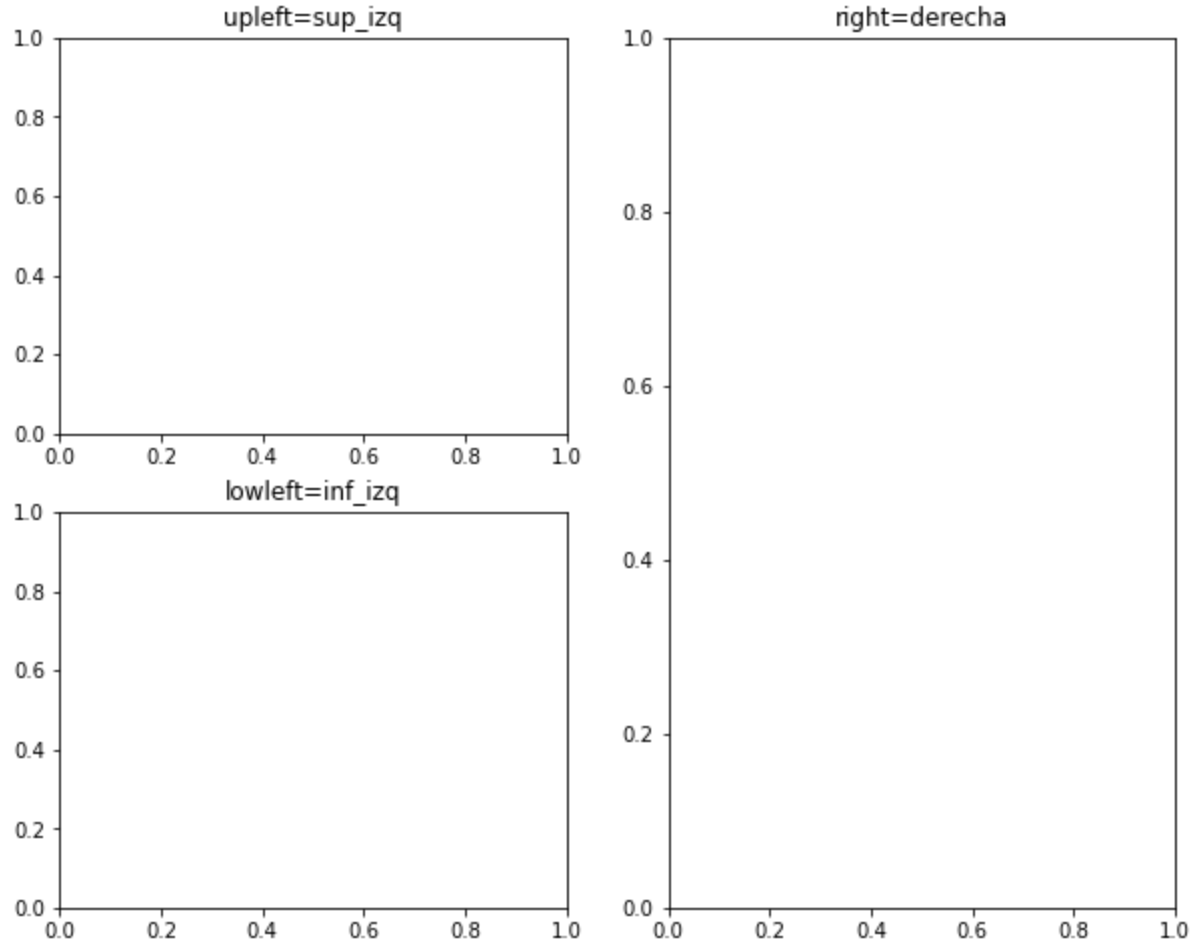
In [4]:
```
# using the variable ax for single a Axes
fig, ax = plt.subplots()

# using the variable axs for multiple Axes
fig, axs = plt.subplots(2, 2)

# using tuple unpacking for multiple Axes
fig, (ax1, ax2) = plt.subplots(1, 2)
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
```





In [34]:
```
fig, axd = plt.subplot_mosaic([['upleft', 'right'],
                               ['lowleft', 'right']], figsize=(10,8))
axd['upleft'].set_title('upleft=sup_izq')
axd['lowleft'].set_title('lowleft=inf_izq')
axd['right'].set_title('right=derecha')
```

Out[34]: Text(0.5, 1.0, 'right=derecha')



La función:

```
matplotlib.pyplot.show(*, block=None)
```

muestra todas las figuras abiertas.

Tambien es posible guardar las imágenes con la función

```
matplotlib.pyplot.savefig(*args, **kwargs)
```

Consideración: La función savefig debe ser ejecutada antes de la función `plt.show()` . De lo contrario, se creará una imagen transparente (sin el gráfico real).

