Programación Avanzada
Daniel Gregorio Longino
Tarea 14

Para el tema del módulo Matplotlib vamos a necesitar las siguientes librerías:

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  import datetime as dt
4  import seaborn as sns
5  import networkx as nx
```

1. General

Del módulo al que hemos renombrado `plt`:

1. usaremos el método `.title()` para añadir un título al plot.
2. usaremos el método `.xlabel()` para añadir una etiqueta al eje horizontal del plot 2D.
3. usaremos el método `.ylabel()` para añadir una etiqueta al eje vertical del plot 2D.
4. usaremos el método `.legend()` para mostrar una leyenda

2. Colores

Python trae los siguietes colores por defecto:

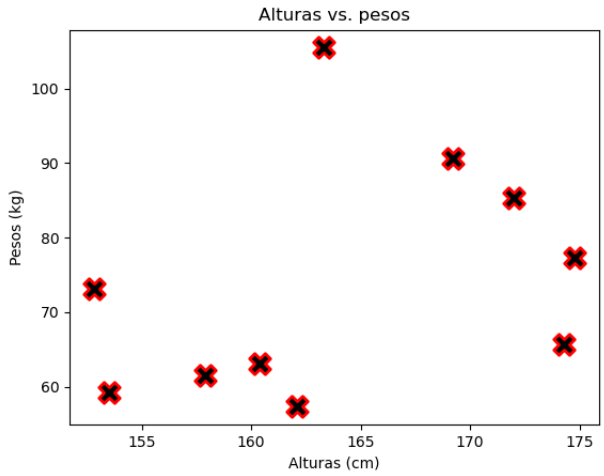
Nombre	Abreviatura	Color
`blue`	`b`	azul
`green`	`g`	verde
`red`	`r`	rojo
`cyan`	`c`	cian
`magenta`	`m`	magenta
`yellow`	`y`	amarillo
`black`	`k`	negro
`white`	`w`	blanco

3. Scatter Plot

Para hacer un gráfico nube de puntos, usamos el método `.scatter()` del módulo plt. Algunos de los parámetros de este método son:

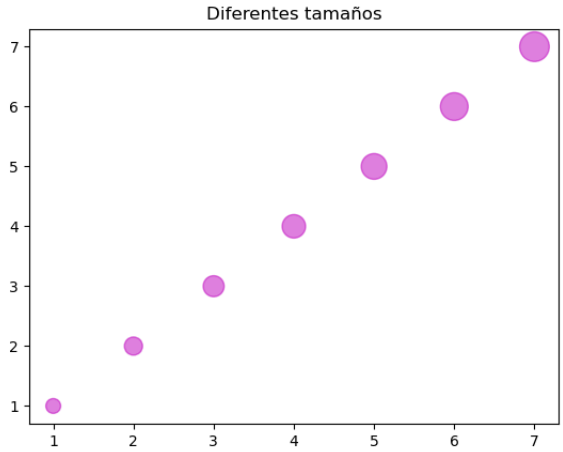
1. `x`: scalar, array o lista que indica la primera coordenada de las observaciones.
2. `y`: scalar, array o lista que indica la segunda coordenada de las observaciones.
3. `c`: para cambiar el color de relleno.
4. `edgecolors`: para cambiar el color del contorno.
5. `alpha`: para cambiar la transparencia.
6. `marker`: para cambiar la forma del punto.
7. `s`: para cambiar el tamaño de los puntos (se mide en puntos).
8. `linewidths`: para cambiar el grosor del contorno.

```
7  height = [174.3, 153.5, 162.1, 157.9, 174.8, 169.2, 172.0, 160.4, 152.8, 163.3]
8  weight = [ 65.7,  59.2,  57.3,  61.5,  77.3,  90.7,  85.4,  63.1,  73.2, 105.5]
9
10 plt.title("Alturas vs. pesos")
11 plt.xlabel("Alturas (cm)")
12 plt.ylabel("Pesos (kg)")
13 plt.scatter(x = height, y = weight,
14               c = "black", edgecolors = "red", alpha = 1,
15               marker = "X", s = 200, linewidths = 2)
16 plt.show()
```



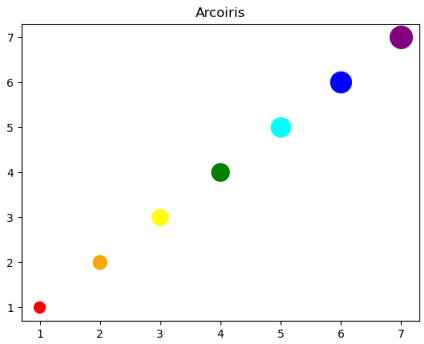
Observación. Podemos dibujar cada punto de un tamaño, pasando una lista al parámetro `s`:

```
 7  x = [1, 2, 3, 4, 5, 6, 7]
 8  sizes = [100, 150, 200, 250, 300, 350, 400]
 9
10  plt.scatter(x = x, y = x, s = sizes, c = "m", alpha = 0.5)
11  plt.title("Diferentes tamaños")
12  plt.show()
```



Observación. Podemos pintar un punto de cada color, pasando como parámetro `c` una lista de colores:

```
 7  x = [1, 2, 3, 4, 5, 6, 7]
 8  colors = ["red", "orange", "yellow", "green", "cyan", "blue", "purple"]
 9  sizes = [100, 150, 200, 250, 300, 350, 400]
10
11  plt.scatter(x = x, y = x, c = colors, s = sizes)
12  plt.title("Arcoiris")
13  plt.show()
```



4. Line Plot

Para hacer un gráfico line plot, usamos el método `.plot()` del módulo plt. Algunos de los parámetros de este método son:

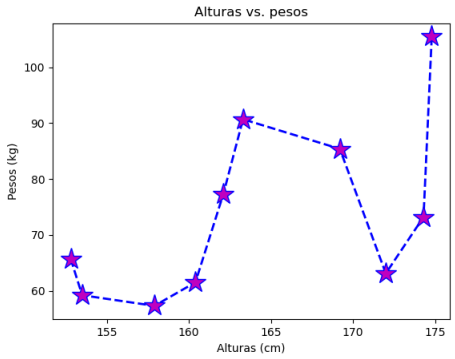
1. `x`: scalar, array o lista que indica la primera coordenada de las observaciones.
2. `y`: scalar, array o lista que indica la segunda coordenada de las observaciones.
3. `color`: para cambiar el color de relleno. También podemos referirnos a este parámetro por su diminutivo `c`.
4. `fmt`: para establecer un formato básico rápidamente como string. Por ejemplo, "`or`" son círculos rojos. El orden recomendado para introducir el formato de este modo es "`[marker][linestyle][color]`" aunque también se admite "`[color][marker][linestyle]`".
5. `linewidth`: para cambiar el grosor de la línea. También podemos referirnos a este parámetro por su diminutivo `lw`.
6. `linestyle`: para cambiar el estilo de la línea. También podemos referirnos a este parámetro por su diminutivo `ls`.
7. `alpha`: para cambiar la transparencia.
8. `marker`: para cambiar la forma del punto. Si no indicamos este parámetro, no se dibujan los puntos.
9. `markersize`: para cambiar el tamaño de los puntos. También podemos referirnos a este parámetro por su diminutivo `ms`.
10. `markeredgecolor`: para cambiar el color del contorno del punto. También podemos referirnos a este parámetro por su diminutivo `mec`.

11. `markerfacecolor`: para cambiar el color de relleno del punto. También podemos referirnos a este parámetro por su diminutivo `mfc`.

Para el parámetro `fmt`, las opciones disponibles para `marker` y `linestyle` son

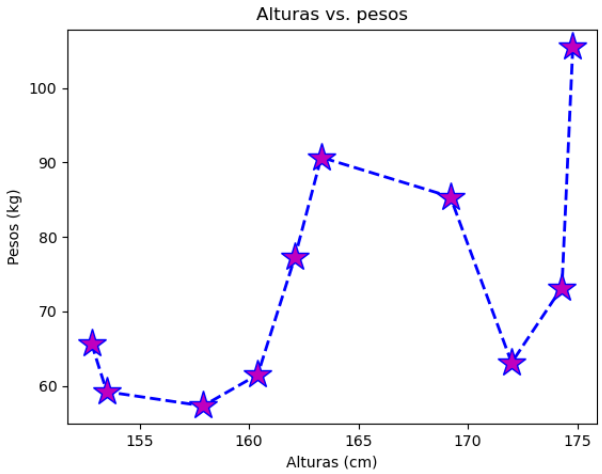
marker	forma	linestyle	estilo de línea
" ."	punto	"-" o "solid"	sólido
","	píxel	"- -" o "dashed"	discontinuo
"o"	círculo	"-." o "dashdot"	guión-punto
"v"	triángulo hacia abajo	":" o "dotted"	puntos
"^"	triángulo hacia arriba	"None"	sin línea
"<"	triángulo hacia izquierda		
">"	triángulo hacia derecha		
"1"	tri hacia abajo		
"2"	tri hacia arriba		
"3"	tri hacia izquierda		
"4"	tri hacia derecha		
"s"	cuadrado		
"p"	pentágono		
"*"	estrella		
"h"	hexágono 1		
"H"	hexágono 2		
"+"	cruz		
"x"	x		
"D"	diamante		
"d"	diamante fino		
"|"	barra vertical		
"_"	barra horizontal		

