Librerías

La librería "matplotlib" se usa para hacer gráficas https://matplotlib.org/

La librería "numpy" se usa para algebra lineal y manejar arrays https://numpy.org/

Otra librería útil para hacer gráficas es "seaborn" https://seaborn.pydata.org/

```
#Se importan las librerías
import matplotlib.pyplot as plt
import numpy as np
import math
```

Algunas referencias de apoyo:

Página de la función "plot" https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

Lista de markers de matplotlib https://matplotlib.org/stable/api/markers_api.html

Lista de linestyles https://matplotlib.org/stable/gallery/lines_bars_and_markers/linestyles.html

Lista de colores: https://matplotlib.org/stable/gallery/color/named_colors.html

Gráficas: plot, bar, barh y pie

plt.plot([x],[y], configurarcion)

plt.bar([x],[y])

plt.barh([x],[y])

plt.pie([y], labels = [], autopct="%0.0f%%")

plt.subplot(entero_3_digitos) Columnas, renglones y cuadrante

```
x = ["uno", "dos", "tres", "cuatro"]
y = [10, 20, 15, 5]
#plt.plot(x,y, "r-*")          # Elige el tipo de gráfico
plt.pie(y, labels = x, autopct="%0.0f%%")
plt.xlabel("Valores en X")
plt.ylabel("Valores en Y")
plt.title("Productos y Precios")
plt.show()
```

```
x = ["uno", "dos", "tres", "cuatro"]
y1 = [10, 20, 15, 5]
y2 = [500, 250, 150, 100]

#plt.plot(x, y1)
#plt.plot(x, y2)
plt.plot(x,y1,"rd",y2)
plt.xlabel("Valores en X")
plt.ylabel("Valores en Y")
plt.title("Precios y Ventas")
plt.show()
```

```
x = ["uno", "dos", "tres", "cuatro"]
y1 = [10, 20, 15, 5]
y2 = [500, 250, 150, 100]

plt.subplot(221)
plt.plot(x, y1, y2)
plt.xlabel("Productos")
plt.ylabel("Precios")
plt.title("Precios")
```

```
plt.subplot(222)
plt.bar(x, y2)
plt.xlabel("Productos")
plt.ylabel("Ventas")
plt.title("Ventas")

plt.show()
```

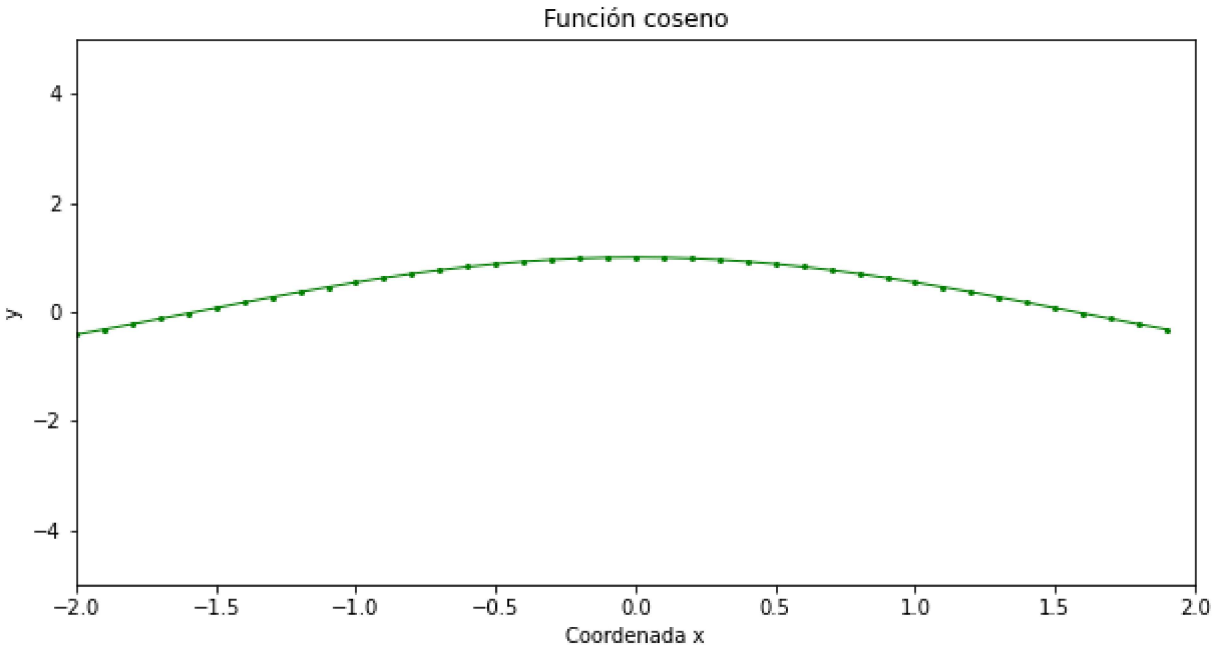
```
x = np.arange(-2, 2, 0.1)
y = 3*x+6        # La función que se quiere graficar, por ejemplo un polinomio
plt.plot(x,y)
plt.xlabel("Valores en X")
plt.ylabel("Valores del eje Y")
plt.title("Ecuación lineal")
plt.show()
```

```
x = np.arange(-2, 2, 0.1)
y = 3*x ** 2+x
plt.plot(x,y)
plt.show()
```

```
# Para crear el gráfico de una función, es necesario crear vectores que contengan puntos sobr
# Si se usa un número de puntos suficiente, el gráfico aparece como una curva continua.

x = np.arange(-2, 2, 0.1)
y = 3*x+6  #La función que se quiere graficar, por ejemplo un polinomio

fig, ax = plt.subplots(figsize = (10, 5)) #Se crean un "pad" (fig) donde se va a dibujar el g

ax.plot(x,y,color='green', marker='o', linestyle='-',linewidth=1, markersize=2)
ax.set_xlim(-2,2)
ax.set_ylim(-0.5,10)
plt.xlabel('Coordenada x')
plt.ylabel('y')
ax.set_title('Ecuación lineal')

plt.show() #para enseñar el plot
```

```
x = np.arange(-2, 2, 0.1)
y = np.cos(-x)  #La función que se quiere graficar, por ejemplo un polinomio
```

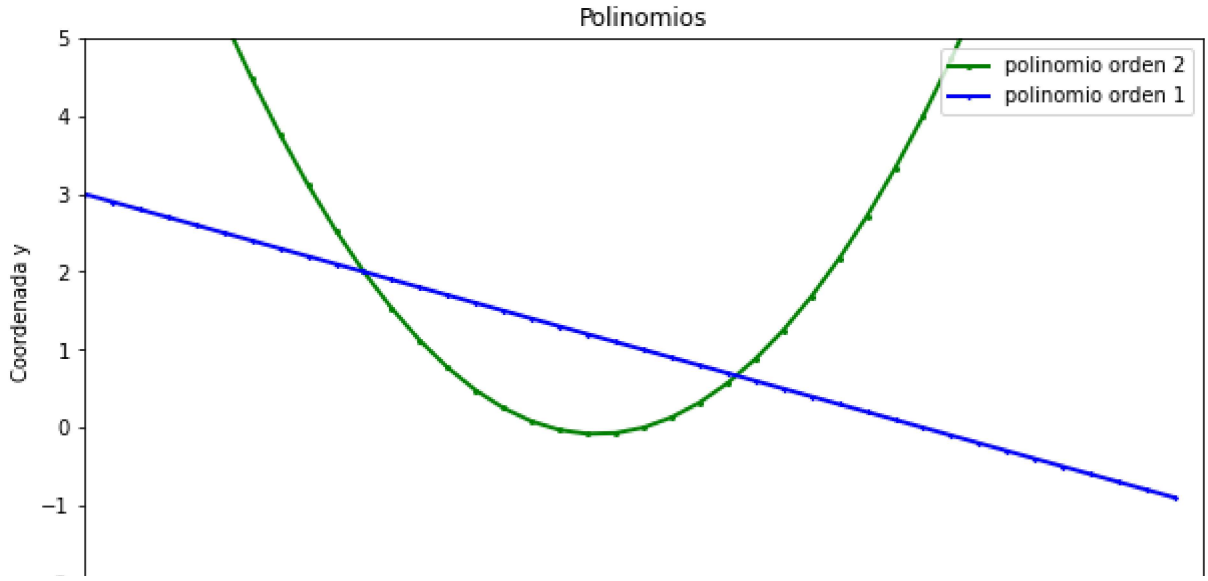
```
fig, ax = plt.subplots(figsize = (10, 5)) #Se crean un "pad" (fig) donde se va a dibujar el g

ax.plot(x,y,color='green', marker='o', linestyle='-',linewidth=1, markersize=2)
ax.set_xlim(-2,2)
ax.set_ylim(-5,5)
plt.xlabel('Coordenada x')
plt.ylabel('y')
ax.set_title('Función coseno')

plt.show() #para enseñar el plot
```



Dos funciones en el mismo plot

```
x = np.arange(-2, 2, 0.1)

y1 = 3*x ** 2+x  #Primera función que se quiere graficar, por ejemplo un polinomio de orden 2
y2 = -x+1        #Segunda función que se quiere graficar, por ejemplo un polinomio de orden 1

fig, ax = plt.subplots(figsize = (10, 5)) #Se crean un "pad" (fig) donde se va a dibujar el g

ax.plot(x,y1,color='green', marker='o', linestyle='-',linewidth=2, markersize=2,label='polino
ax.plot(x,y2,color='blue', marker='*', linestyle='-',linewidth=2, markersize=2,label='polinom
ax.set_xlim(-2,2)
ax.set_ylim(-2,5)
plt.xlabel('Coordenada x',size=12)
plt.ylabel('Coordenada y')
plt.legend(loc='upper right')
ax.set_title('Polinomios',size=12)
```

```
Text(0.5, 1.0, 'Polinomios')
```



Funciones log, exp, sin, cos

Las funciones de la librería **math** permiten calcular un valor a la vez (un solo valor x). Dado que aquí x es un vector, se debe usar **numpy** para calcular el valor de y.

```
x = np.arange(0,10,0.1)
y1 = np.exp(-x)
y2 = np.sqrt(x)

fig, ax = plt.subplots(figsize = (10, 5)) #Se crean un "pad" (fig) donde se va a dibujar el g

ax.plot(x,y1,color='green', marker='o', linestyle='-',linewidth=2, markersize=2,label='exp(x)
ax.plot(x,y2,color='red', marker='o', linestyle='-',linewidth=2, markersize=2,label='sqrt(x)'

plt.xlabel('Coordenada x',size=12)
plt.ylabel('y')
ax.set_title('Exponencial',size=12)
ax.legend()
```

```
x = np.arange(0,10,0.1)
y1 = np.exp(-x)
y2 = np.sqrt(x)

fig, ax = plt.subplots(2,2,figsize = (10, 5)) #Se crean un "pad" (fig) donde se va a dibujar

ax[0][0].plot(x,y1,color='green', marker='o', linestyle='-',linewidth=2, markersize=2,label='
ax[0][1].plot(x,y2,color='red', marker='o', linestyle='-',linewidth=2, markersize=2,label='sq
ax[1][0].plot(x,y1,color='green', marker='o', linestyle='-',linewidth=2, markersize=2,label='
ax[1][1].plot(x,y2,color='red', marker='o', linestyle='-',linewidth=2, markersize=2,label='sq
plt.xlabel('Coordenada x',size=12)
plt.ylabel('y')
```

Graficar funciones en 2D

Graficación: https://matplotlib.org/2.0.2/mpl_toolkits/mplot3d/tutorial.html

```
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
import numpy as np
```

```
fig, ax = plt.subplots(figsize=(8,6),subplot_kw={"projection": "3d"})

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)

# Dibujar la superficie
superficie = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Dar los valores para el eje z
ax.set_zlim(-1.01, 1.01)

# Agregar una barra de color, la cual mapea los valores a los colores
fig.colorbar(superficie, shrink=0.5, aspect=5)

plt.show()
```

Referencia: https://glowingpython.blogspot.com/2012/01/how-to-plot-two-variable-functions-with.html

```
import numpy as np
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show

def z_func(x,y):
 return (1-(x**2+y**3))*np.exp(-(x**2+y**2)/2)
```

```
x = np.arange(-3.0,3.0,0.1)
y = np.arange(-3.0,3.0,0.1)

X,Y = meshgrid(x, y)  # Generando el espacio
Z = z_func(X, Y)      # Evaluando la función

im = imshow(Z,cmap=cm.RdBu)    # Dibuja la función Z

#agrega etiquetas en la línea del contorno
cset = contour(Z,np.arange(-1,1.5,0.2),linewidths=2,cmap=cm.Set2)
cset = contour(Z,linewidths=2,cmap=cm.Set2)

clabel(cset,inline=True,fmt='%1.1f',fontsize=10)
colorbar(im) # agregando la barra del indicador de color (derecha)
```

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                       cmap=cm.RdBu,linewidth=0, antialiased=False)



fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```

Otros ejemplos...

```
import matplotlib.pyplot as plt

#plt.style.use('fivethirtyeight')   # Para https://matplotlib.org/3.5.0/gallery/style_sheets/
#plt.style.use('ggplot')     # https://matplotlib.org/3.5.0/gallery/style_sheets/ggplot.html
plt.style.use('grayscale')   # https://matplotlib.org/3.5.0/gallery/style_sheets/grayscale.ht

x = [1,2,3,4,5,6]
y = [2,4,1,5,2,6]

fig = plt.figure(figsize=(6, 3), dpi=75)
plt.plot(x, y, color='green', linestyle='dashed', linewidth = 3,
         marker='o', markerfacecolor='blue', markersize=12)

plt.ylim(0,8)
plt.xlim(0,8)

plt.xlabel('Eje x')
plt.ylabel('Eje y')

plt.title('Personalizando las gráficas :) ')

plt.show()
fig.savefig('figura_nueva.png',dpi=200)
```