Estilos de codificación

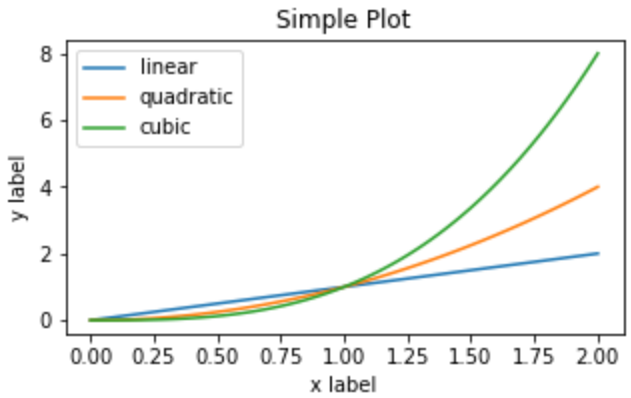
Hay dos formas esenciales de usar Matplotlib:

- Explícita: creando figuras y Axes, y llamar a los métodos sobre ellos (Estilo orientado a objetos)
- Implícita: Confiar en Pyplot para crear u gestionar figuras y axes implícitamente y usar punciones de graficado de Pyplot

In [24]:
```
# Estilo OO (Orientado a objetos)
```

```
x = np.linspace(0, 2, 100)  # Sample data.

# Note that even in the OO-style, we use `.pyplot.figure` to create the Figure.

fig, ax = plt.subplots(figsize=(5, 2.7))

ax.plot(x, x, label='linear')  # Plot some data on the Axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the Axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.

ax.set_xlabel('x label')  # Add an x-label to the Axes.
ax.set_ylabel('y label')  # Add a y-label to the Axes.
ax.set_title("Simple Plot")  # Add a title to the Axes.

ax.legend()  # Add a legend.
```

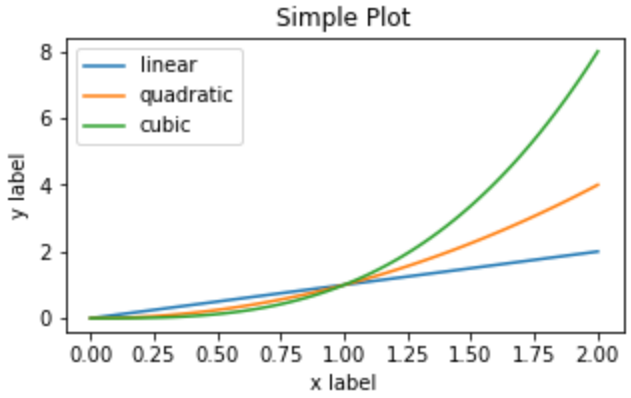
Out[24]: <matplotlib.legend.Legend at 0x1cdb40826d0>



In [26]:

```
# Estilo Pyplot

x = np.linspace(0, 2, 100)  # Sample data.

plt.figure(figsize=(5, 2.7))

plt.plot(x, x, label='linear')  # Plot some data on the (implicit) Axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")

plt.legend()
```

