Matplotlib_Ricard_Perpinyà

March 28, 2021

1 Resum de llibreria *Matplotlib* ~ Ricard Perpinyà Alòs

Per importar la llibreria matplotlib.pyplot, i fer que els gràfics ens apareguin al notebook:

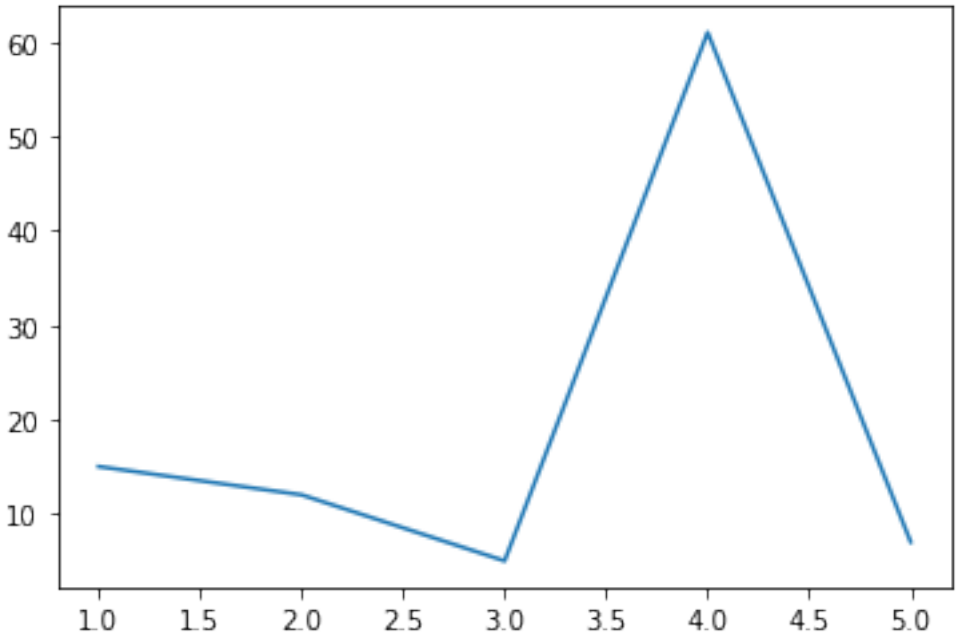
[4]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

1.1 1. plot()

Aquest mètode serveix per dibuixar gràfiques a partir de conjunts de punts. Si no indiquem res, ens uneix els punts i ens crea línies contínues. Si volem punts o un altre tipus de línia hem d'indicar-ho amb `'o'` (punts) o `'--'` (línies discontínues).

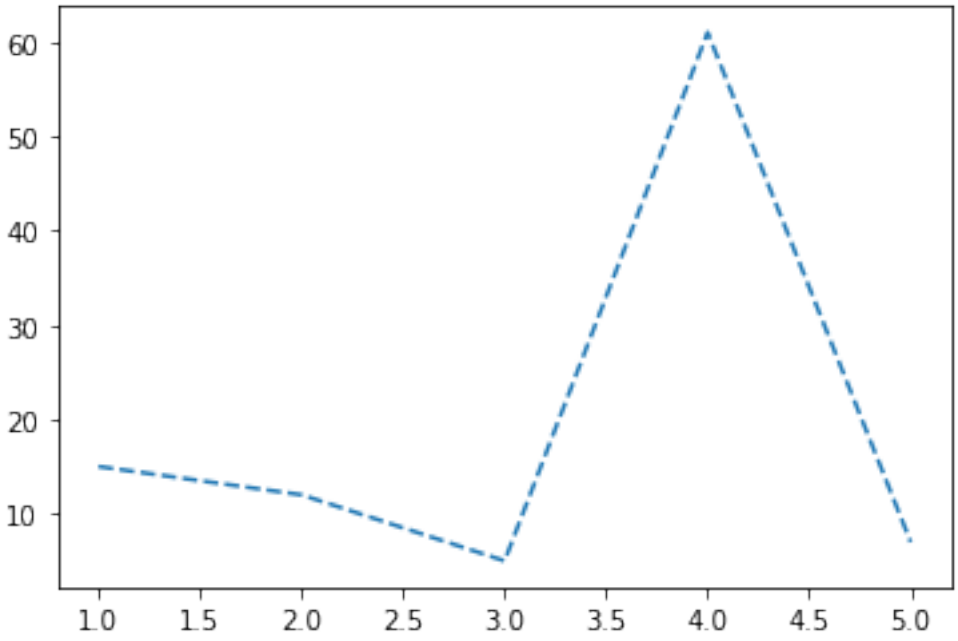
1.1.1 Exemple 1: línies contínues

[6]:
```
x = [1,2,3,4,5]
y = [15,12,5,61,7]

plt.plot(x,y) #també es pot escriure "-" si volem una línia contínua

plt.show()
```



