1. En esta práctica Usted se familiarizará con el poderoso módulo Matplotlib que permite crear gráficos de variados tipos. Explore los distintos tipos de gráficos que este módulo puede crear visitando la galeria oficial del proyecto, siguiente este link. Seleccione un par de estos de ejemplos, ingresando al correspondiente link en la imagen, copie el código Python del ejemplo. Incorpórelo a un archivo Python y ejecútelo para que vea qué hace.

2. En este video disponible en Canvas, se explican los aspectos básicos de Matplotlib. Vaya desarrollando esta guía consultando simultaneamente este video y esta guía (que es un Jupyter Notebook que puede también descargar).

3. Escriba y ejecute el siguiente programa, que hace uso de `Numpy` y del módulo gráfico `Matplotlib`

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

d = np.genfromtxt("datos.txt")
x = d[:,0]
y = d[:,1]
plt.plot(x,y, marker="o", markersize=5, color="green",
        label="Datos experimentales")
plt.title("Voltaje versus Frecuencia")
plt.xlabel("Frecuencia $f$ [Hertz]")
plt.ylabel("Voltaje $V$ [Volt]")
plt.legend()
plt.savefig("g1.pdf")
```

   Este programa grafica los datos en las listas `x` e `y` usando círculos verdes, que guarda en el archivo `g1.pdf`.

4. Copie el archivo `g1.py` a `g2.py`, que en adelante usará para realizar pruebas.

5. La opción `marker="o"` indica que los puntos son representados por círculos. Note que, por defecto, estos puntos son unidos por rectas. Otros símbolos ("markers") disponibles son listados en la tabla 1. Por ejemplo, la opción `marker="s"` indica al comando `plot` que grafique cuadrados. Además, la opción `color` puede adoptar los valores `blue` (b), `green` (g), `red` (r), `cyan` (c), `magenta` (m), `yellow` (y), `black` (k) y `white` (w). Puede encontrar más colores listados aquí. Cambie los colores y símbolos del grafico en `g2.py` para familiarizarse con estas opciones.

6. Agregue una grilla (malla) a su gráfico usando el comando `plt.grid(True)` antes de `np.savefig`, y vea qué efecto tiene esto sobre el gráfico creado.

7. Cambie los límites del gráfico agregando los comandos

```
plt.xlim(0,90)
plt.ylim(0,15)
```

   y vea el cambio que produce.

`"."`	point
`","`	pixel
`"o"`	circle
`"v"`	`triangle_down`
`"^"`	`triangle_up`
`"<"`	`triangle_left`
`">"`	`triangle_right`
`"1"`	`tri_down`
`"2"`	`tri_up`
`"3"`	`tri_left`
`"4"`	`tri_right`
`"8"`	octagon
`"s"`	square
`"p"`	pentagon
`"*"`	star
`"h"`	hexagon1
`"H"`	hexagon2
`"+"`	plus
`"x"`	x
`"D"`	diamond
`"d"`	`thin_diamond`

Cuadro 1: Algunos símbolos disponibles para gráficar puntos con el comando `plot`. Ver este link para más detalles y símbolos.

8. Exporte el gráfico anterior directamente a formato `.png`, simplemente cambiando `plt.savefig("g2.pdf")` por `plt.savefig("g2.png")` en su programa `g2.py`. Comparta su lindo gráfico personalizado subiendo el archivo `p2.png` al foro de Teams de la práctica.