Out[26]: <matplotlib.legend.Legend at 0x1cdb4031040>



Tipos de `inputs` para funciones de graficación

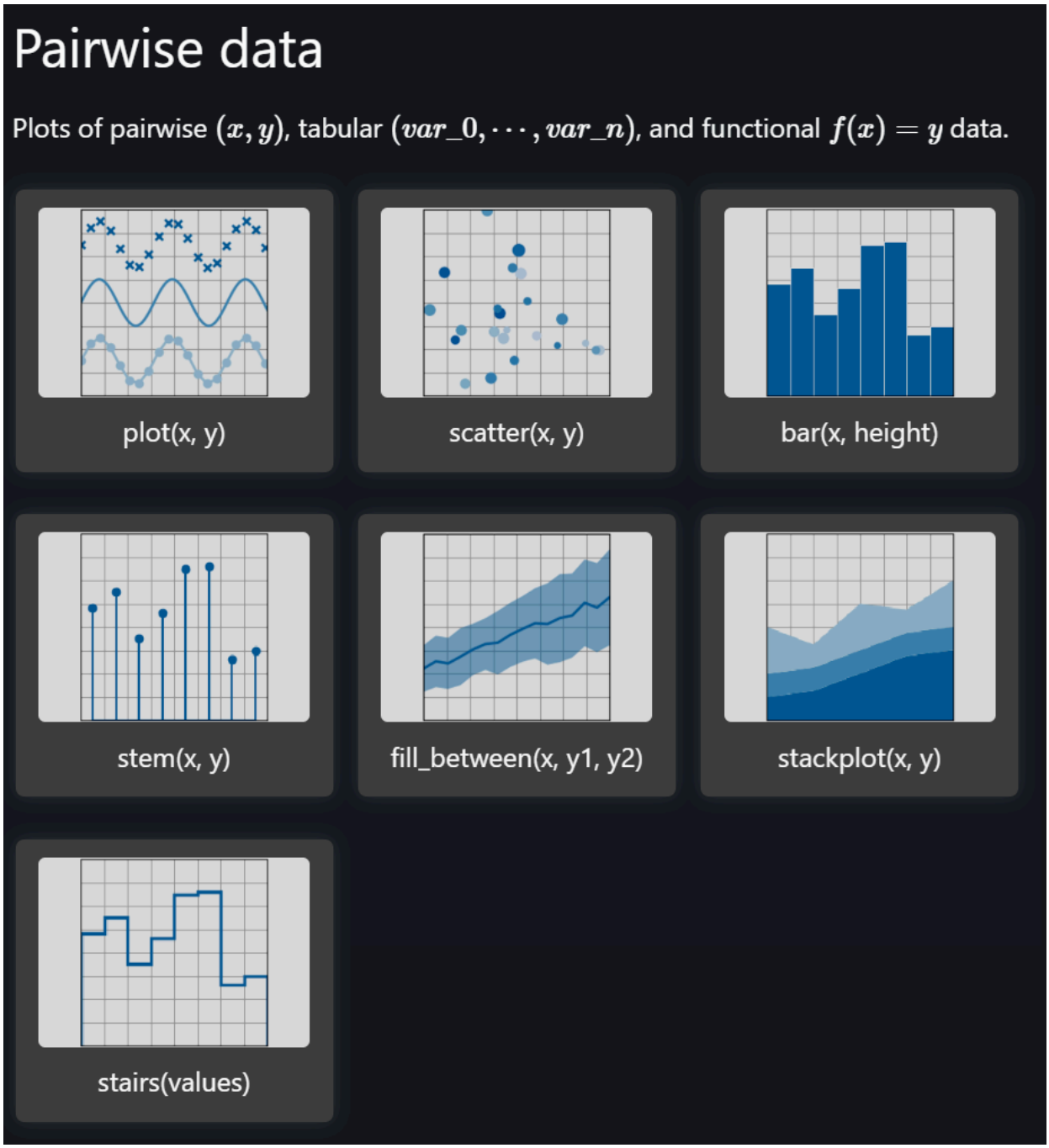
Las funciones para graficar esperan arreglos Numpy como input. Clases que son similares a los arreglos como objetos `pandas` o `numpy.matrix` podrian no trabajar como es la intención.

La mayoria de los métodos si analizan diccionarios (`dict`) o `pandas.DataFrame`

Tipos de gráficos

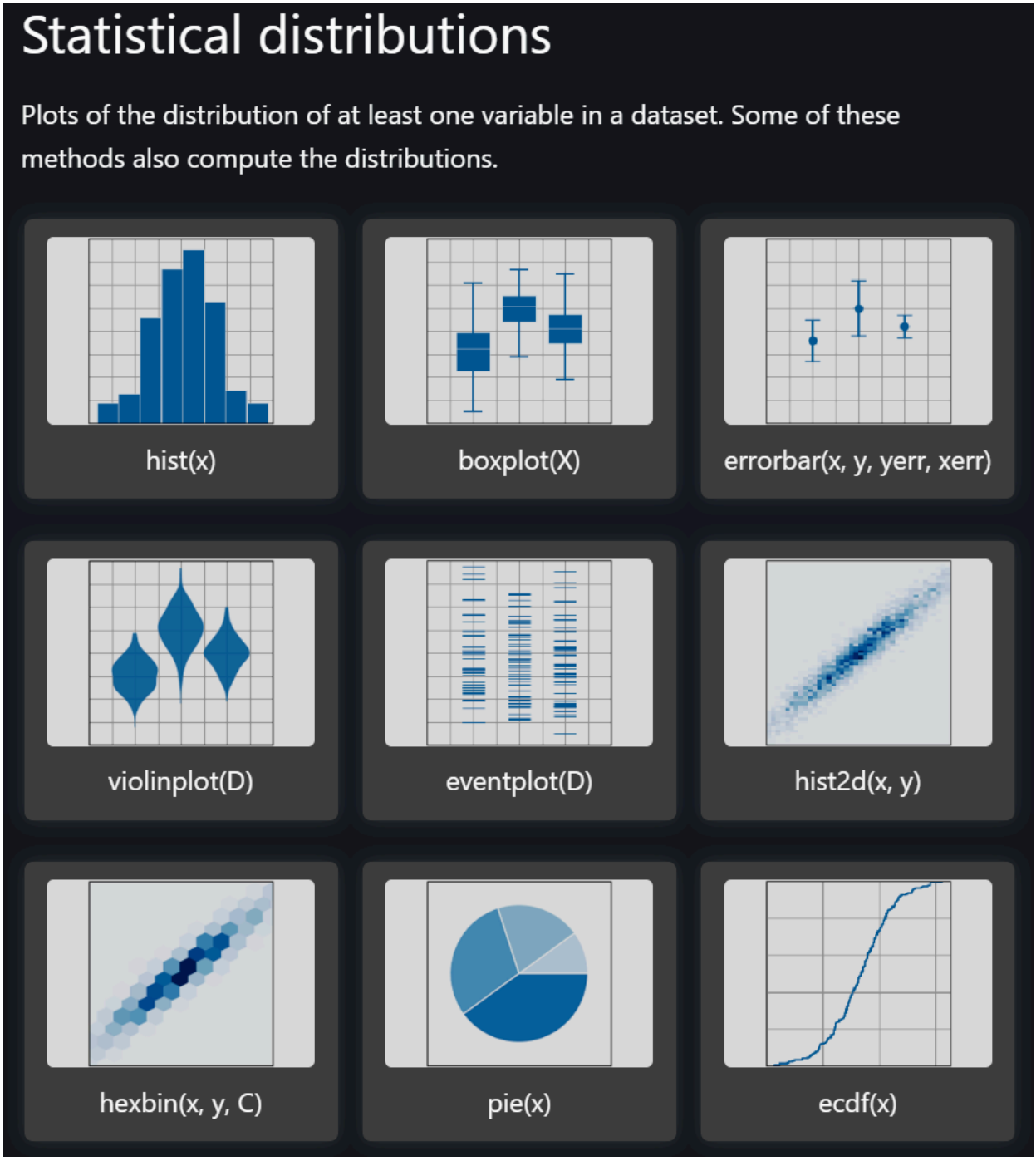
Datos pareados

Gráficos para datos pareados de la forma (x, y)), tabulares $(var_0, \ldots, var_n)$ y funcionales $f(x) = y$.



Distribuciones estadísticas

Gráficos de distribución de por lo menos una variable de un conjunto de datos. Algunos de estos métodos tambien computan las distribuciones de probabilidad.



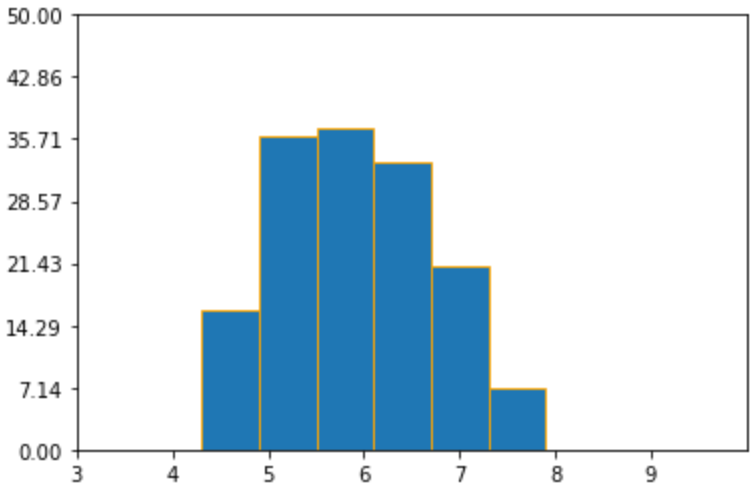
Histograma

La función se aplica sobre el objeto Axes:

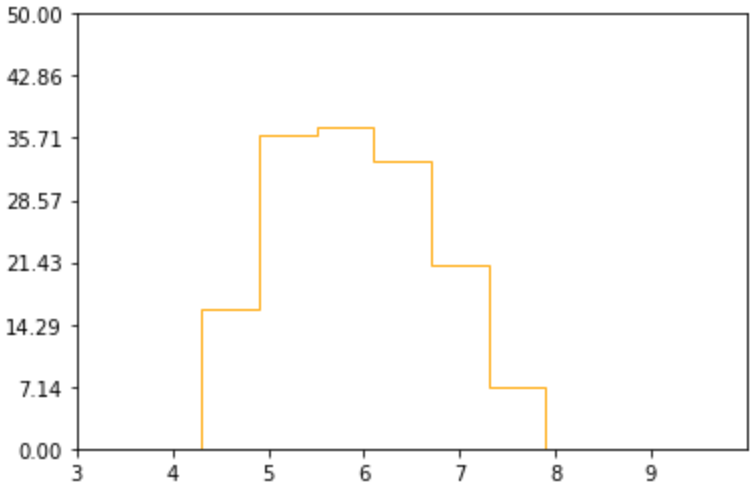
```
Axes.hist(x, bins=None, *, range=None, density=False, weights=None,
cumulative=False, bottom=None, histtype='bar', align='mid',
orientation='vertical', rwidth=None, log=False, color=None, label=None,
stacked=False, data=None, **kwargs)
```

Este método utiliza `numpy.histogram` para agrubar datos en categorias en x y contar el número de valores en cada categoría, entonces dibuja la distribución en barras o poligonos.

