Authors Sonia Estradé
José M Gómez
Ricardo Graciani
Franc Guell
Manuel López
Xavier Luri
Josep Sabater

Introducció a Matplotlib

Matplotlib és una llibreria de python per a la creació de gràfics 2D que inclou una gran varietat de funcions i eines. Veurem aquí només una petita introducció a les seves funcionalitats i us podeu remetre a la pàgina oficial de la llibreria, http://matplotlib.org/ (http://matplotlib.org/) i en particular al tutorial de **pyplot**, una de les seves components que discutirem en aquest document:

http://matplotlib.org/users/pyplot_tutorial.html (http://matplotlib.org/users/pyplot_tutorial.html)

matplotlib.pyplot proporciona funcions que faciliten la creació de gràfics del tipus habitualment usat en física i funciona de forma similar a MATLAB. Per a usar-la haurem de fer la importació següent:

```
import matplotlib.pyplot
```

Creació d'un gràfic bàsic: comanda `plot()`

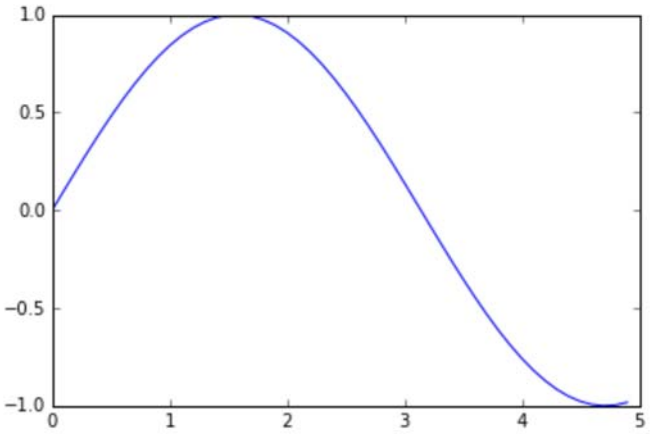
La funció `plot()` permet fer una gràfica d'un conjunt de punts. L'ús més senzill és a partir de dues llistes que donin els valors de `x` i els valors de `y` dels punts que es passen com a arguments, com es veu en el següent exemple. Per defecte els punts s'uneixen sequencialment amb línies, de manera que s'obté una representació en forma de linia contínua.

In [2]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (0,5) i a partir d'ells valors de sin(x)
x = np.arange(0, 5, 0.1);
y = np.sin(x)

# Representem els punts
plt.plot(x, y)
```

Out[2]: `[<matplotlib.lines.Line2D at 0x389bd50>]`



Nota: si només es dóna una llista com a argument `plot()` la interpreta com una llista de valors de `y` i realitza la gràfica assumint valors de `x` equiespaiats `x = [0,1,2,...]`

Opcions de `plot()`

La funció `plot()` permet configurar la forma com es representen les llistes de valors mitjançant arguments suplementaris.

La forma abreviada de fer-ho és una cadena de dos caràcters:

- El primer caràcter és una lletra indicant el color de traçat (noteu la correspondència entre el caràcter i el nom del color en anglès):

caràcter	**color**
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

- Després es poden afegir un o dos caràcters indicnt el tipus de traçat. Per exemple `'-'` indica una línia contínua, `'--'` una línia discontínua, `'o'` no s'uneixen els punts amb línies sinó que es marquen amb un símbol rodó, etc. Es pot trobar la llista completa d'opcions a la documentació de la funció plot (http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot).

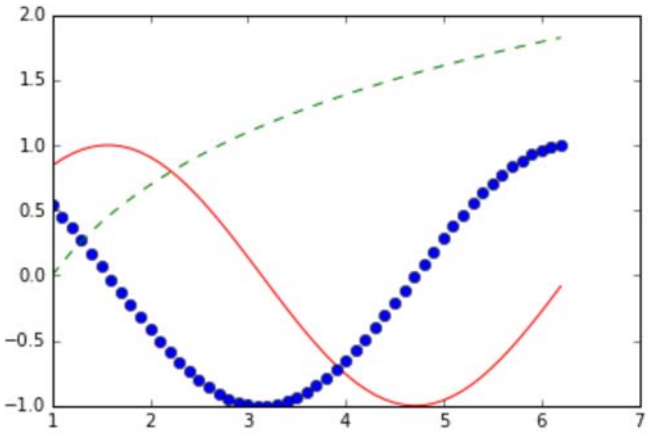
En l'exemple següent usem aquesta opció per a representar tres conjunts de punts en una sola gráfica. Noteu que es poden traçar diverses llistes en una única crida a la funció `plot()`.

In [3]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (1,2*pi) i a partir d'ells tres funcions
x = np.arange(1., 2*np.pi, 0.1);
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.log(x)

# Representem els punts
plt.plot(x, y1, "r-")
plt.plot(x, y2, "bo", x, y3, "g--")
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x3881f90>,
 <matplotlib.lines.Line2D at 0x38fc390>]
```



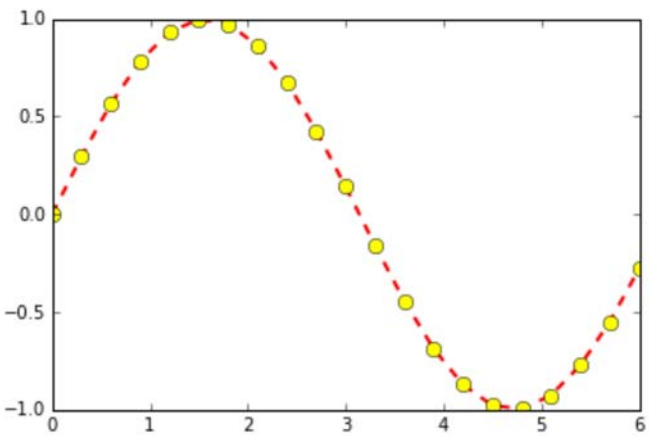
La forma expandida d'especificar l'aparença del traçat és especificar cada característica per separat. En l'exemple següent es poden veure algunes opcions i es pot trobar la llista completa d'opcions a la documentació de la funció plot (http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot).

In [9]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (0,2*pi) i a partir d'ells valors de sin(x)
x = np.arange(0., 2*np.pi, 0.3);
y1 = np.sin(x)

# Representem els punts
plt.plot(x, y1, color='red', linestyle='dashed', marker='o',
         markerfacecolor='yellow', markersize=8, linewidth=2)
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x63529b0>]
```



Control dels eixos

Es pot controlar l'aparença dels eixos amb les funcions:

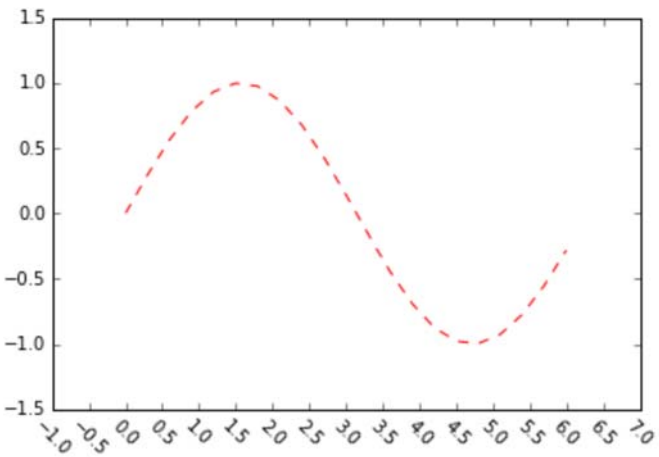
- `xlim(a,b)` límits de l'eix `x`
- `ylim(c,d)` límits de l'eix `y`
- `axis(a,b,c,d)` requadre amb els límits dels eixos
- `xticks()` marcadors de l'eix x
- `yticks()` marcadors de l'eix y
- `xscale()` escala de l'eix x (lineal, logarítmica)
- `yscale()` escala de l'eix y (lineal, logarítmica)

In [6]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (0,2*pi) i a partir d'ells valors de sin(x)
x = np.arange(0., 2*np.pi, 0.3);
y1 = np.sin(x)

# Representem els punts
plt.plot(x, y1, color='red', linestyle='dashed')
plt.xlim(-1,7)
plt.ylim(-1.5,1.5)
plt.xticks(np.arange(-1, 7.5, 0.5), rotation = -45)
```

Out[6]:
```
([<matplotlib.axis.XTick at 0x391da50>,
  <matplotlib.axis.XTick at 0x39e00b0>,
  <matplotlib.axis.XTick at 0x39f7090>,
  <matplotlib.axis.XTick at 0x3a19190>,
  <matplotlib.axis.XTick at 0x3a19750>,
  <matplotlib.axis.XTick at 0x3a19d10>,
  <matplotlib.axis.XTick at 0x3a1c2f0>,
  <matplotlib.axis.XTick at 0x3a1c8b0>,
  <matplotlib.axis.XTick at 0x3a1ce70>,
  <matplotlib.axis.XTick at 0x4aa1450>,
  <matplotlib.axis.XTick at 0x4aa1a10>,
  <matplotlib.axis.XTick at 0x4aa1fd0>,
  <matplotlib.axis.XTick at 0x4aa45b0>,
  <matplotlib.axis.XTick at 0x4aa4b70>,
  <matplotlib.axis.XTick at 0x4aa8150>,
  <matplotlib.axis.XTick at 0x4aa8710>,
  <matplotlib.axis.XTick at 0x4aa8cd0>],
 <a list of 17 Text xticklabel objects>)
```



Gràfics múltiples

Es poden crear figures amb diversos gràfics usant la funció `subplot()`. Aquesta funció reb tres arguments: `subplot(nFiles,nCols,numPlot)`:

- La figura es divideix conceptualment en una quadrícula de *nFiles* × *nCols*
- El número *numPlot* indica en quina posició de la quadrícula ens situem per treballar
- A partir de la crida les funcions gràfiques s'aplicaran al la posició de la quadrícula indicada

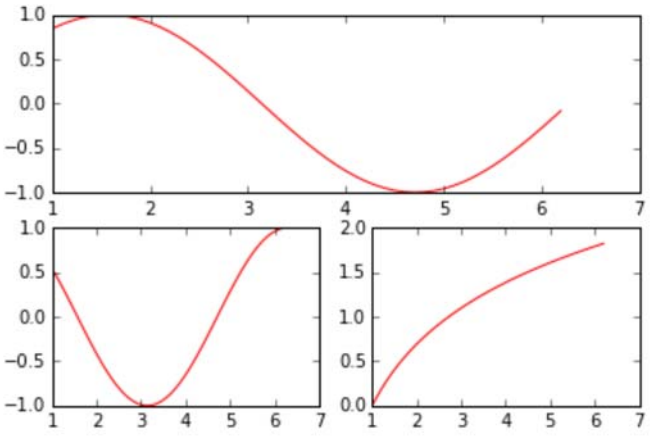
En l'exemple següent fem una figura amb dues gràfiques:

In [7]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (1,2*pi) i a partir d'ells tres funcions
x = np.arange(1., 2*np.pi, 0.1);
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.log(x)

# Representem els arrays anteriors en tres gràfiques
plt.subplot(2,1,1)
plt.plot(x, y1, "r-")
plt.subplot(2,2,3)
plt.plot(x, y2, "r-")
plt.subplot(2,2,4)
plt.plot(x, y3, "r-")
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x4b06690>]
```



Les crides a `subplot()` es poden fer de forma abreviada:

```
subplot(2,2,1) = subplot(221)
```

Si realiteu crides successives amb valors diferents dels arguments *nFiles*,*nCols* les gràfiques es posicionaran el la figura de la forma corresponent i poden sobreescriure gràfiques anteriors.

Text

Les següent funcions permeten afegir text a les gràfiques:

- `xlabel()` etiqueta de l'eix `x`
- `ylabel()` etiqueta de l'eix `y`
- `title()` títol del gràfic

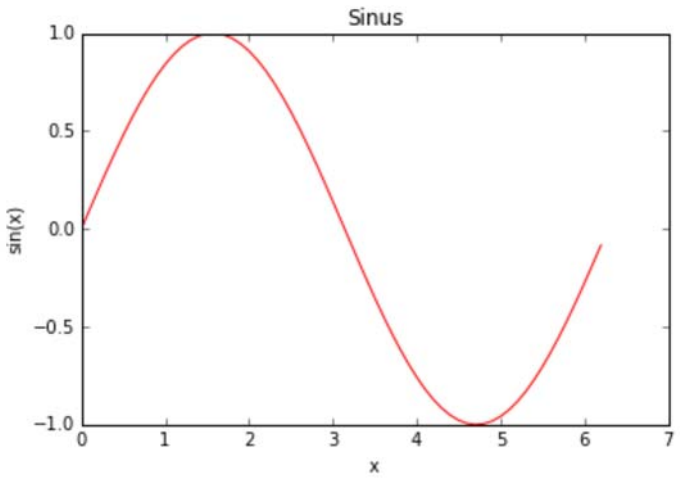
In [8]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (0,2*pi) i a partir d'ells valors de sin(x)
x = np.arange(0., 2*np.pi, 0.1);
y1 = np.sin(x)

# Representem els punts
plt.plot(x, y1, color='red')

# Afegim text
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.title("Sinus")
```

Out[8]: `<matplotlib.text.Text at 0x4b477b0>`



Més en general, la funció `text(x,y,str)` permet escriure un text en qualsevol posició del gràfic, on `x,y`) són les coordenades on posicionar-lo a la gràfica i *str* la cadena amb el text a incloure.

Nota: per a les quatre funcions de text

- Si doneu text entre "$" s'interpretarà com una fórmula de LaTeX
- Podeu canviar la mida i color del text usant com a arguments `, fontsize=NN, color='xxxx'`

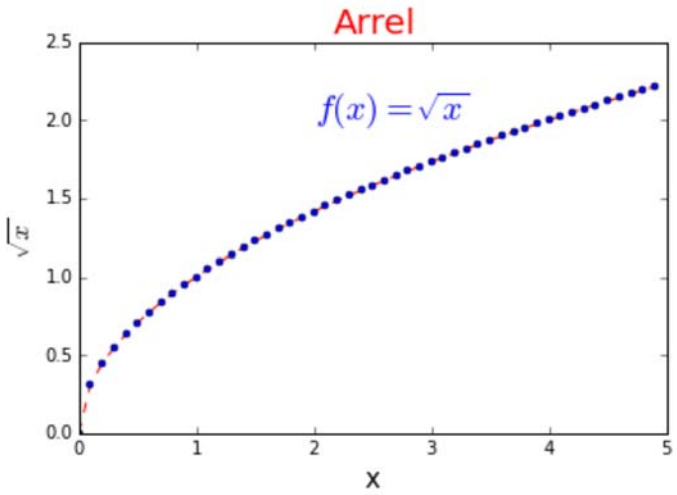
In [10]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (0,5) i a partir d'ells valors de sqrt(x)
x = np.arange(0., 5, 0.1);
y1 = np.sqrt(x)

# Representem els punts
plt.plot(x, y1, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=4)

# Afegim text
plt.xlabel("x",fontsize=15)
plt.ylabel("$\sqrt{x}$",fontsize=15)
plt.title("Arrel", fontsize=20, color='red')
plt.text(2,2,"$f(x)=\sqrt{x}$", fontsize=20, color='blue')
```

Out[10]:
```
<matplotlib.text.Text at 0x4bb64f0>
```



Finalment, la funció `annotate()` crea un text amb una fletxa per crear una anotació senyalant un punt de la gràfica. Reb com a arguments:

- El text a mostrar
- `xy=(a,b)` les coordenades de la punta de la fletxa
- `xytext=(c,d)` la posició del text
- `arrowprops=dict(facecolor='xxx', ...)` les característiques de la fletxa

Podeu veure l'ajuda de annotate() (http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.annotate) per a més detalls.

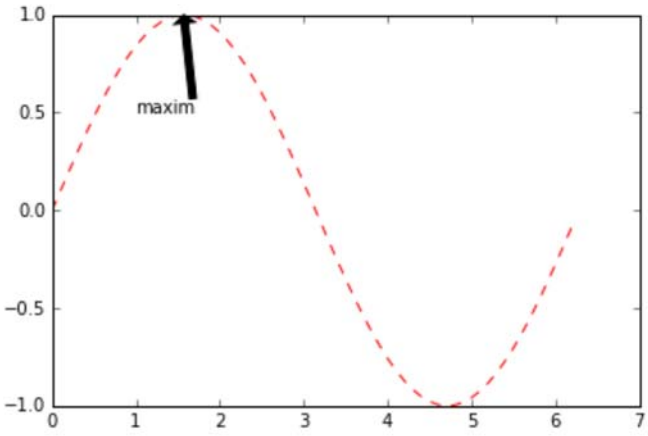
In [12]:
```
%matplotlib inline
import math
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (0,2*pi) i a partir d'ells valors de sin(x)
x = np.arange(0., 2*np.pi, 0.1);
y1 = np.sin(x)

# Representem els punts
plt.plot(x, y1, color='red', linestyle='dashed')

# Marquem el màxim
plt.annotate("maxim", xy=(math.pi/2.,1), xytext=(1,0.5), arrowprops=dict(fac
ecolor='black'))
```

Out[12]:
```
<matplotlib.text.Annotation at 0x5652130>
```



Guardant les figures en un fitxer

Podeu usar la comanda `savefig()` per a guardar la figura en un fitxer. Pren com a arguments:

- Nom del fitxer
- Format del fitxer (pdf, jpg, ps, ...)
- Resolució, expressada en *punts per polzada* (*dots per inch* = *dpi*)

Per exemple:

```
plt.savefig('nom_fitxer', format='jpg', dpi=900)
```

In [13]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Calculem valors de x en (0,5) i a partir d'ells valors de sin(x)
x = np.arange(0, 5, 0.1);
y = np.sin(x)

# Representem els punts
plt.plot(x, y)
plt.savefig('sinus.jpg', format='jpg', dpi=300)
```