```
 9  plt.title("Alturas vs. pesos")
10  plt.xlabel("Alturas (cm)")
11  plt.ylabel("Pesos (kg)")
12  plt.plot(sorted(height), weight,
13                  c = "blue", ls = "--", lw = 2,
14                  marker = "*", ms = 20, mfc = "m")
15  plt.show()
```



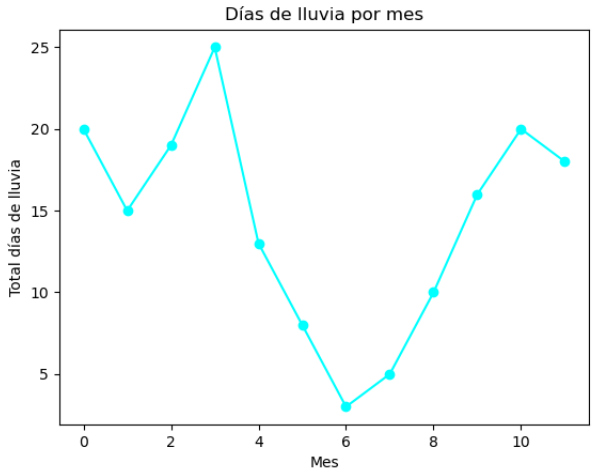
Podríamos obtener el mismo resultado usando el parámetro `fmt`.

```
 9  plt.title("Alturas vs. pesos")
10  plt.xlabel("Alturas (cm)")
11  plt.ylabel("Pesos (kg)")
12  plt.plot(sorted(height), weight, "*--b", lw = 2,
13           markersize = 20, mfc = "m")
14  plt.show()
```



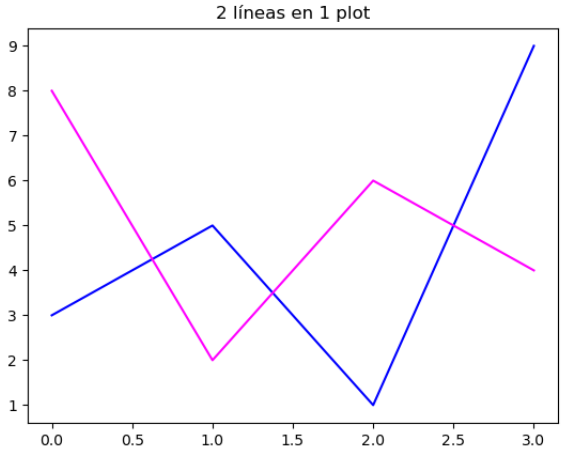
Observación. Si no indicamos parámetro `x`, se consideran como primeras coordenadas los números enteros $0, 1, 2, \ldots, n-1$, siendo n el número total de observaciones:

```
 6  rainy_days = [20, 15, 19, 25, 13, 8, 3, 5, 10, 16, 20, 18]
 7
 8  plt.title("Días de lluvia por mes")
 9  # (0 = Enero, 1 = Febrero, ..., 11 = Diciembre)
10  plt.xlabel("Mes")
11  plt.ylabel("Total días de lluvia")
12  plt.plot(rainy_days, c = "cyan", marker = "o")
13  plt.show()
```



Observación. En un mismo plot podemos dibujar más de una línea:

```
 6  l1 = np.array([3, 5, 1, 9])
 7  l2 = np.array([8, 2, 6, 4])
 8
 9  plt.plot(l1, c = "blue")
10  plt.plot(l2, c = "magenta")
11  plt.title("2 líneas en 1 plot")
12  plt.show()
```



5. Series temporales con `matplotlib`

Para representar series temporales con `matplotlib.pyplot`, usamos el método `.plot_date()`

```
dates = ["1/9/2020", "2/9/2020", "3/9/2020", "4/9/2020", "5/9/2020",
         "6/9/2020", "7/9/2020", "8/9/2020", "9/9/2020", "10/9/2020",
         "11/9/2020", "12/9/2020", "13/9/2020", "14/9/2020", "15/9/2020",
         "16/9/2020", "17/9/2020", "18/9/2020", "19/9/2020", "20/9/2020",
         "21/9/2020", "22/9/2020", "23/9/2020", "24/9/2020", "25/9/2020",
         "26/9/2020", "27/9/2020", "28/9/2020", "29/9/2020", "30/9/2020"]

x = [dt.datetime.strptime(d, "%d/%m/%Y").date() for d in dates]
y = np.random.randint(10000, 20000, len(x))

plt.title("Total Ventas en Septiembre 2020")
plt.xlabel("Día del mes")
plt.xticks(rotation = 90)
plt.ylabel("Total ventas")
plt.plot_date(x, y, c = "blue", ls = "--", lw = 2, tz = "Europe/Madrid")
plt.show()
```