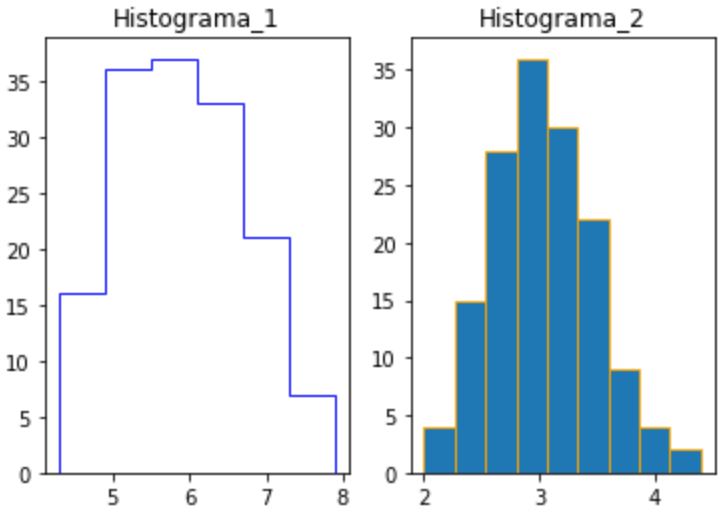
In [20]:
```
# Histograma

iris = pd.read_csv("iris.csv")

fig, ax = plt.subplots()

ax.hist(iris["Sepal.Length"], bins=6, linewidth=0.9, edgecolor="orange")

ax.set(xlim=(3, 10), xticks=np.arange(3, 10),
       ylim=(0, 50), yticks=np.linspace(0, 50, 8))

plt.show()
#plt.savefig("hist_prueba")
```



In [35]:
```
iris = pd.read_csv("iris.csv")

fig, ax = plt.subplots()

ax.hist(iris["Sepal.Length"], bins=6, histtype='step', linewidth=0.9, edgecolor="orange

ax.set(xlim=(3, 10), xticks=np.arange(3, 10),
       ylim=(0, 50), yticks=np.linspace(0, 50, 8))

plt.show()
```



In [50]:
```
iris = pd.read_csv("iris.csv")

fig, axs = plt.subplots(1, 2)

axs[0].hist(iris["Sepal.Length"], bins=6, histtype='step', linewidth=0.9, edgecolor="bl

axs[1].hist(iris["Sepal.Width"], bins=9, histtype='bar', linewidth=0.9, edgecolor="oran

axs[0].set_title("Histograma_1")
axs[1].set_title("Histograma_2")

plt.show()
```

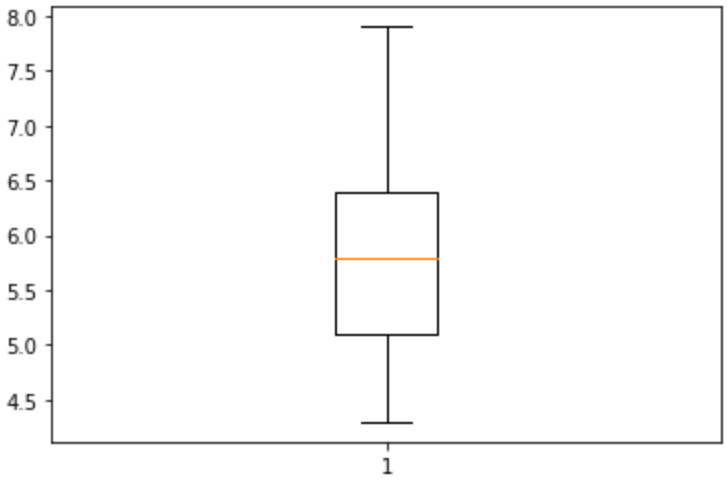


Boxplot

La función se aplica sobre el objeto Axes:

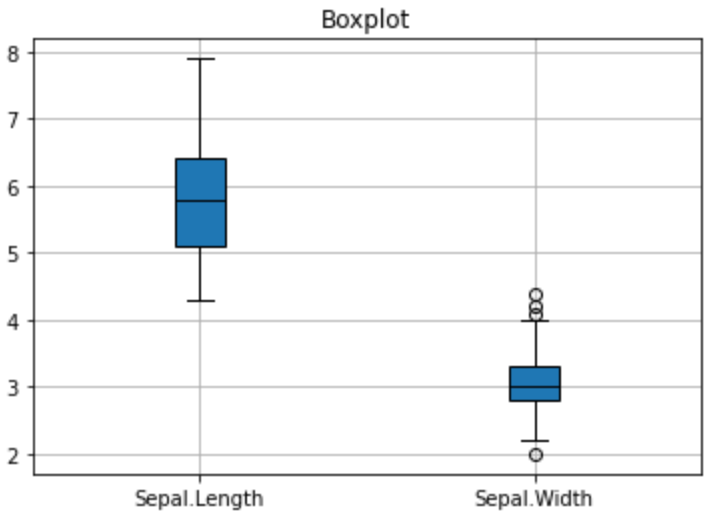
```
Axes.boxplot(x, *, notch=None, sym=None, vert=None,
orientation='vertical', whis=None, positions=None, widths=None,
patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=None,
meanline=None, showmeans=None, showcaps=None, showbox=None,
showfliers=None, boxprops=None, tick_labels=None, flierprops=None,
medianprops=None, meanprops=None, capprops=None, whiskerprops=None,
manage_ticks=True, autorange=False, zorder=None, capwidths=None,
label=None, data=None)
```

In [44]:
```
fig, ax = plt.subplots()

ax.boxplot(iris["Sepal.Length"])

plt.show()
```



In [51]:

```
fig, ax = plt.subplots()

ax.boxplot((iris["Sepal.Length"], iris["Sepal.Width"]),
           patch_artist=True, medianprops={"color": "black", "linewidth": 0.9},
           labels = ["Sepal.Length", "Sepal.Width"])

ax.grid(True)
ax.set_title("Histograma_2")

plt.show()
```



Violinplot

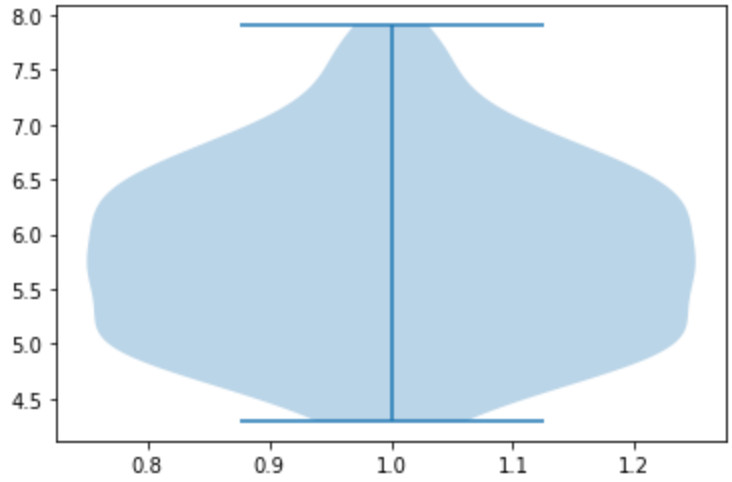
La función se aplica sobre el objeto Axes:

```
Axes.violinplot(dataset, positions=None, *, vert=None,
orientation='vertical', widths=0.5, showmeans=False, showextrema=True,
showmedians=False, quantiles=None, points=100, bw_method=None,
side='both', data=None)
```

In [52]:

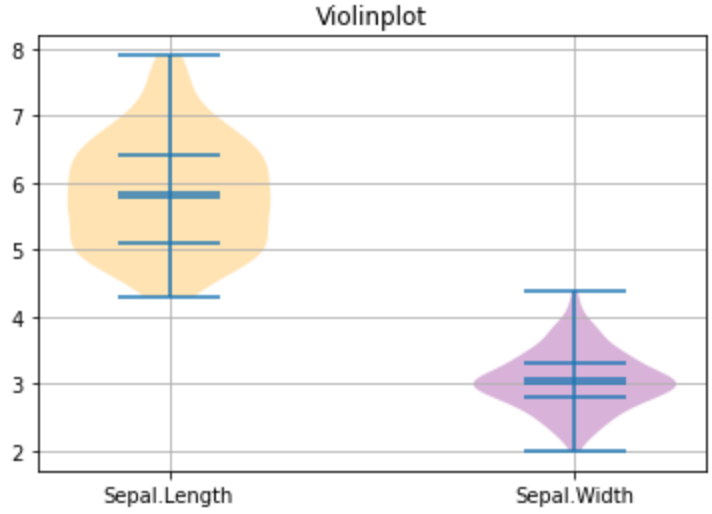
```
fig, ax = plt.subplots()

ax.violinplot(iris["Sepal.Length"])
```

```
plt.show()
```



In [73]:

```
fig, ax = plt.subplots()

violins = ax.violinplot((iris["Sepal.Length"], iris["Sepal.Width"]),
              showmeans=True, showmedians=True,
              quantiles = ([0.25, 0.75], [0.25, 0.75]))

colors = ['orange', 'purple']

for violin, color in zip(violins['bodies'], colors):
    violin.set_facecolor(color)

ax.grid(True)
ax.set_title("Violinplot")


ax.set(xticks = (1, 2), xticklabels = ("Sepal.Length", "Sepal.Width") )

plt.show()
```



Pie o pastel

La función se aplica sobre el objeto Axes:

```
Axes.pie(x, *, explode=None, labels=None, colors=None, autopct=None,
pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0, radius=1,
counterclock=True, wedgeprops=None, textprops=None, center=(0, 0),
frame=False, rotatelabels=False, normalize=True, hatch=None, data=None)
```

In [90]:
```
a = iris["Species"].value_counts()
print(a)
type(a)
```

```
versicolor    50
setosa        50
virginica     50
Name: Species, dtype: int64
```

Out[90]: pandas.core.series.Series

In [92]:
```
fig, ax = plt.subplots()

x = [50, 50, 50]

ax.pie(x)

plt.show()
```

In [99]:
```
fig, ax = plt.subplots()

x = [50, 50, 50]
especies = ["versicolor", "setosa", "virginica"]

colores = ["#EE6055","#60D394","#AAF683","#FFD97D","#FF9B85"]

desfase = (0, 0, 0.2)

ax.pie(x, labels = especies, autopct="%0.1f %%", colors=colores, explode=desfase)

ax.set_title("Gráfico de pastel")

plt.show()
```

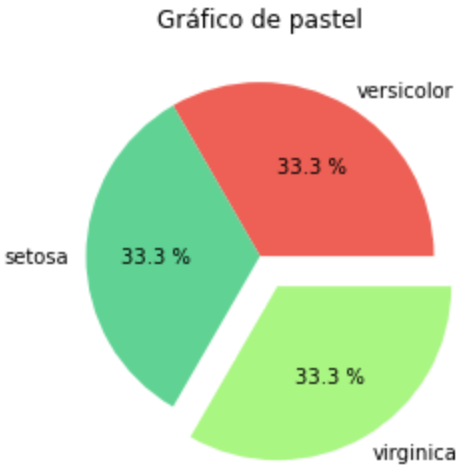


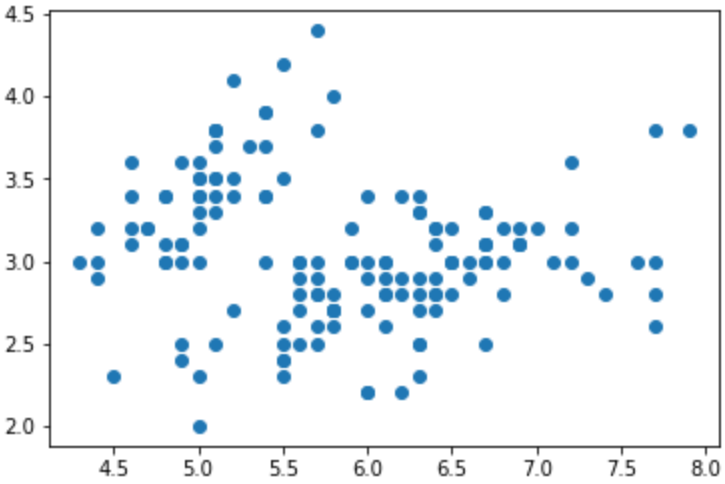
Gráfico de dispersión

La función se aplica sobre el objeto Axes:

```
Axes.scatter(x, y, s=None, c=None, *, marker=None, cmap=None, norm=None,
vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None,
colorizer=None, plotnonfinite=False, data=None, **kwargs)
```

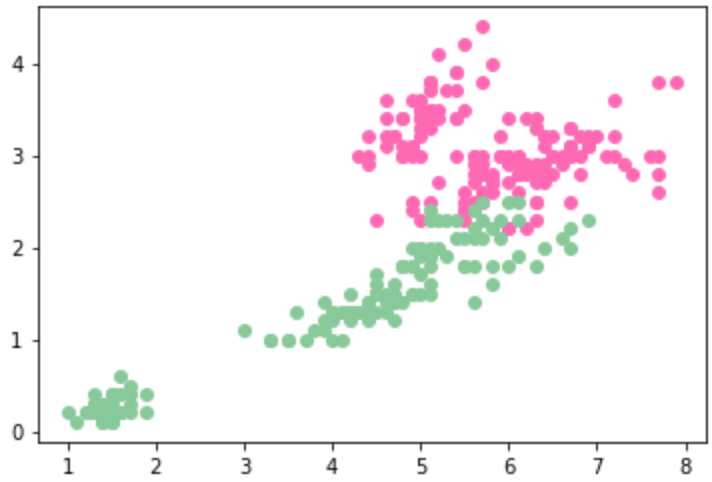
In [104…

```
fig, ax = plt.subplots()

ax.scatter(iris["Sepal.Length"], iris["Sepal.Width"])

plt.show()
```