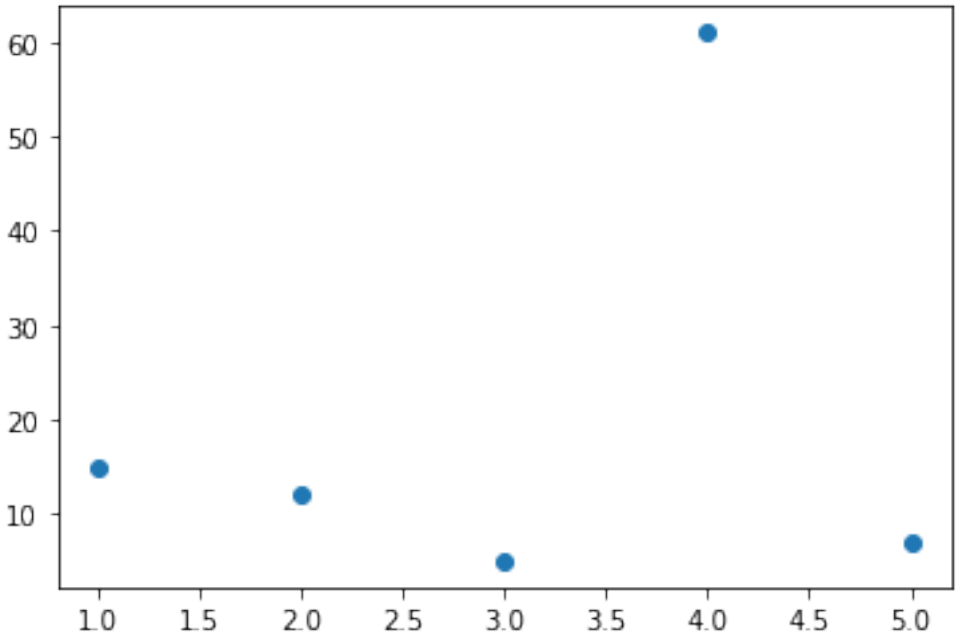
1.1.2 Exemple 2: Línies discontínues

[7]:
```
#Exemple 2: línies discontínues

x = [1,2,3,4,5]
y = [15,12,5,61,7]

plt.plot(x,y,"--")

plt.show()
```



1.1.3 Exemple 3: punts

[8]:
```
x = [1,2,3,4,5]
y = [15,12,5,61,7]

plt.plot(x,y,"o")

plt.show()
```



1.2 1.1 Opcions de `plot()`

La forma abreviada és la següent:

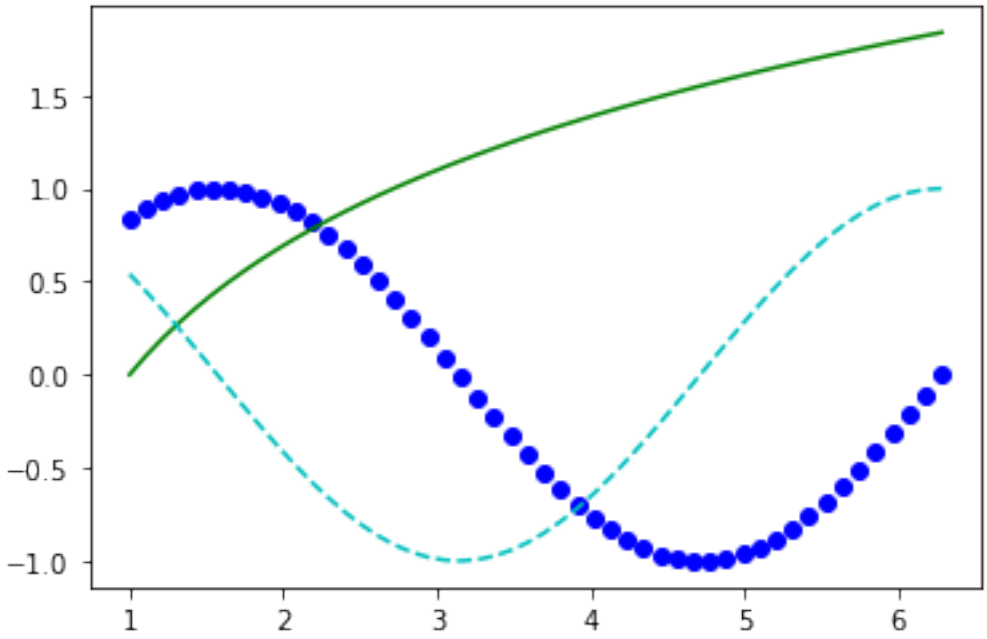
- “b” - blue
- “g” - green
- “r” - red
- “c” - cyan

etc.

El tipus de traçat (`"o"`,`"--"`,`"-"`) ja l’hem vist en un dels apartats anteriors.

1.2.1 Exemple 4: Forma abreviada

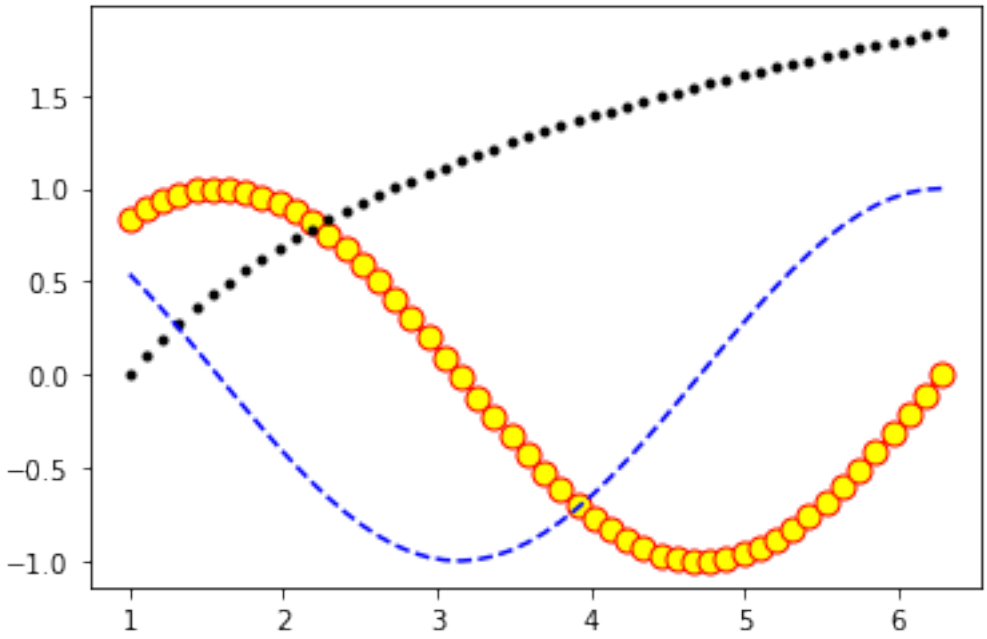
[9]:
```
import numpy as np

x = np.linspace(1, 2*np.pi, 50)
y1 = np.sin(x)
y2 = np.log(x)
y3 = np.cos(x)

plt.plot(x,y1,"bo")
plt.plot(x,y2,"g-")
plt.plot(x,y3,"c--")
plt.show()
```



La forma expandida o completa és la següent:

1.2.2 Forma completa:

[24]:
```
import numpy as np

x = np.linspace(1, 2*np.pi, 50)
y1 = np.sin(x)
y2 = np.log(x)
y3 = np.cos(x)

plt.plot(x,y1, color = "red", linestyle = "dashed", marker = "o",␣
 ↪markerfacecolor = "yellow", markersize = 9, linewidth = 2)
plt.plot(x,y2, color = "black", linestyle = "", marker = ".")
plt.plot(x,y3, color = "blue", linestyle = "--",)
plt.show()
```



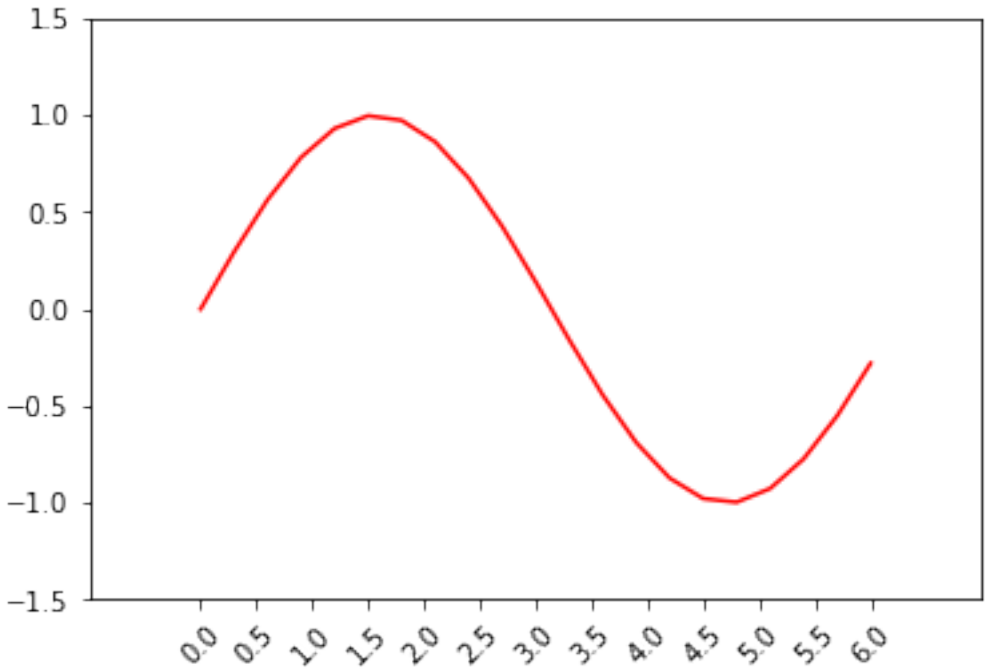
1.3 1.2 Control dels eixos

Les següents funcions són útils per editar l'aparença dels eixos:

- **`xlim(a,b)`** límits de l'eix x
- **`ylim(a,b)`** límits de l'eix y
- **`axis(a,b,c,d)`** requadre amb els límits dels eixos
- **`xticks()`** marcadors de l'eix x
- **`yticks()`** marcadors de l'eix y
- **`xscale()`** escala de l'eix x (lineal, logarítmica)
- **`yscale()`** escala de l'eix y (lineal, logarítmica)

1.3.1 Exemple 6: Control dels eixos

[10]:
```
x = np.arange(0, 2*np.pi, 0.3)
y1 = np.sin(x)

plt.xlim(-1,7) #Si anem canviant aquests valors, el límit de les x canviarà
plt.ylim(-1.5,1.5) #El mateix però per l'eix y

plt.xticks(np.arange(0,2*np.pi,0.5), rotation = 45) #Editem els ticks de les x␣
 ↪i els rotem amb "rotation"

plt.plot(x,y1, color = "red", linestyle = "-")
plt.show()
```



1.4 2. `subplot()`

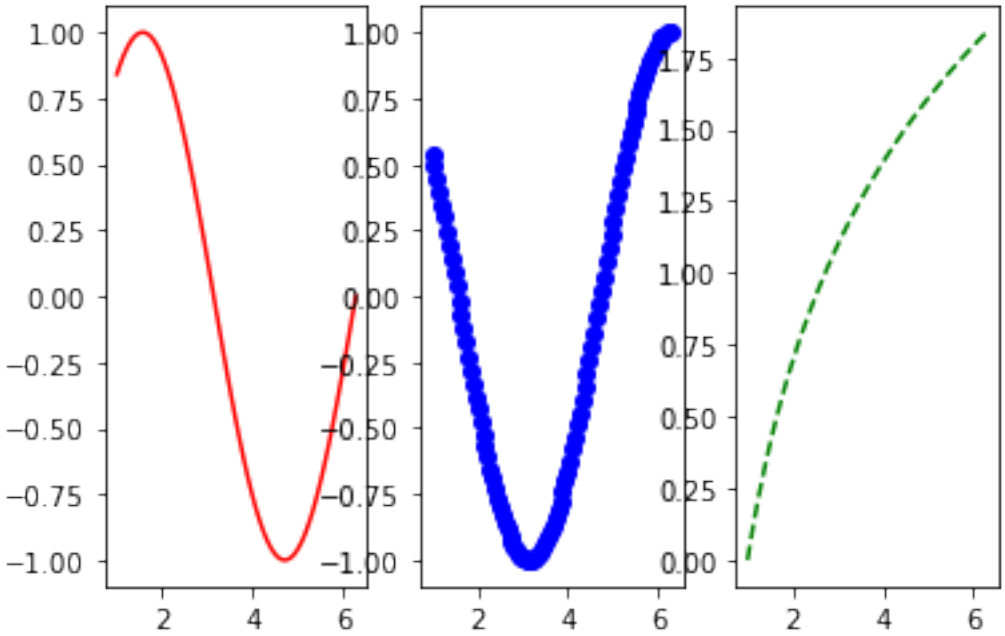
Es poden crear figures amb diversos gràfics usant la funció `subplot()`. Aquesta funció pot rebre 3 arguments:

`subplot(nº files, nº columnes, nombre casella on treballem)`

1.4.1 Exemple 7: subplot()

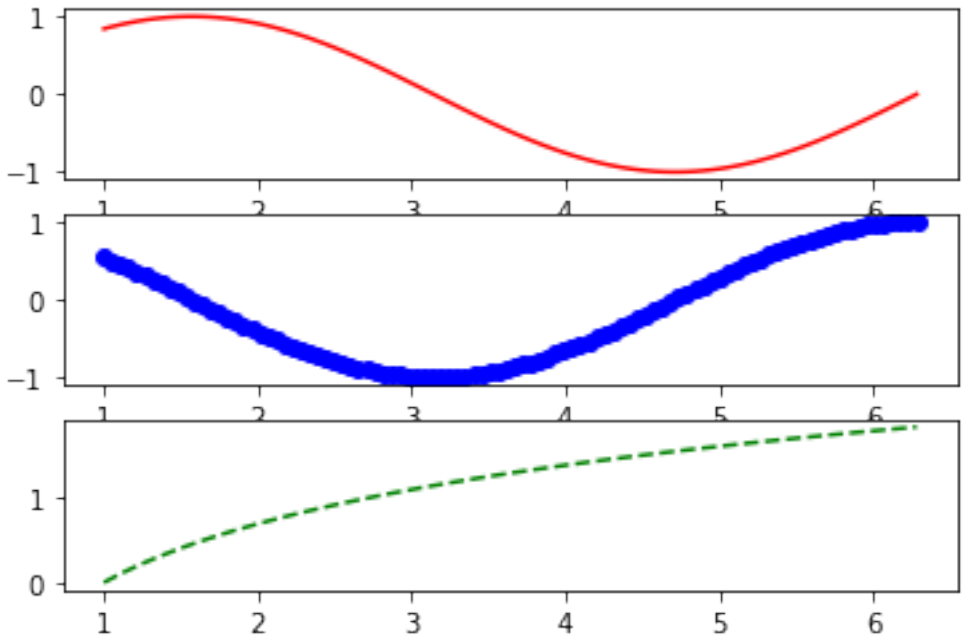
[64]:
```
x = np.linspace(1, 2*np.pi, 100)

y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.log(x)

#Ho representarem en 3 gràfiques a la mateixa línia

plt.subplot(1,3,1)
plt.plot(x,y1,"r-") #y1

plt.subplot(1,3,2)
plt.plot(x,y2,"bo") #y2

plt.subplot(1,3,3)
plt.plot(x,y3,"g--") #y3
```

```
plt.show()
```



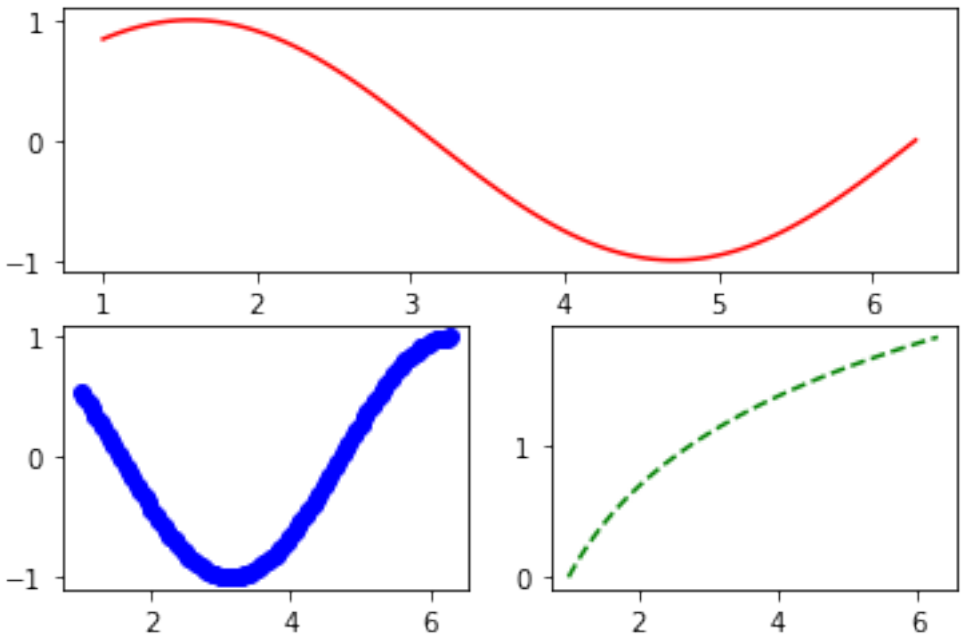
1.4.2 Exemple 8: subplot() 2

[65]:
```
x = np.linspace(1, 2*np.pi, 100)

y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.log(x)

#Ho representarem en 3 gràfiques en 3 files, 1 columna

plt.subplot(3,1,1)
plt.plot(x,y1,"r-") #y1

plt.subplot(3,1,2)
plt.plot(x,y2,"bo") #y2

plt.subplot(3,1,3)
plt.plot(x,y3,"g--") #y3

plt.show()
```



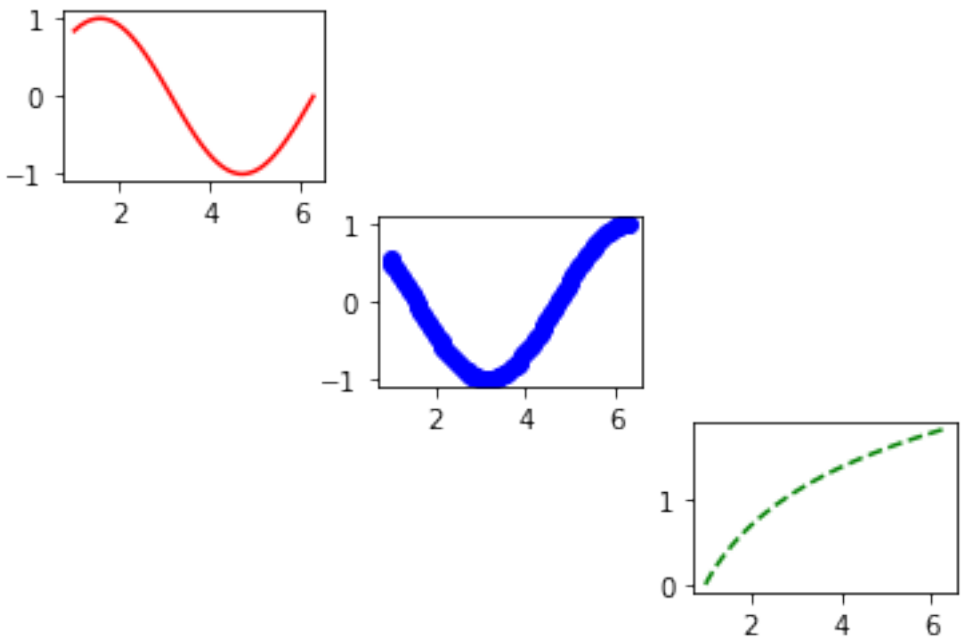
1.4.3 Exemple 9: subplot() 3

[83]:
```
x = np.linspace(1, 2*np.pi, 100)

y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.log(x)

#Ho representarem en 3 gràfiques en 2 files, 1 gràfica a dalt i 2 a baix

plt.subplot(2,1,1)
plt.plot(x,y1,"r-") #y1

plt.subplot(2,2,3)
plt.plot(x,y2,"bo") #y2

plt.subplot(2,2,4)
plt.plot(x,y3,"g--") #y3

plt.show()
```



1.4.4 Exemple 10: subplot() 4

[92]:
```
x = np.linspace(1, 2*np.pi, 100)

y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.log(x)

#Ho representarem en 3 gràfiques, escalonades

plt.subplot(3,3,1)
plt.plot(x,y1,"r-") #y1

plt.subplot(3,3,5)
plt.plot(x,y2,"bo") #y2

plt.subplot(3,3,9)
plt.plot(x,y3,"g--") #y3

plt.show()
```



1.5 3. Text

Aquestes funcions permeten escriure text a les gràfiques:

- **`xlabel()`** etiqueta eix x
- **`ylabel()`** etiqueta eix y
- **`title()`** títol del gràfic

També es pot utilitzar **`text(x,y,str)`** de forma més general.

Es pot utilitzar LaTeX en les quatre funcions anteriors escrivint entre dos signes `$`.

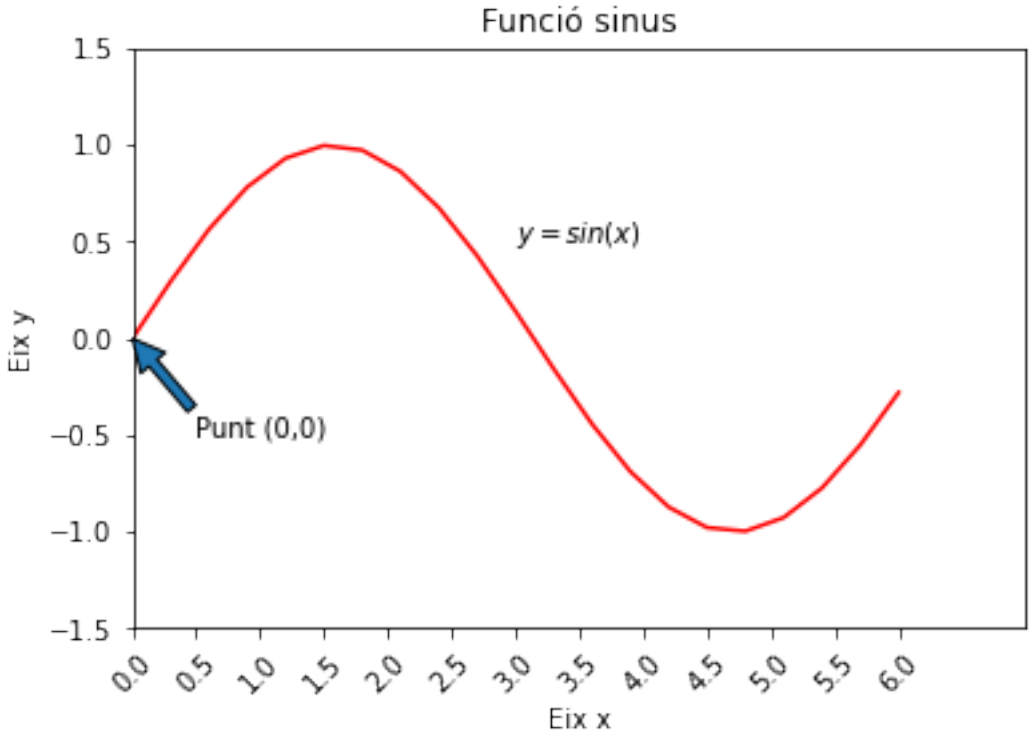
Per escriure fletxes amb **anotacions** utilitzarem **`plt.annotate("Text",xy=(a,b), xytext(c,d), arrowprops=dict(facecolor="xxx"...))`**. L'argument d'`arrowprops=dict(facecolor="xxx"...)` és opcional, si no li indiquem res ens generarà una fletxa genèrica.

1.5.1 Exemple 11: Text

Ficarem text al plot de l'exemple 6:

[105]:
```
x = np.arange(0, 2*np.pi, 0.3)
y1 = np.sin(x)

plt.xlim(0,7) #Si anem canviant aquests valors, el límit de les x canviarà
plt.ylim(-1.5,1.5) #El mateix però per l'eix y
```

```
plt.xticks(np.arange(0,2*np.pi,0.5), rotation = 45) #Editem els ticks de les x
 ↪i els rotem amb "rotation"

plt.xlabel("Eix x")
plt.ylabel("Eix y")
plt.title("Funció sinus")
plt.text(3,0.5,"$y=sin(x)$")#Utilitzem LaTeX
plt.annotate("Punt (0,0)",xy=(0,0), xytext=(0.5,-0.5), arrowprops=dict())

plt.plot(x,y1, color = "red", linestyle = "-")
plt.show()
```



1.6 Guardar les figures en un fitxer

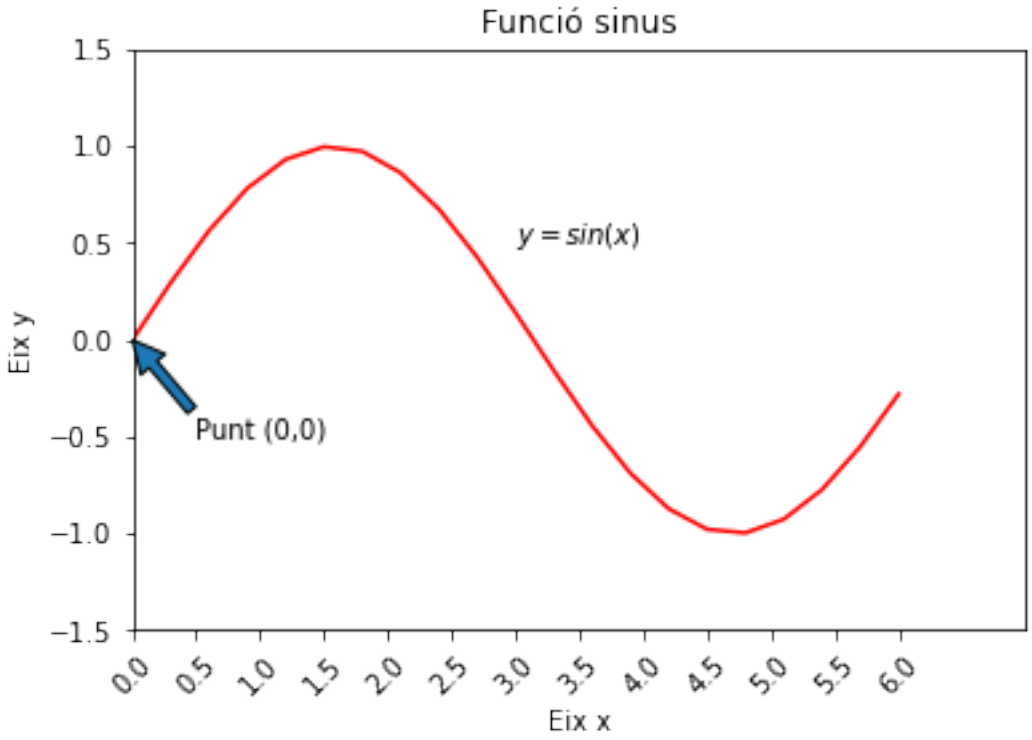
La funció **`plt.savefig("nom fitxer", format = format del fitxer, dpi = resolució en punts per polzada)`** serveix per guardar la figura generada en un fitxer.

1.6.1 #Exemple 12: Guardar figura

Guardarem la figura de l'exemple 11 en un fitxer PDF.

[7]:
```
x = np.arange(0, 2*np.pi, 0.3)
y1 = np.sin(x)
```

```
plt.xlim(0,7) #Si anem canviant aquests valors, el límit de les x canviarà
plt.ylim(-1.5,1.5) #El mateix però per l'eix y

plt.xticks(np.arange(0,2*np.pi,0.5), rotation = 45) #Editem els ticks de les x␣
 ↪i els rotem amb "rotation"

plt.xlabel("Eix x")
plt.ylabel("Eix y")
plt.title("Funció sinus")
plt.text(3,0.5,"$y=sin(x)$")#Utilitzem LaTeX
plt.annotate("Punt (0,0)",xy=(0,0), xytext=(0.5,-0.5), arrowprops=dict())

plt.plot(x,y1, color = "red", linestyle = "-")
plt.savefig("Sinus", format="pdf", dpi = 300)
plt.show()
```



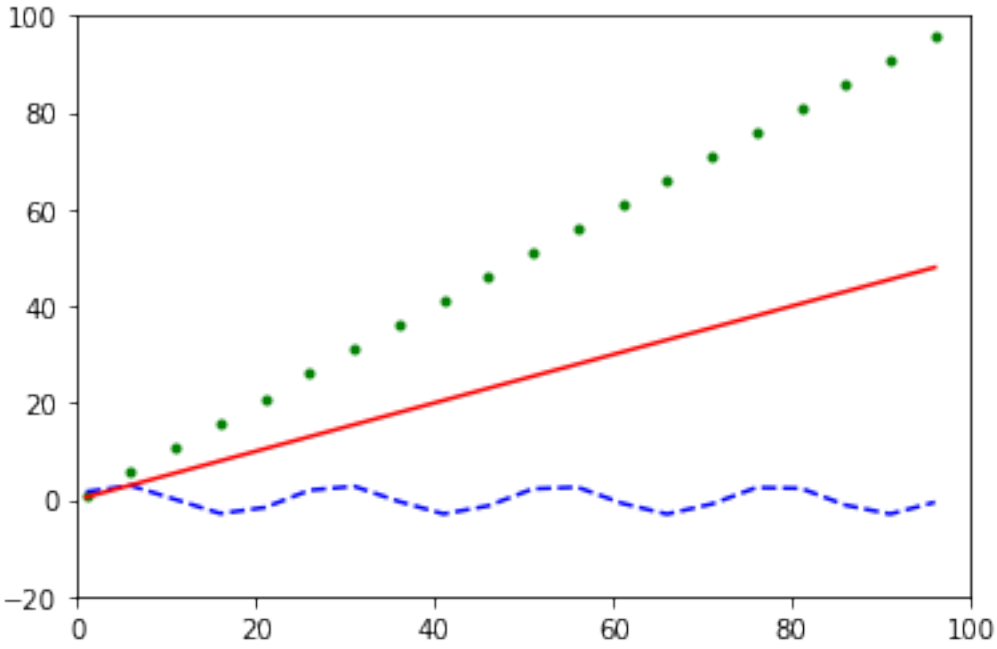
1.7 Exercicis

1.7.1 Exercici 1

Genera un array 1D de valors en el rang (1,100,5) i representa en un gràfic conjunt: els seus valors amb punts de color verd, el triple del cosinus amb una línia a traços de color blau i la meitat del

valor dels seus elements amb una línia contínua de color vermell.

[15]:
```
x = np.arange(1,100,5)

y1 = x
y2 = 3*np.cos(x)
y3 = x/2

plt.xlim(0,100)
plt.ylim(-20,100)
plt.plot(x,y1, color = "green", linestyle = "", marker = ".")
plt.plot(x,y2, color = "blue", linestyle = "--")
plt.plot(x,y3, color = "red", linestyle = "-")
plt.show()
```

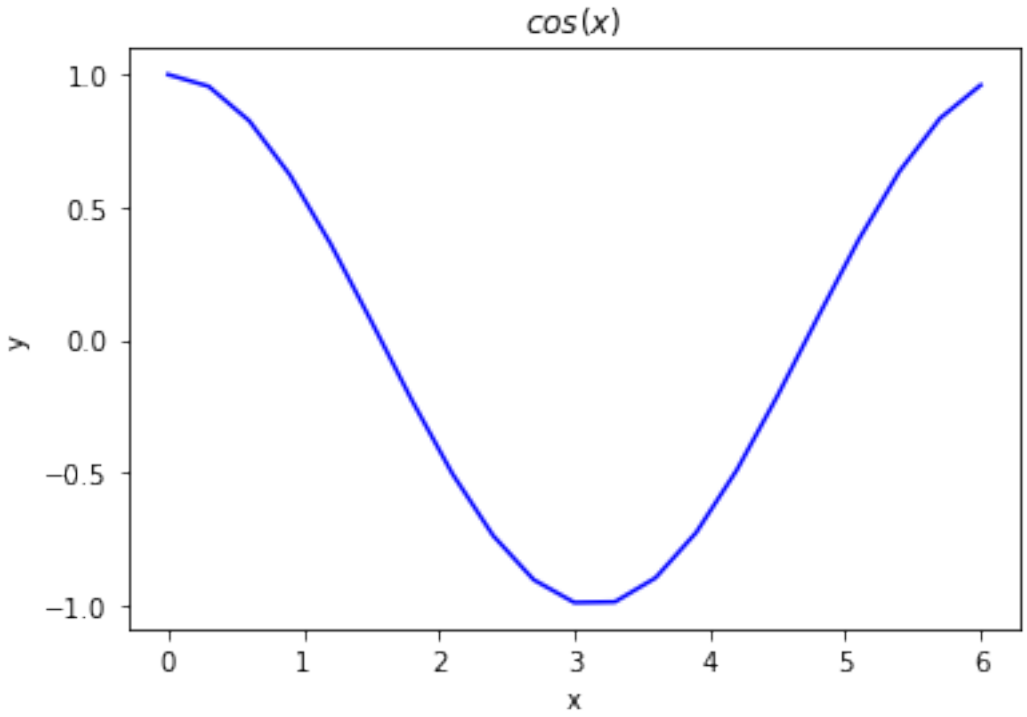


1.7.2 Exercici 2

Dibuixa la funció cos(x) per a x en el rang (0,2pi,0.3). Indica quins són els eixos x i y i posa-hi un títol.

[48]:
```
x = np.arange(0,2*np.pi,0.3)
y = np.cos(x)

plt.xlabel("x")
plt.ylabel("y")
plt.title("$cos(x)$")
```

```
#Canviaré els ticks de l'eix y per tal d'imitar la figura de l'enunciat
plt.yticks(np.linspace(-1.,1.,5))

plt.plot(x,y, color = "blue", linestyle = "-")
plt.show()
```

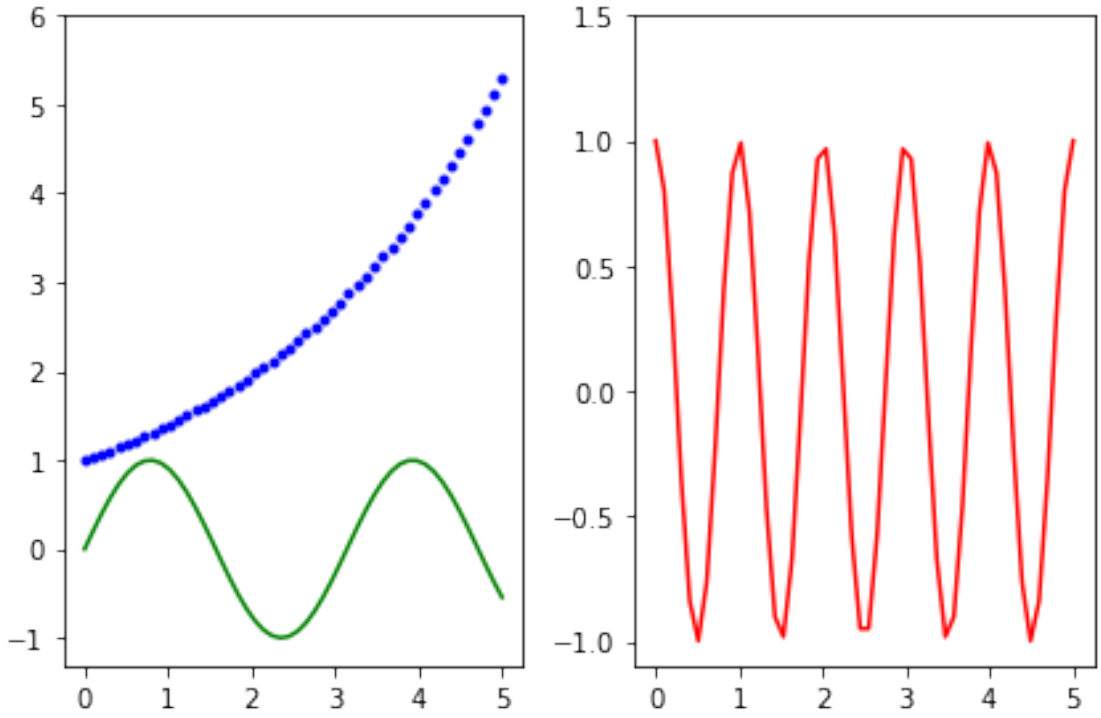


1.7.3 Exercici 3

Dibuixa dos gràfics un al costat de l'altre. El primer ha de contenir superposades les gràfiques de les funcions $y = e^{\frac{x}{3}}$ de color blau traçada a punts i $y = sin(2x)$ de color verd amb línia contínua; el segon, la funció $y = cos(2\pi x)$ de color vermell i línia contínua.

[70]:
```
x = np.linspace(0.,5.,50) #Definim un linspace a les x

y1 = np.exp(x/3) #Definim les funcions sobre x
y2 = np.sin(2*x)
y3 = np.cos(2*np.pi*x)

plt.subplot(1,2,1) #Subplot figura 1
plt.plot(x,y1, "b.")
plt.plot(x,y2, "g-")
plt.xticks(np.arange(0,6,1)) #Edito els eixos perquè quedin com a l'enunciat
```

```
plt.yticks(np.arange(-1,7,1))

plt.subplot(1,2,2) #Subplot figura 2
plt.plot(x,y3,"r-")
plt.xticks(np.arange(0,6,1))
plt.yticks(np.arange(-1,2,0.5))

plt.tight_layout() #Aquesta comanda serveix per separar els gràfics

plt.show()
```



1.7.4 Referències

1. 'Programació en Python', José M. Gómez, Ricardo Graciani, Manuel López, Xavier Luri, Sònia Estradé, Angela Rossell, Universitat de Barcelona (2016).
2. https://matplotlib.org
3. Syllabus INFO-H-100 Informatique', T. Massart, Universit'e Libre de Bruxelles (2016).

[ ]: