1. En esta práctica usted ejercitará algunas características muy básicas del poderoso módulo `matplotlib`, y en particular usando las funciones definidas en `matplotlib.pyplot`. Para ello, cargue el módulo usando:

   ```
   from matplotlib.pyplot import *
   ```

2. Escriba un script Python en el archivo `g1.py` con el siguiente contenido:

   ```
   # -*- coding: UTF-8 -*-
   from matplotlib.pyplot import *

   x=[5,20,40,60,80]
   y=[2.6,5.5,7,12.5,11.9]

   plot(x,y, marker="o", markersize=5, color="green", label="Datos experimentales")
   title("Voltaje versus Frecuencia")
   xlabel("Frecuencia $f$ [Hertz]")
   ylabel("Voltaje $V$ [Volt]")
   legend(loc=2) # esquina superior izquierda
   savefig("g1.pdf")
   ```

   Note que la primera línea es necesaria si se quieren incluir caracteres latinos (tildes, letra ñ, etc). Este programa grafica los datos en las listas `x` e `y` usando círculos verdes, que guarda en el archivo `g1.pdf`.

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

4. 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.

5. Cambie el comando `plot` anterior por

   ```
   plot(x,y, "v", markersize=5, color="green", label="Datos experimentales")
   ```

   ¿Qué diferencia produce esto en el gráfico final?

6. Verifique que lo anterior es equivalente a

   ```
   plot(x,y, "vg", markersize=5, label="Datos experimentales")
   ```

   Aquí la `g` en `"vg"` abrevia "green" (verde).

7. Agregue una grilla (malla) a su gráfico usando el comando `grid()` antes de `savefig`.

8. Cambie los límites del gráfico agregando los comandos

`"."`	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.

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

9. Prepare un programa `g3.py` que, usando también el módulo `numpy`, grafique la (curva continua correspondiente a la) función
$$y(x) = \frac{\sin(x)}{x} \tag{1}$$
en el intervalo $x \in [-30, 30]$, y que guarde el resultado en el archivo `g3.pdf`.

10. Pruebe cambiando el comando `plot` de su archivo script por el comando `fill`, con las mismas opciones (por ejemplo, sustituyendo `plot(x,y)` por `fill(x,y)`). ¿Qué efecto tiene este cambio?

11. Mejore su programa `g3.py` para que el gráfico incorpore todos los elementos necesarios para obtener un resultado aceptable (Título que indique la función graficada, ejes con nombres adecuados, grilla (opcional), etc.).

12. Finalmente, confeccione un programa `g4.py` que cargue y grafique (lo mejor que pueda) los datos en el archivo `datos.txt` usado en la guía 04. Envíe los archivos `g4.py` y `g4.pdf` al email del profesor G. Rubilar.

13. **Bonus Track (opcional)**: Es posible modificar el estilo de la figura completa agregando el comando `style.use("estilo")` al comienzo de los comandos que definen el gráfico. Aquí `"estilo"` es el nombre de uno de los estilos disponibles. Pruebe, por ejemplo, usar el estilo

"ggplot", agregando la línea `style.use("ggplot")` a uno de sus gráficos. Una lista completa de los estilos predefinidos en su instalalción de `Matplotlib` puede obtenerse con el comando `print(style.available)`

14. **Bonus Track (opcional)**: Además de `style.use`, las últimas versiones de `Matplotlib` incluyen el comando `xkcd`, que cambia el estilo de su gráfico al estilo del famoso webcomic xkcd. Este estilo es bastante útiles cuando usted quiere que su gráfico luzca como "hecho a mano". Vea cómo cambia uno de sus gráficos agregando `xkcd()` a su programa (antes del comando `plot`).