In [106…

```
fig, ax = plt.subplots()

ax.scatter(iris["Sepal.Length"], iris["Sepal.Width"], color = 'hotpink')
ax.scatter(iris["Petal.Length"], iris["Petal.Width"], color = '#88c999')

plt.show()
```



In [116…

```
fig, ax = plt.subplots()

colors = np.array(len(iris["Sepal.Length"]))

sizes = np.array(len(iris["Sepal.Length"]))

ax.scatter(iris["Sepal.Length"], iris["Sepal.Width"], cmap = "Accent", s = sizes, alpha
ax.scatter(iris["Petal.Length"], iris["Petal.Width"], color = '#88c999')

ax.set_xlabel("Largo")
ax.set_ylabel("Ancho")
ax.set_title("Scatter plot")

ax.grid(True)

plt.show()
```

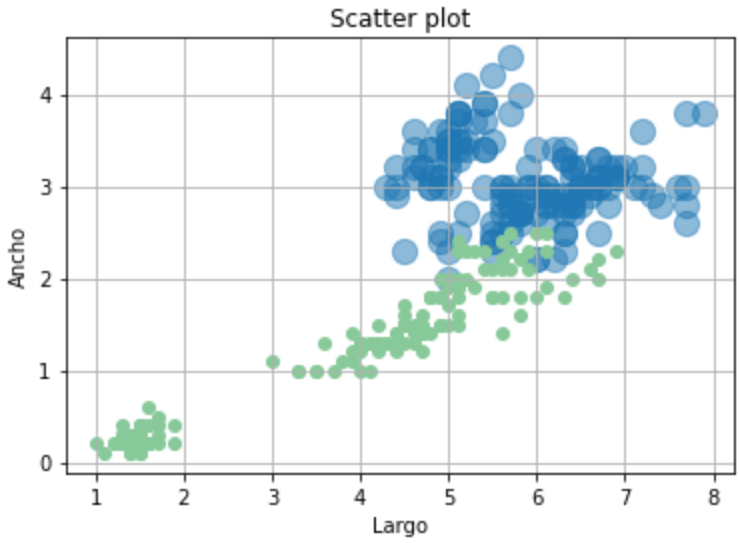


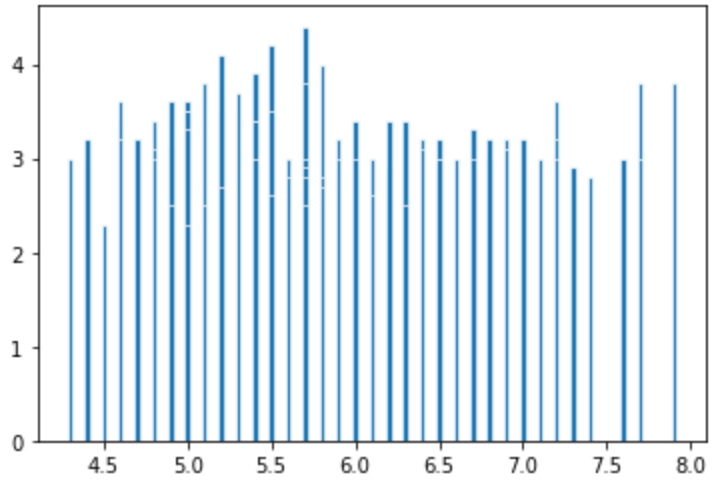
Gráfico de barras

La función se aplica sobre el objeto Axes:

```
Axes.bar(x, height, width=0.8, bottom=None, *, align='center', data=None,
**kwargs)
```

In [123…

```
fig, ax = plt.subplots()

ax.bar(iris["Sepal.Length"], iris["Sepal.Width"], width=.03, edgecolor="white", linewid

plt.show()
```



Ejercicio 5

Realice dos gráficos apropiados y personalizados de un conjunto de datos real que haya creado, o bien, cree uno ficticio con las funciones de Python

Referencias

https://matplotlib.org/stable/users/explain/quick_start.html#parts-of-a-figure

https://matplotlib.org/stable/plot_types/index.html

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.hist.html#matplotlib.axes.Axes.hist

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.boxplot.html#matplotlib.axes.Axes.boxplo

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.violinplot.html#matplotlib.axes.Axes.violi

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.pie.html#matplotlib.axes.Axes.pie

In [ ]: