MATPLOTLIB

Es una librería para crear visualizaciones estáticas, animadas e interactivas. Las raíces de esta librería están en el programa Matlab, aunque en su implementación nos podemos apoyar de otras librerías, como lo es numpy.

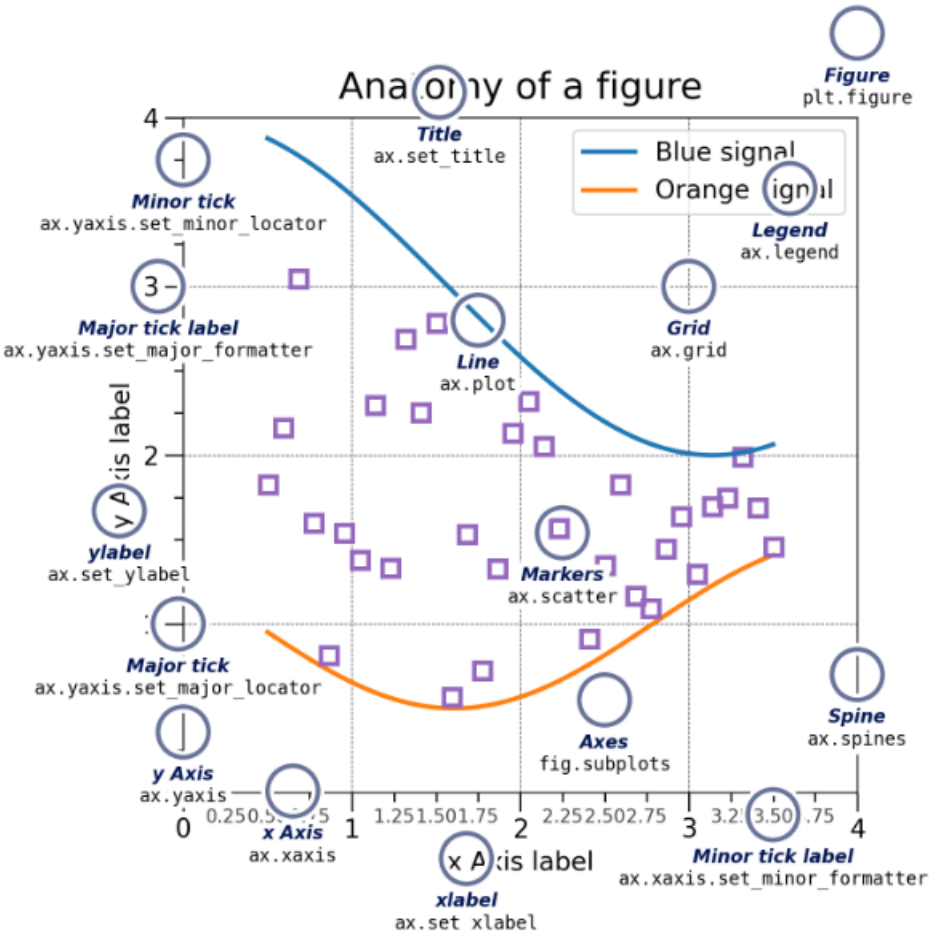
El primer paso es descargar las siguientes librerías:

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

Matpltlib puede graficar datos con diferentes formatos, por ejemplo, como dos dimensiones, tres dimensiones, formato polar.

La forma más simple de crear una figura con ejes es usando **pyplot.subplots**, y para figurar datos podemos apoyarnos en el uso de **Axes.plot** .

Las componentes de una figura son las siguientes:



Para construir una figura con ejes, usando pyplot, se pueden seguir los siguientes pasos:

fig = plt.figure() # crea una figura vacia, sin ejes.

fig, ax = plt.subplots() # una figura con un solo eje.

fig, axs = plt.subplots(2, 2) # una figura con una cuadricula 2x2

Axes.

Un axes es un adjunto de artista para graficar, que contiene una región para dibujar datos, usualmente incluye dos o tres objetos axis, y estos a su vez, proporcionan marcas y etiquetas para proporcionar escalas para los datos. Además, cada axes también posee:

Un título, con el comando **set_title()**

Un x-label, con el comando **set_xlabel()**

Y el y-label, con el comando **set_ylabel()**

Axis.

Estos objetos marcan la escala y límites y generan ticks (que son las marcas en los ejes), y ticklables (cadenas que etiquetan los ticks). La localización de los ticks está determinado por un objeto **Locator** y las cadenas ticklabels son formateadas por un **Formatter**. Esto brinda la localización de tick y labels.

Artist.

Todos los elementos visibles en la figura es un Artist (incluso los objetos de figura, Axes, y Axis). Estos incluyen, objetos de: text, Line2D, collections, Path, entre otros. Cuando la figura es representada, todos elementos de Arttsts son dibujados en canvas.

Tipos de inputs para dibujar funciones.

Las funciones para dibujar esperan tener como input numpy.array o numpy.ma.masked_array u objetos que pueden ser pasados numpy.asarray. Clases como similares array, tales como panda y numpy.matrix no funcionan como lo visto, usualmente es convertido a objetos numpy.array.

Ejemplo: Queremos convertir el objeto numpy.matrix [[1, 2], [3, 4]]. Hacemos lo siguiente:

```
b = np.matrix([[1, 2], [3, 4]])
b_asarray = np.asarray(b)
```

Existen dos diferentes estilos de código para usar Matplotlib:

- Explícitamente creando figures y axes, usando los métodos de ellos.
- Confiar en pyplot para que automáticamente administre las figuras y axes, y use funciones pyplot para dibujar.

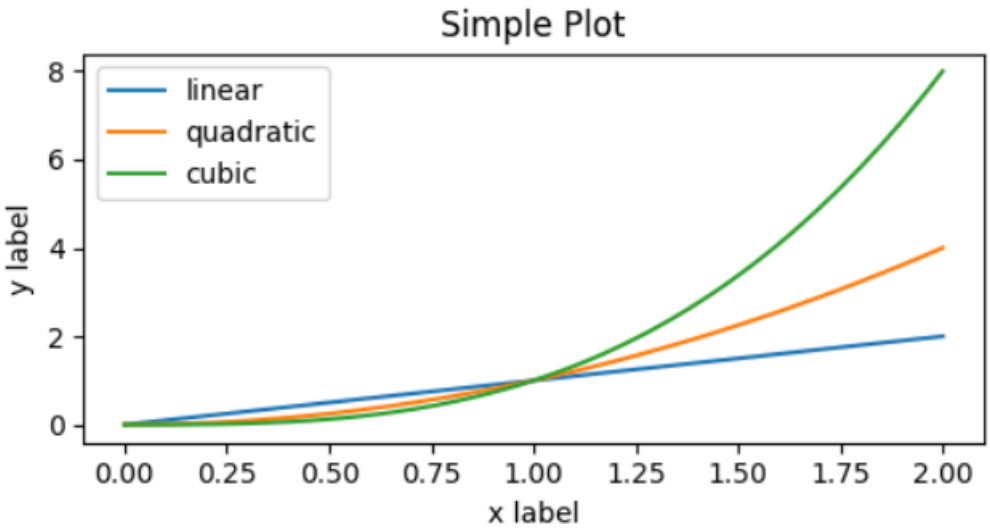
Ejemplos:

Usando el primer estilo:

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

# Note que aún cuando usamos este estilo, nosotros usamos `.pyplot.figure` para crear la figura.
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.plot(x, x, label='linear')
ax.plot(x, x**2, label='quadratic')
ax.plot(x, x**3, label='cubic')
ax.set_xlabel('x label')  # Adiciona un x-label.
ax.set_ylabel('y label')  # Adiciona un y-label.
ax.set_title("Simple Plot")  # Adiciona titulo.
ax.legend();  # Adiciona un legenda.
```

O el otro estilo:

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

plt.figure(figsize=(5, 2.7), layout='constrained')
plt.plot(x, x, label='linear')  # Dibuja alguna data en los axes implicitos.
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend();
```

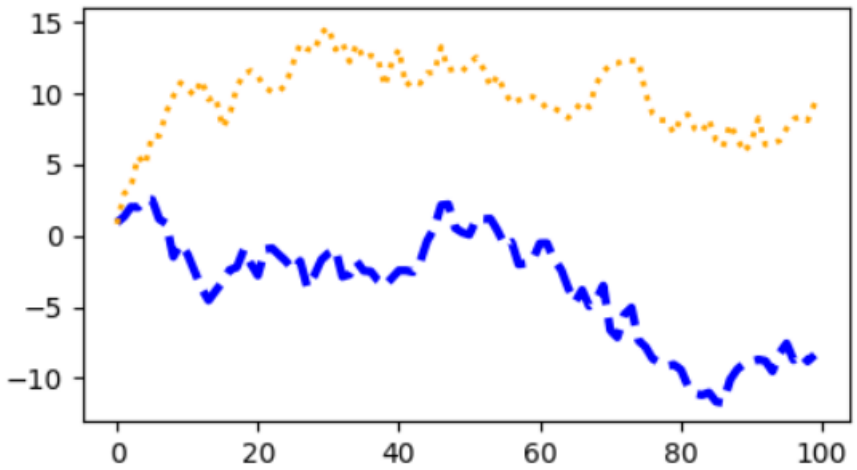


Estilos para el Artists

Ejemplo:

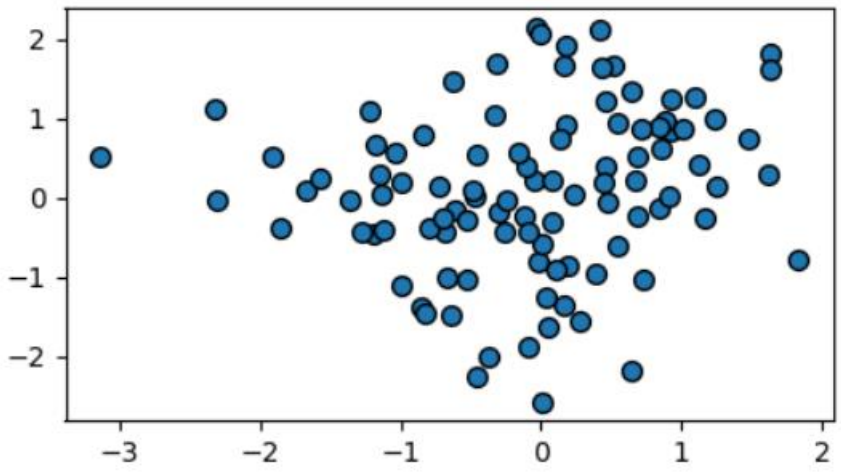
Tipo de línea1.

```
fig, ax = plt.subplots(figsize=(5, 2.7))
x = np.arange(len(data1))
ax.plot(x, np.cumsum(data1), color='blue', linewidth=3, linestyle='--')
l, = ax.plot(x, np.cumsum(data2), color='orange', linewidth=2)
l.set_linestyle(':');
```



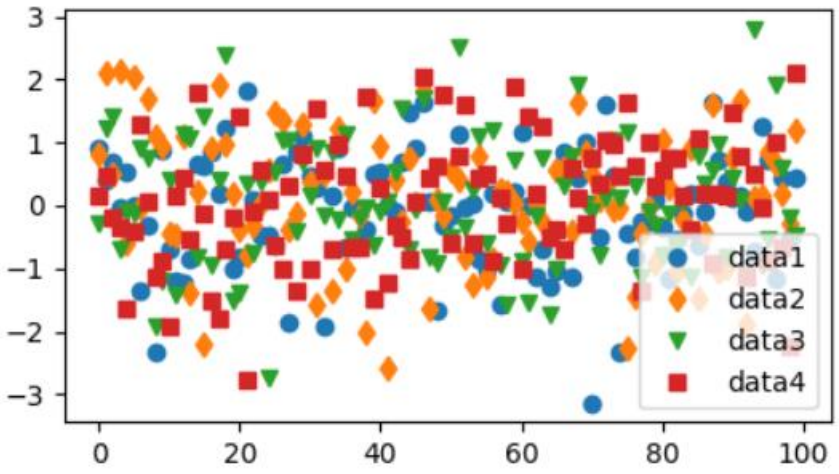
Colores:

```
fig, ax = plt.subplots(figsize=(5, 2.7))
ax.scatter(data1, data2, s=50, facecolor='C0', edgecolor='k');
```



Marcas:

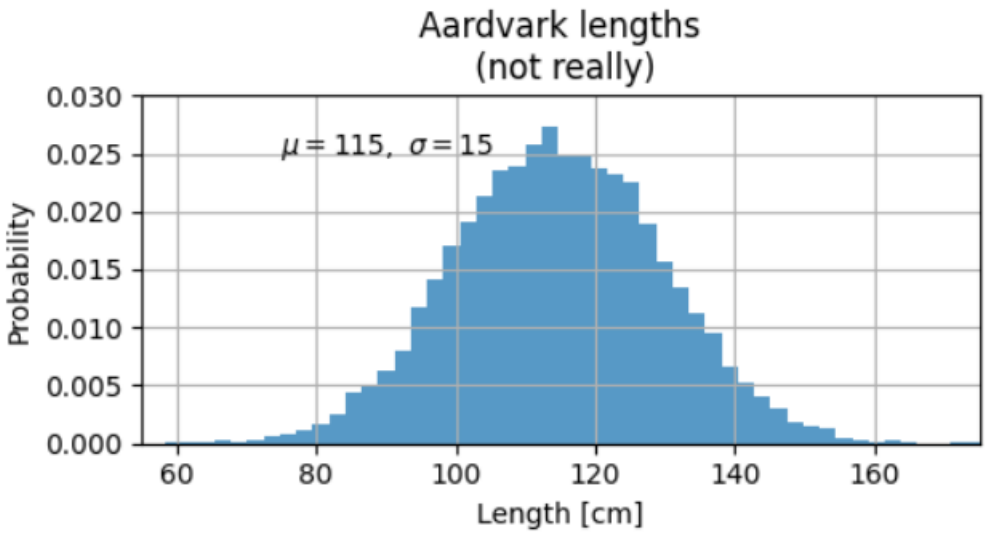
```
fig, ax = plt.subplots(figsize=(5, 2.7))
ax.plot(data1, 'o', label='data1')
ax.plot(data2, 'd', label='data2')
ax.plot(data3, 'v', label='data3')
ax.plot(data4, 's', label='data4')
ax.legend();
```



Etiquetas y texto:

set_xlabel, set_ylabel, and **set_title** son usados para añadir texto en los espacios indicados.

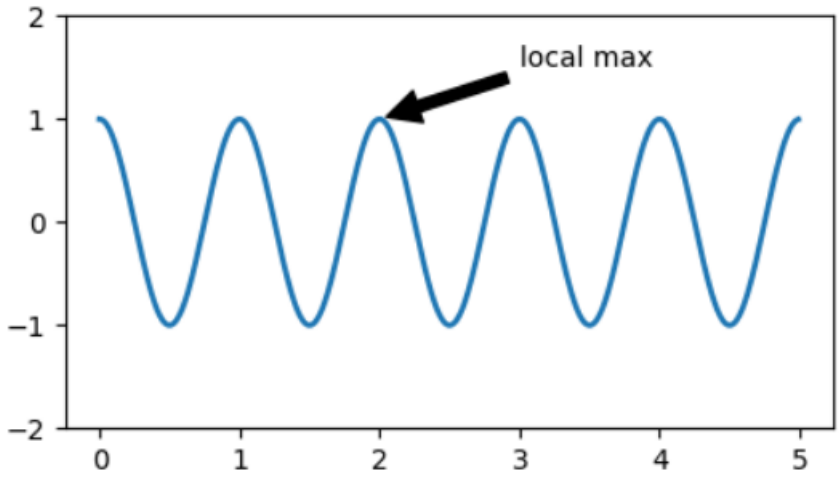
```
mu, sigma = 115, 15
x = mu + sigma * np.random.randn(10000)
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
# El histograma de la data
n, bins, patches = ax.hist(x, 50, density=1, facecolor='C0', alpha=0.75)

ax.set_xlabel('Length [cm]')
ax.set_ylabel('Probability')
ax.set_title('Aardvark lengths\n (not really)')
ax.text(75, .025, r'$\mu=115,\ \sigma=15$')
ax.axis([55, 175, 0, 0.03])
ax.grid(True);
```



Anotaciones:

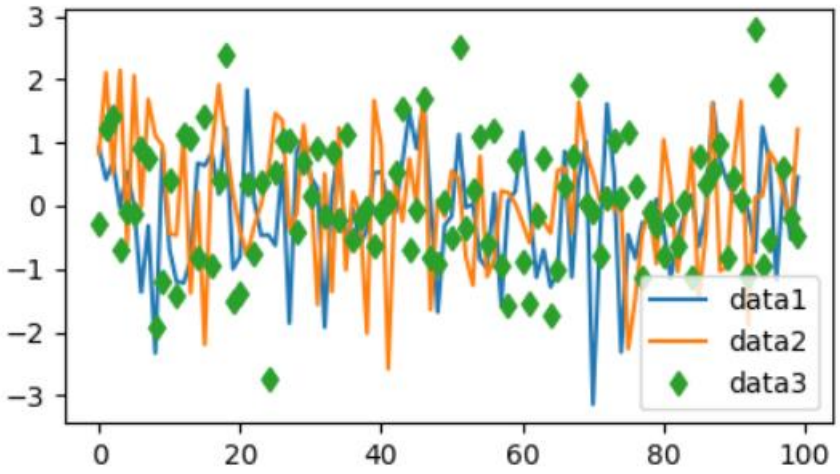
```
fig, ax = plt.subplots(figsize=(5, 2.7))

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

```
ax.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
            arrowprops=dict(facecolor='black', shrink=0.05))

ax.set_ylim(-2, 2);
```



Leyendas:

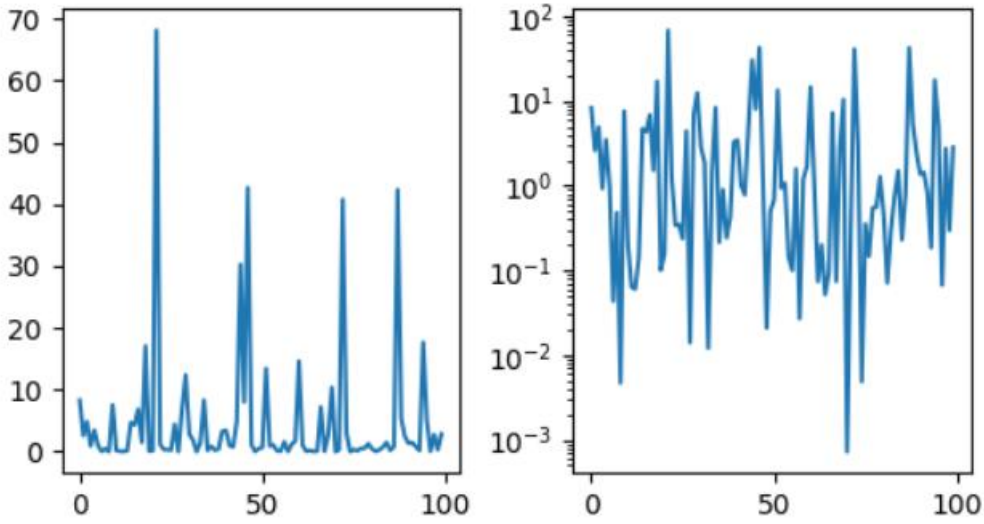
```
fig, ax = plt.subplots(figsize=(5, 2.7))
ax.plot(np.arange(len(data1)), data1, label='data1')
ax.plot(np.arange(len(data2)), data2, label='data2')
ax.plot(np.arange(len(data3)), data3, 'd', label='data3')
ax.legend();
```



Escalas:

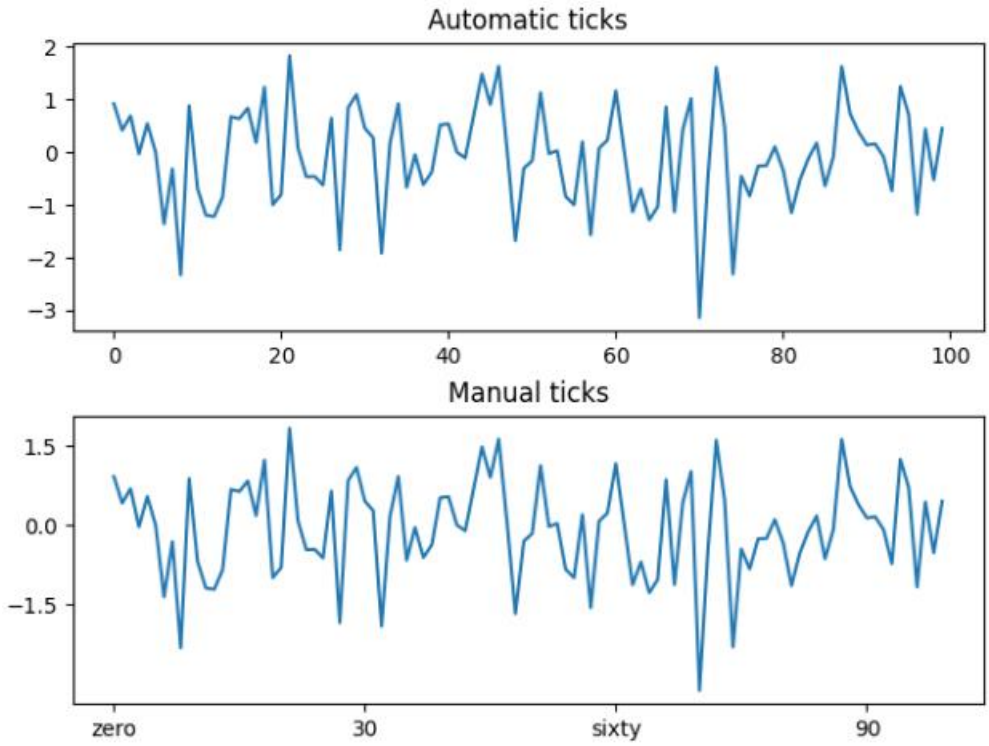
Esta librería brinda una escala no lineal, como lo son **loglog**, **semilogx**, y **semilogy**.

```
fig, axs = plt.subplots(1, 2, figsize=(5, 2.7), layout='constrained')
xdata = np.arange(len(data1))  # Hace un ordinal para esto.
data = 10**data1
axs[0].plot(xdata, data)

axs[1].set_yscale('log')
axs[1].plot(xdata, data);
```



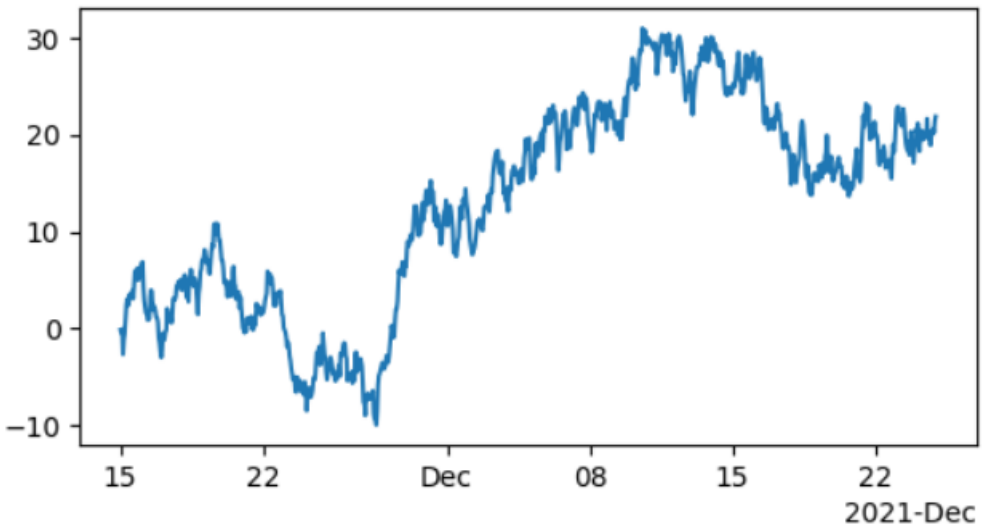
Localizadores de ticks y fomateadores:

```
fig, axs = plt.subplots(2, 1, layout='constrained')
axs[0].plot(xdata, data1)
axs[0].set_title('Automatic ticks')

axs[1].plot(xdata, data1)
axs[1].set_xticks(np.arange(0, 100, 30), ['zero', '30', 'sixty', '90'])
axs[1].set_yticks([-1.5, 0, 1.5])  # note that we don't need to specify labels
axs[1].set_title('Manual ticks');
```



Dibujos de datos y cadenas:

```
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
dates = np.arange(np.datetime64('2021-11-15'), np.datetime64('2021-12-25'),
                  np.timedelta64(1, 'h'))
data = np.cumsum(np.random.randn(len(dates)))
ax.plot(dates, data)
cdf = mpl.dates.ConciseDateFormatter(ax.xaxis.get_major_locator())
ax.xaxis.set_major_formatter(cdf);
```



Dibujo de cadenas:

```
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
categories = ['turnips', 'rutabaga', 'cucumber', 'pumpkins']

ax.bar(categories, np.random.rand(len(categories)));
```