INDICE

Lezione basata sul seguente tutorial (parecchio ben fatto):

http://scipy-lectures.github.com/intro/matplotlib/matplotlib.html

Simple plot

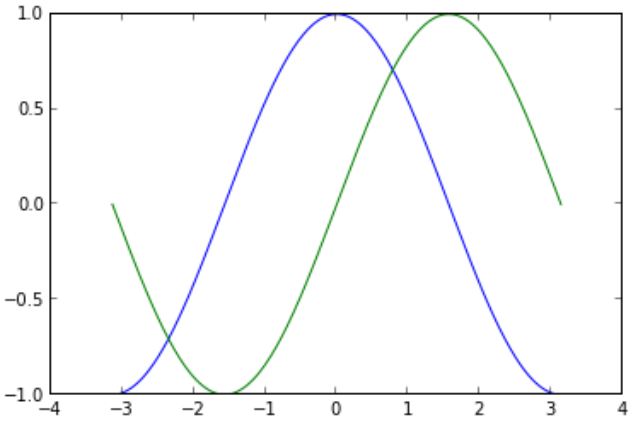
In questa sezione, disegneremo una figura che rappresenta l'andamento di seno e coseno tra -pigreco e pigreco.
Partiremo dalle impostazioni predefinite, ed arricchiremo la figura passo passo per migliorarla.

primo passo, generiamo i dati da rappresentare

In [6]:
```
import numpy as np
X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)
#print(X)
```

uso le impostazioni predefinite

creo questo grafico



- pyplot tutorial
- comando plot

Useremo il modulo pyplot che mette a disposizione una interfaccia alle matplotlib che ricalca quella di matlab.

In [7]:
```
#import matplotlib
import matplotlib.pyplot as pl
pl.plot(X, C)
pl.plot(X, S)
pl.savefig("./img/matplotlib00.png", dpi=72)
pl.show()
```

Eseguiire lo script /work/matplotlib_00.py e commentare cosa succede:

- l'istruzione plt.show() fa comparire una finestra che mette a disposizione alcune funzionalità interattive (pan, zoom, salvataggio, ridimensionamento assi)
- l'interprete rimane 'congelato fino a che la finestra interattiva è chiusa.
- In questo tutorial lavoreremo in questa modalità cioè prima creeremo con degli script tutti gli elementi del grafico e una volta finito li visualizzeremo in una finestra interattiva (e/o li salveremo come immagine).
- E' anche possibile aggiornare interattivamente, via codice, una figura, facendo in modo che l'interprete non si blocchi quando la figura è visualizzata.

Impostazione esplicita dei defaults

Nell'esempio sopra abbiamo usato molte impostazioni di default delle matplotlib. Nel seguente ricreeremo un grafico identico settando esplicitamente le varie impostazioni.

In [8]:
```
#creo una figura di 8x6 pollici con risoluzione di 80 punti/pollice
pl.figure(figsize=(8,6),dpi=72)

# creo un unico sistema di assi cartesiani nella figura
pl.subplot(1, 1, 1) #(1riga,1colonna,grafico1)

# Plot cosine with a blue continuous line of width 1 (pixels)
pl.plot(X, C, color="blue", linewidth=1.0, linestyle="-")

# Plot sine with a green continuous line of width 1 (pixels)
pl.plot(X, S, color="green", linewidth=1.0, linestyle="-")

# Set x limits
pl.xlim(-4.0, 4.0)

# Set x ticks
pl.xticks(np.linspace(-4, 4, 9, endpoint=True))

# Set y limits
```

```
pl.ylim(-1.0, 1.0)

# Set y ticks
pl.yticks(np.linspace(-1, 1, 5, endpoint=True))

# Save figure using 72 dots per inch
pl.savefig("./img/matplotlib01.png", dpi=72)

# Show result on screen
pl.show()
```

note:

- non c'è alcun legame 'formale' tra un comando ed il successivo.
- ogni comando agisce sulla figura corrente (attiva)

Per approfondimenti:

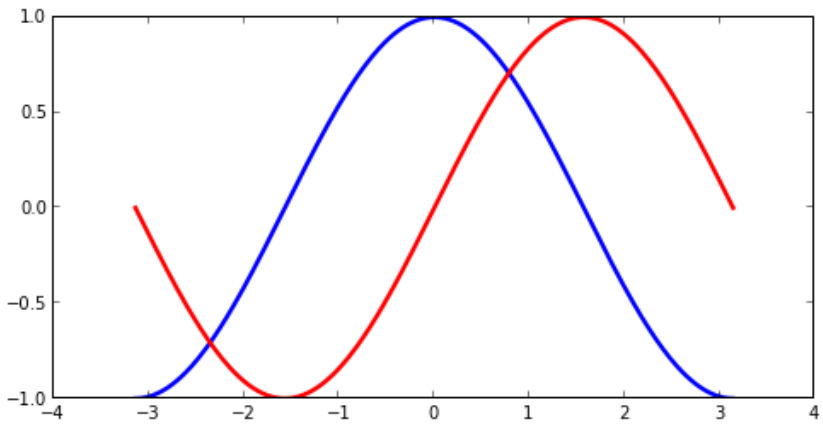
- Customizzare i default di matplotlib

Riferimenti alle funzioni usate:

- pyplot.figure(): crea una nuova figura.
- pyplot.subplot(): crea un sistema di assi cartesiane.
- pyplot.plot(): crea le curve.
- pyplot.xlim(): imposta i limiti dell'asse x.
- pyplot.xticks(): imposta le etichette dell'asse x.
- pyplot.savefig(): salva la figura come png.
- pyplot.show(): mostra la figura sullo schermo.

Modifico colori e spessori

creo questo grafico



- stiro la figura orizzontalmente
- cambio i colori delle curve
- aumento lo spessore delle linee

In [9]:
```
pl.figure(figsize=(8, 4))
pl.plot(X, C, color="blue", linewidth=2.5, linestyle="-")

#le seguenti istruzioni sono equivalenti (vedi http://matplotlib.org/api/colors_api.html)
pl.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
#pl.plot(X, S, color="#ff0000",  linewidth=2.5, linestyle="-")
#pl.plot(X, S, color=(1.0,0.0,0.0),  linewidth=2.5, linestyle="-")
pl.savefig("./img/matplotlib02.png")
```

- Proprietà delle linee 2D
- Controllare le proprietà delle linee 2D
- Possibili stili di linea
- Possibili marker
- Colori in matplotlib.

Come sopra ma orientato agli oggetti

In [10]:
```
pl.figure(figsize=(8, 4))

#recuper il riferimento al grafico della figura (in questo caso l'unico, ma ce ne potrebbe
essere più di uno)
ax=pl.subplot(111)

#conservo i riferimenti ad ogni linea creata (plot restituisce una sequenza di linee quindi
metto la virgola dopo il nome della linea...)
lineacos, = ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
lineasen, = ax.plot(X, S)#, color="red",  linewidth=2.5, linestyle="-")


#imposto le proprietà della linea coseno
#usando i metodi dell'oggetto linea
lineacos.set_color('b')
lineacos.set_linestyle('-')
lineacos.set_linewidth(3.5)

#imposto le proprietà della linea seno
#usando la funzione matplotlib.artist.setp e riferendomi all'oggetto linea
matplotlib.artist.setp(lineasen,color="red",  linewidth=2.5, linestyle="-")

plt.show()
```

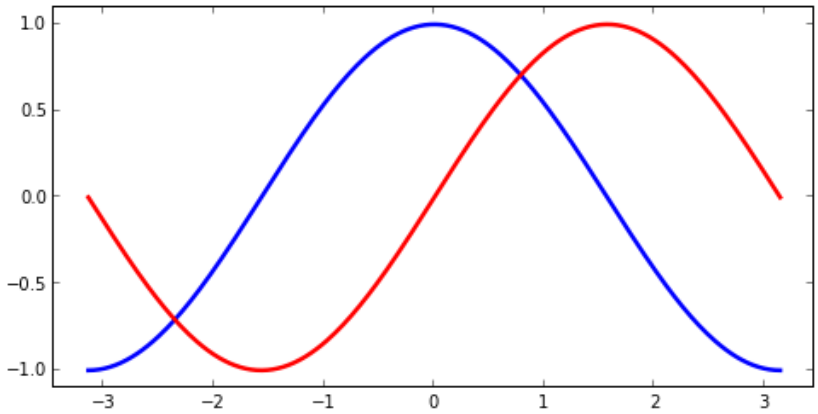
In [11]:
```
matplotlib.artist.getp(lineasen)
```

```
    agg_filter = None
    alpha = None
    animated = False
    antialiased or aa = True
    axes = Axes(0.125,0.125;0.775x0.775)
    children = []
    clip_box = TransformedBbox(Bbox('array([[ 0.,  0.],\n       [...
    clip_on = True
    clip_path = None
    color or c = red
    contains = None
    dash_capstyle = butt
    dash_joinstyle = round
    data = (array([-3.14159265, -3.11695271, -3.09231277, -3....
    drawstyle = default
    figure = Figure(640x320)
    fillstyle = full
    gid = None
    label = _line1
    linestyle or ls = -
    linewidth or lw = 2.5
    marker = None
    markeredgecolor or mec = red
    markeredgewidth or mew = 0.5
    markerfacecolor or mfc = red
    markerfacecoloralt or mfcalt = none
    markersize or ms = 6
    markevery = None
    path = Path([[ -3.14159265e+00  -1.22464680e-16]  [ -3.11...
```

```
picker = None
pickradius = 5
rasterized = None
snap = None
solid_capstyle = projecting
solid_joinstyle = round
transform = CompositeGenericTransform(TransformWrapper(Blended...
transformed_clip_path_and_affine = (None, None)
url = None
visible = True
xdata = [-3.14159265 -3.11695271 -3.09231277 -3.06767283 -...
xydata = [[ -3.14159265e+00  -1.22464680e-16]  [ -3.1169527...
ydata = [ -1.22464680e-16  -2.46374492e-02  -4.92599411e-0...
zorder = 2
```

Cambio i limiti degli assi

creo questo grafico



imposto i limiti degli assi in modo che si adattino ai dati visualizzati con

- pyplot.xlim: Get or set the x limits of the current axes

In [12]:
```
pl.figure(figsize=(8, 4), dpi=80)
pl.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
pl.plot(X, S, color="red",  linewidth=2.5, linestyle="-")

######################-INIZIO-MODIFICHE-##############################
#cambio i limiti degli assi
pl.xlim(X.min() * 1.1, X.max() * 1.1)
pl.ylim(C.min() * 1.1, C.max() * 1.1)

#EQUIVALENTE
#pl.axis(X.min()*1.1 , X.max()*1.1 , pl.ylim(C.min()*1.1 , C.max()*1.1)

######################-FINE-MODIFICHE-################################
pl.savefig("./img/matplotlib03.png")
pl.show()
```

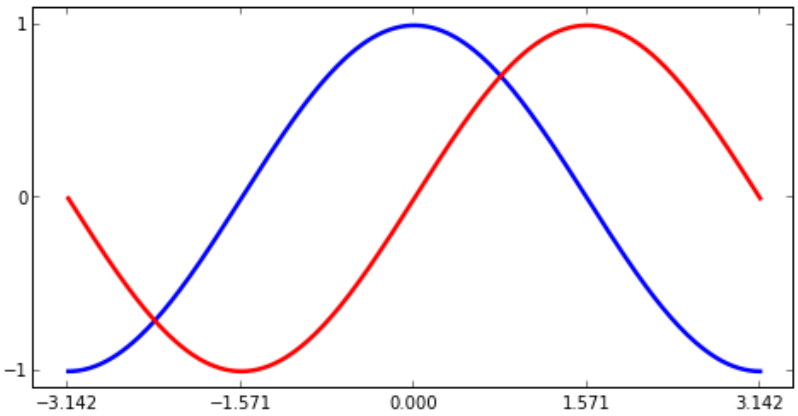
come sopra ma orientato agli oggetti

In [13]:
```
f = pl.figure(figsize=(8, 4))
ax = plt.subplot(111)
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-")

######################-INIZIO-MODIFICHE-##############################
```

```
#cambio i limiti degli assi
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
######################-FINE-MODIFICHE-################################
pl.show()
```

Imposto i marcatori e le etichette degli assi

creo questo grafico



- pyplot.xticks: Get or set the the current x tick locations and labels.
- matplotlib.axis.Axis :handle the drawing of the tick lines, the grid lines, the tick labels and the axis label.
- modulo ticker: this module contains classes to support completely configurable tick locating and formatting.

cambio la posizione dei marcatori degli assi (e conseguentemente le etichette)

In [14]:
```
pl.figure(figsize=(8, 4))
pl.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
pl.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
pl.xlim(X.min() * 1.1, X.max() * 1.1)
pl.ylim(C.min() * 1.1, C.max() * 1.1)

######################-INIZIO-MODIFICHE-################################

#cambio la posizione dei marcatori degli assi (e conseguentemente le etichette)
#impostando esplicitamente solo la posizione dei marcatori
print('vecchie ticks:' , pl.xticks())
print(type(pl.xticks()[0]))##NB: array numpy

pl.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
pl.yticks([-1, 0, +1])

#########################################################################
pl.savefig("./img/matplotlib04.png")
pl.show()
```

```
vecchie ticks: (array([-4., -3., -2., -1.,  0.,  1.,  2.,  3.,  4.]), <a list of 9 Text xticklabel
objects>)
<class 'numpy.ndarray'>
```

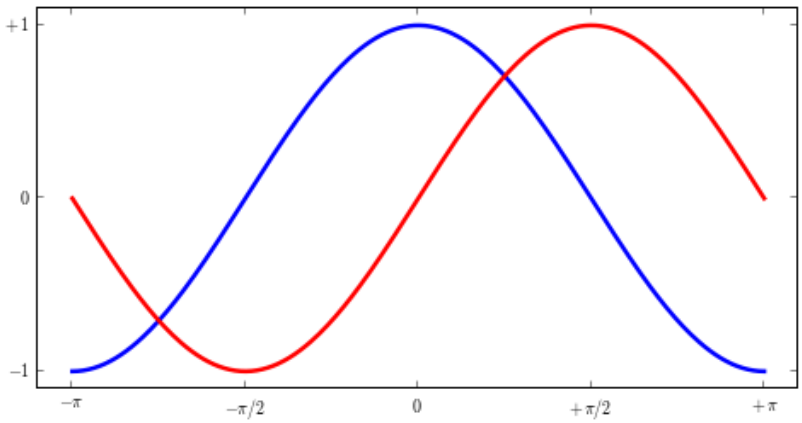
come sopra ma orientato agli oggetti

In [15]:
```
f = pl.figure(figsize=(8, 4))
ax = plt.subplot(111)
```

```
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-")

######################-INIZIO-MODIFICHE-#################################

#cambio i limiti degli assi
#i membri xaxis e yaxis di ogni axes permettono di gestire i marcatori e le etichette di
ciascun asse
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.yaxis.set_ticks([-1, 0, +1])

#ax.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])

######################-FINE-MODIFICHE-###################################
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
pl.show()
```

Modifico il testo delle etichette dei marcatori degli assi

creo questo grafico



- Le etichette sono piazzate correttamente ma adesso vogliamo riportare π invece di 3.142.
- Usiamo xticks() e yticks() specificando la lista delle stringhe da visualizzare come etichetta come lista (secondo parametro).
- Usiamo LateX per rappresentare le etichette (testo delimitato dal carattere $).

In [16]:
```
pl.figure(figsize=(8, 4), dpi=80)
pl.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
pl.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
pl.xlim(X.min() * 1.1, X.max() * 1.1)
pl.ylim(C.min() * 1.1, C.max() * 1.1)
######################-INIZIO-MODIFICHE-#################################

#imposto sia la posizione dei marcatori sia le corrispondenti etichette
pl.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

#visualizzo il segno a +1
pl.yticks([-1, 0, +1],
          ['$-1$', '$0$', r'$+1$'])

########################################################################
pl.savefig("./img/matplotlib05.png")
pl.show()
```

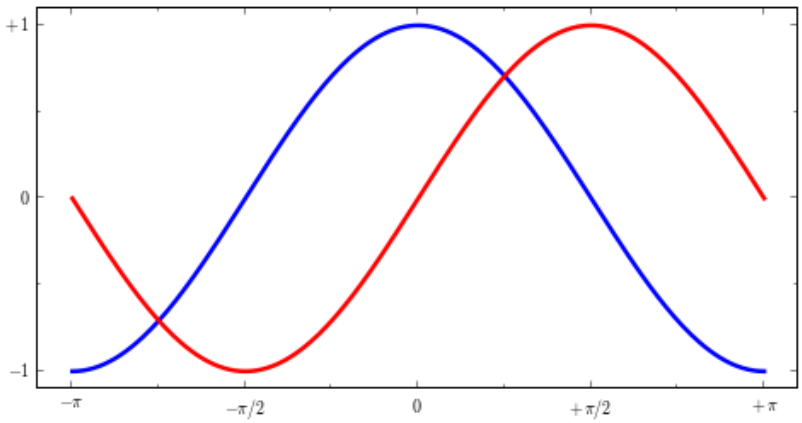
- Working with text
- xticks(): : Get or set the the current x tick locations and labels.
- yticks(): : Get or set the the current y tick locations and labels.
- LateX Math Symbols

come sopra ma orientato agli oggetti

In [17]:
```
f = pl.figure(figsize=(8, 4))
ax = plt.subplot(111)
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
######################-INIZIO-MODIFICHE-##############################
#cambio i limiti degli assi
#i membri xaxis e yaxis di ogni axes permettono di gestire i marcatori e le etichette di
ciascun asse
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

ax.yaxis.set_ticks([-1, 0, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$0$', r'$+1$'])
######################-FINE-MODIFICHE-################################

pl.show()
```

Aggiungo dei marcatori minori agli assi

creo questo grafico



d'ora in poi continuo solo con approccio orientato agli oggetti

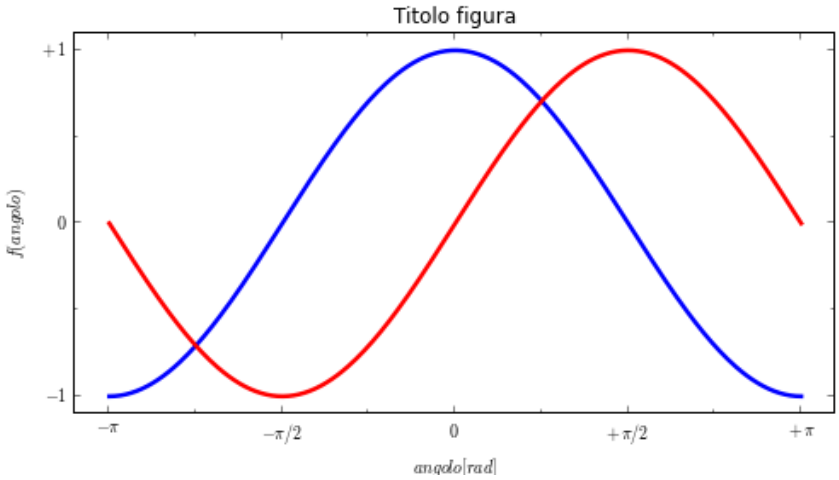
Nell esempio sotto uso un approccio di base, per un maggiore controllo usare i tick-locators (vedi anche questo esempio)

In [18]:
```
f = pl.figure(figsize=(8, 4))
ax = plt.subplot(111)
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
#cambio i limiti degli assi
#i membri xaxis e yaxis di ogni axes permettono di gestire i marcatori e le etichette di
ciascun asse
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, 0, +1])
```

```
ax.yaxis.set_ticklabels([r'$-1$', r'$0$', r'$+1$'])
#####################-INIZIO-MODIFICHE-##############################
# NB:
# - di defaul le etichette dei marcatori minori sono spente
# - in xaxis.set_ticks minor=False di default
ax.xaxis.set_ticks([-3*np.pi/4, -np.pi/4, np.pi/4, 3*np.pi/4] , minor=True)
ax.yaxis.set_ticks([-0.5, 0.5] , minor=True)

#####################-FINE-MODIFICHE-################################
pl.savefig("./img/matplotlib06.png")
pl.show()
```

Aggiungo un titolo alla figura ed una etichetta agli assi

creo questo grafico



In [19]:
```
f = pl.figure(figsize=(8, 4))
ax = plt.subplot(111)
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, 0, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$0$', r'$+1$'])
ax.xaxis.set_ticks([-3*np.pi/4, -np.pi/4, np.pi/4, 3*np.pi/4] , minor=True)
ax.yaxis.set_ticks([-0.5, 0.5] , minor=True)
#####################-INIZIO-MODIFICHE-##############################
ax.set_title('Titolo figura')

ax.set_xlabel('$angolo [rad]$')
#ax.xaxis.set_label_text('$angolo [rad]$')


ax.set_ylabel('$f(angolo)$')
#ax.yaxis.set_label_text('$angolo [rad]$')
#####################-FINE-MODIFICHE-################################
pl.savefig("./img/matplotlib07.png")
pl.show()
```

Sposto gli assi al centro della figura

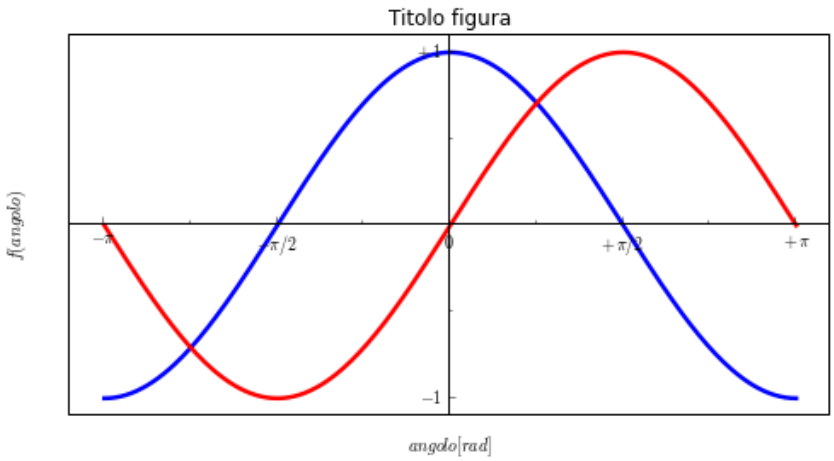
- Spines
- Axis container
- Transformations tutorial

- di default ci sono quattro 'spines' per ogni axes, due orizzontali e due verticali, piazzate ai margini dell'axes' (cioé del sistema di assi cartesiane corrente per la figura, come vedremo ce ne possono essere più di uno)
- di default i marcatori (ticks) e le etichette sono associate alle 'spines' in basso ('bottom') e a sinistra ('left')
- le 'spines' in alto ('top') e a destra ('right') hanno i marcatori ma non le etichette
- ogni axes 'incapsula' due oggetti Axis cioé xaxis e yaxis che permettono di gestire i marcatori e le etichette su ciascun asse cartesiano. Come detto sopra i marcatori sono associati ad entrambe le spines di riferimento (top e bottom per xaxis)

Nel seguente esempio:

- cancelliamo le 'spines' in alto e a destra assegnandogli il colore 'none' con set_color
- di default i marcatori (ticks) e le etichette sono associate alle 'spines' in basso ('bottom') e a sinistra ('left')
- le 'spines' in alto ('top') e a destra ('right') hanno i marcatori ma non le etichette
- faccio in modo che i marcatori (ticks) siano presenti solo sull'asse x 'bottom' con set_label_position
- faccio in modo che i marcatori (ticks) siano presenti solo sull'asse y 'left' con set_label_position
- imposto la posizione di ciascun asse con set_position in modo che si incrocino nel punto (0,0) in coordinate dati.

creo questo grafico



Versione di base

In [20]:
```
ax=pl.subplot(1,1,1)
ax.plot(X, C,label='coseno')
ax.plot(X, S,label='seno')
##########################################
#cancello le 'spines' in alto e a destra
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
#imposto la posizione di degli altri due in modo che si incrocino nell'origine
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0,))
##########################################
pl.show()
```

come si vede sono rimasti i marcatori delle due spines cancellate. Correggeremo questo problema nell'esempio sotto

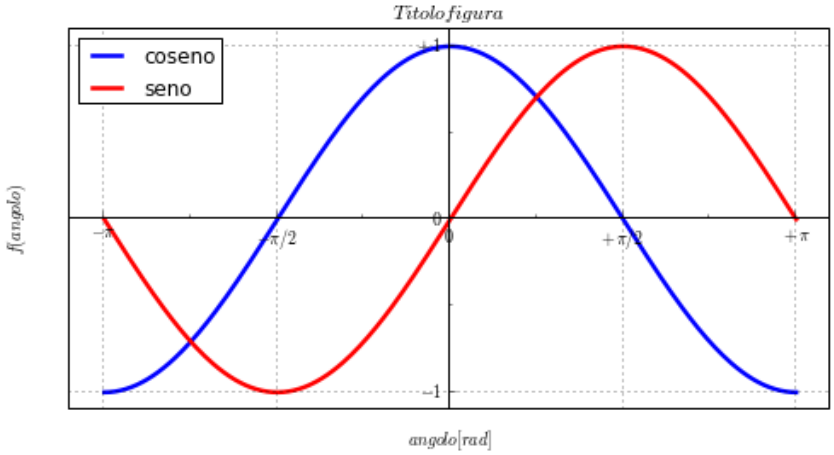
Versione che riprende l'esempio del tutorial

In [21]:
```
pl.figure(figsize=(8, 4))
ax=pl.subplot(111)
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
```

```
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
ax.set_title('Titolo figura')
#####################-INIZIO-MODIFICHE-##############################
#recupero l'axis corrente che mi permette di accedere
# - a ciascuno dei quattro oggetti spines (sta per spina dorsale, cioè l'asse vero e proprio)
# - agli oggetti xaxis e yaxis che permettono di gestire marcatori ed etichette su ciascun asse
#(approccio orientato agli oggetti)

#cancello le 'spines' in alto e a destra
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
#imposto la posizione di degli altri due in modo che si incrocino nell'origine
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))

#CANCELLO I MARCATORI DALLE SPINES CANCELLATE
#faccio in modo che i marcatori (ticks) siano presenti solo sull'asse x 'bottom'
ax.xaxis.set_ticks_position('bottom')#both di default, vedi figura sopra
#faccio in modo che i marcatori (ticks) siano presenti solo sull'asse y 'left'
ax.yaxis.set_ticks_position('left')#both di default, vedi figura sopra


# rimpiazzo il bordo della figura
# rendendo visibile il bordo del rettangolo
# che delimita l'area dati
ax.patch.set_edgecolor('k')
ax.patch.set_linewidth(1)
#ax.patch.set_color('yellow')
#ax.patch.set_alpha(0.4)


#SPOSTO LE ETICHETTE DELL'ASSE!!!!!!!!
ax.xaxis.set_label_text('$angolo [rad]$')
ax.xaxis.set_label_coords( x = 0.5 , y = -0.05 , transform = ax.transAxes)

ax.yaxis.set_label_text('$f(angolo)$')
ax.yaxis.set_label_coords( x = -0.05 , y = 0.5)

#####################-FINE-MODIFICHE-##############################
#da mettere dopo lo spostamento delle spines altrimenti non hanno effetto!!!!!
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$+1$'])
ax.xaxis.set_ticks([-3*np.pi/4, -np.pi/4, np.pi/4, 3*np.pi/4] , minor=True)
ax.yaxis.set_ticks([-0.5, 0.5] , minor=True)


pl.savefig("./img/matplotlib08.png")
pl.show()
```

Aggiungo una legenda ed una griglia

creo questo grafico

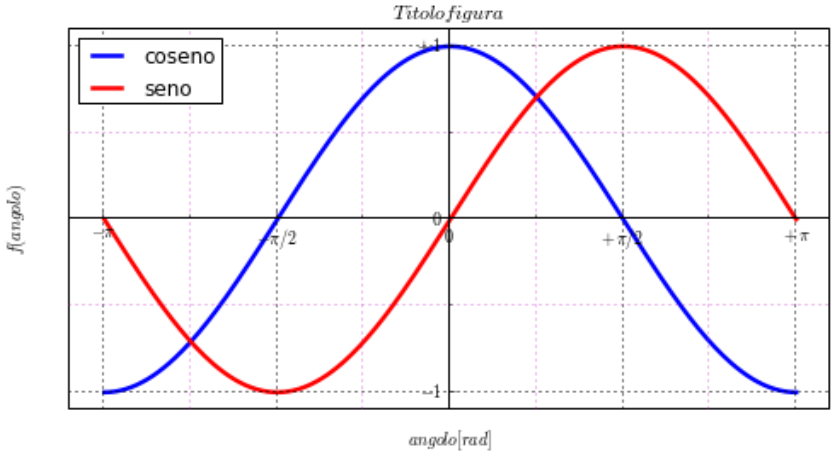


- Legend guide
- pyplot.legend()
- Legend API
- pyplot.grid()

In [22]:
```
fig=pl.figure(figsize=(8, 4))
ax = pl.subplot(111)
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))
ax.patch.set_edgecolor('k')
ax.patch.set_linewidth(1)
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, 0, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$0$', r'$+1$'])
ax.xaxis.set_ticks([-3*np.pi/4, -np.pi/4, np.pi/4, 3*np.pi/4] , minor=True)
ax.yaxis.set_ticks([-0.5, 0.5] , minor=True)
ax.xaxis.set_label_text('$angolo [rad]$')
ax.xaxis.set_label_coords( x = 0.5 , y = -0.05)
ax.yaxis.set_label_text('$f(angolo)$')
ax.yaxis.set_label_coords( x = -0.05 , y = 0.5)
ax.set_title('$Titolo figura$')
##########################-INIZIO-MODIFICHE-##############################

#aggiungo una etichetta ad ogni linea
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="coseno")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="seno")
#creo una legenda in alto a sinistra che farà riferimento alle etichette di cui sopra
ax.legend(loc='best')#
#aggiungo la griglia in corrispondenza agli x tick definiti
ax.grid()

###############################################################################
pl.savefig("./img/matplotlib09.png")
pl.show()
```

creo una griglia per i ticks secondari

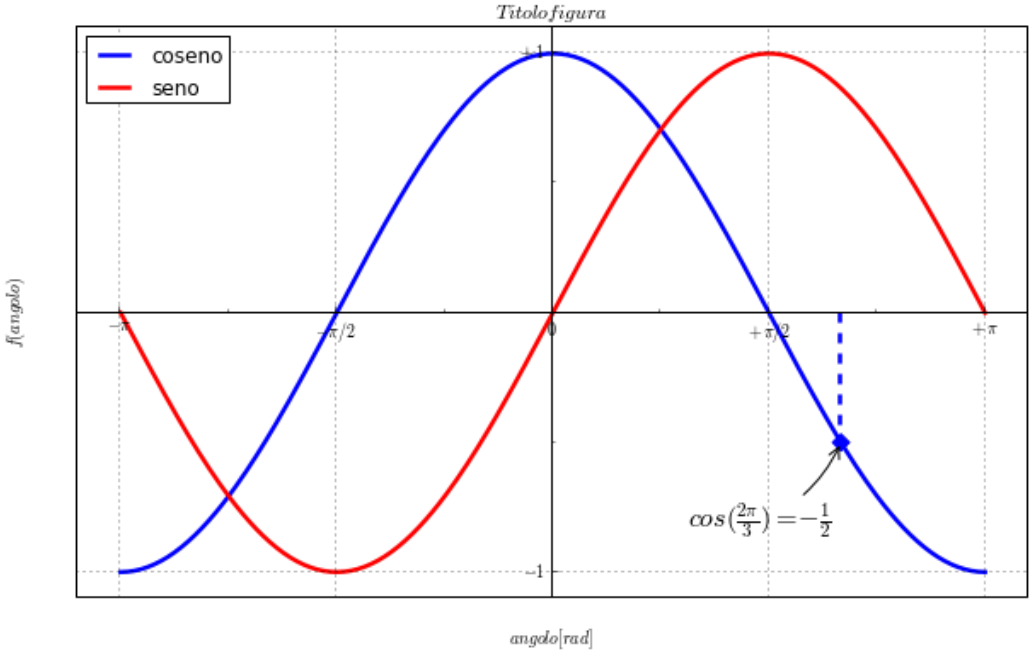
creo questo grafico



In [23]:
```
fig=pl.figure(figsize=(8, 4))
ax = pl.subplot(111)
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))
ax.patch.set_edgecolor('k')
ax.patch.set_linewidth(1)
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, 0, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$0$', r'$+1$'])
ax.xaxis.set_ticks([-3*np.pi/4, -np.pi/4, np.pi/4, 3*np.pi/4] , minor=True)
ax.yaxis.set_ticks([-0.5, 0.5] , minor=True)
ax.xaxis.set_label_text('$angolo [rad]$')
ax.xaxis.set_label_coords( x = 0.5 , y = -0.05)
ax.yaxis.set_label_text('$f(angolo)$')
ax.yaxis.set_label_coords( x = -0.05 , y = 0.5)
ax.set_title('$Titolo figura$')
#aggiungo una etichetta ad ogni linea
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="coseno")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="seno")
#creo una legenda in alto a sinistra che farà riferimento alle etichette di cui sopra
ax.legend(loc='upper left')

##########################-INIZIO-MODIFICHE-##############################

ax.xaxis.grid(True,'major',linewidth=1)
ax.yaxis.grid(True,'major',linewidth=1)

ax.xaxis.grid(True,'minor',color='m')
ax.yaxis.grid(True,'minor',color='m')
###############################################################################
pl.savefig("./img/matplotlib09_bis.png")
pl.show()
```

Aggiungo annotazioni

creo questo grafico (ingrandisco un po'la figura)



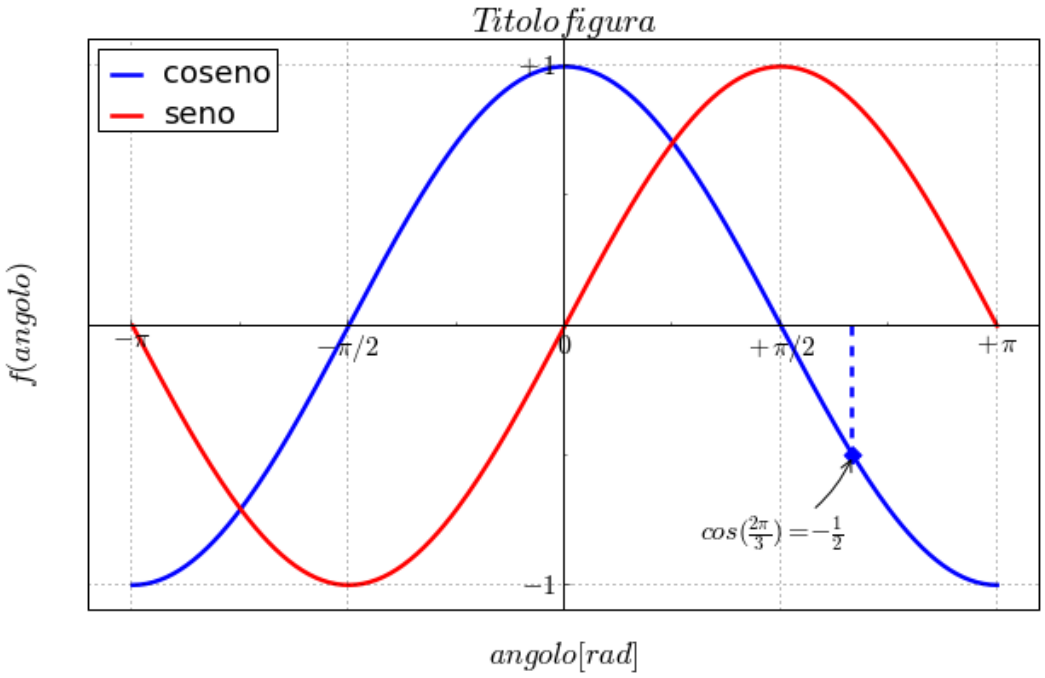
- Guida alle annotazioni
- pyplot.annotate()

In [24]:
```
fig=pl.figure(figsize=(10, 6),dpi = 80)
ax = pl.subplot(111)
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))
ax.patch.set_edgecolor('k')
ax.patch.set_linewidth(1)
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$+1$'])
ax.xaxis.set_ticks([-3*np.pi/4, -np.pi/4, np.pi/4, 3*np.pi/4] , minor=True)
ax.yaxis.set_ticks([-0.5, 0.5] , minor=True)
ax.xaxis.set_label_text('$angolo [rad]$')
ax.xaxis.set_label_coords( x = 0.5 , y = -0.05)
ax.yaxis.set_label_text('$f(angolo)$')
ax.yaxis.set_label_coords( x = -0.05 , y = 0.5)
ax.set_title('$Titolo figura$')

l1,=ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="coseno")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="seno")

ax.grid()
##########################-INIZIO-MODIFICHE-##############################

#ascissa delle linee verticali tratteggiate
t = 2 * np.pi / 3
#linea verticale blu
ax.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyle="--" )

#punto blu
ax.scatter([t ], [np.cos(t) ], 50, color='blue',marker='D')

#testo + freccia che indica il punto blu
```

```
ax.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$',#testo
            xy=(t, np.cos(t)), xycoords='data',   #coordinate e sistema di coordinate del punto 'mirato'
            xytext=(-90, -50), textcoords='offset points',# punto di inserimento del testo, offset in punti rispetto alla punta della freccia
            fontsize=16, # dimensione del testo
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2")) #caratteristiche del connettore

ax.legend(loc='upper left')
#############################################################################
pl.savefig("./img/matplotlib10.png")
pl.show()
```

Aumento le dimensioni del font della legenda delle etichette degli assi e dei ticks e creo intorno a queste ultime una bbox semitrasparente

creo questo grafico



In [25]:
```
fig=pl.figure(figsize=(10, 6), dpi=80)
#recupero l'axis corrente che mi permette di accedere a ciascuna delle quattro spines
#(approccio orientato agli oggetti)
ax = pl.subplot(111)
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))
ax.patch.set_edgecolor('k')
ax.patch.set_linewidth(1)
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, +1])
```

```
ax.yaxis.set_ticklabels([r'$-1$', r'$+1$'])
ax.xaxis.set_ticks([-3*np.pi/4, -np.pi/4, np.pi/4, 3*np.pi/4] , minor=True)
ax.yaxis.set_ticks([-0.5, 0.5] , minor=True)
ax.xaxis.set_label_text('$angolo [rad]$')
ax.xaxis.set_label_coords( x = 0.5 , y = -0.05)
ax.yaxis.set_label_text('$f(angolo)$')
ax.yaxis.set_label_coords( x = -0.05 , y = 0.5)
ax.set_title('$Titolo figura$')
ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="coseno")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="seno")
ax.legend(loc='upper left')
ax.grid()
t = 2 * np.pi / 3
ax.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyle="--")
ax.scatter([t, ], [np.cos(t), ], 50, color='blue',marker='D')
ax.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$',#testo
            xy=(t, np.cos(t)), xycoords='data',   #coordinate e sistema di coordinate del punto
'mirato'
            xytext=(-90, -50), textcoords='offset points',# punto di inserimento del testo,
offset in punti rispetto alla punta della freccia
            fontsize=16, # dimensione del testo
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2")) #caratteristiche
del connettore


##########################-INIZIO-MODIFICHE-##############################
#recupero la legenda, non avevo creato una variabile nel momento in cui la ho creata
leg = ax.get_legend()

#cambio font alla legenda
for label in leg.get_texts():
    label.set_fontsize(18)
#cambio font alle etichette
for label in ax.get_xticklabels() + ax.get_yticklabels():
    label.set_fontsize(16)
    #creo un rettangolo bianco semitrasparente senza bordi
    label.set_bbox(dict(facecolor='none', edgecolor='none', alpha=0.65))
    #posso anche ruotare le etichette ...
    #label.set_rotation(-45)

ax.xaxis.get_label().set_fontsize(18)
ax.yaxis.get_label().set_fontsize(18)
ax.set_title(ax.get_title() , fontsize=20)

#NB: potevo creare labels, etichette e legende direttamente con il font desiderato
# passando fontsize come argomento
# ax.legend(loc='upper left', fontsize=18)
# ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'],fontsize=18)
##########################################################################
pl.savefig("./img/matplotlib11.png")
pl.show()
```

- set_xticklabels():Set the xtick labels with list of strings labels. Return a list of axis text instances.
- set_yticklabels():Set the ytick labels with list of strings labels. Return a list of axis text instances.

Aggiungo un asse y secondario per rappresentare una serie di dati con diversa scala

vedi anche multiple_yaxis_with_spines

definisco una nuova serie di dati con scala diversa

In [26]:
```
Z = 1e-3*(C**3+S**2)
```

Se plotto le tre serie con la stessa scala ....

In [27]:
```
ax=subplot(111)
ax.plot(X, C)
ax.plot(X, S)
ax.plot(X, Z)
pl.show()
```

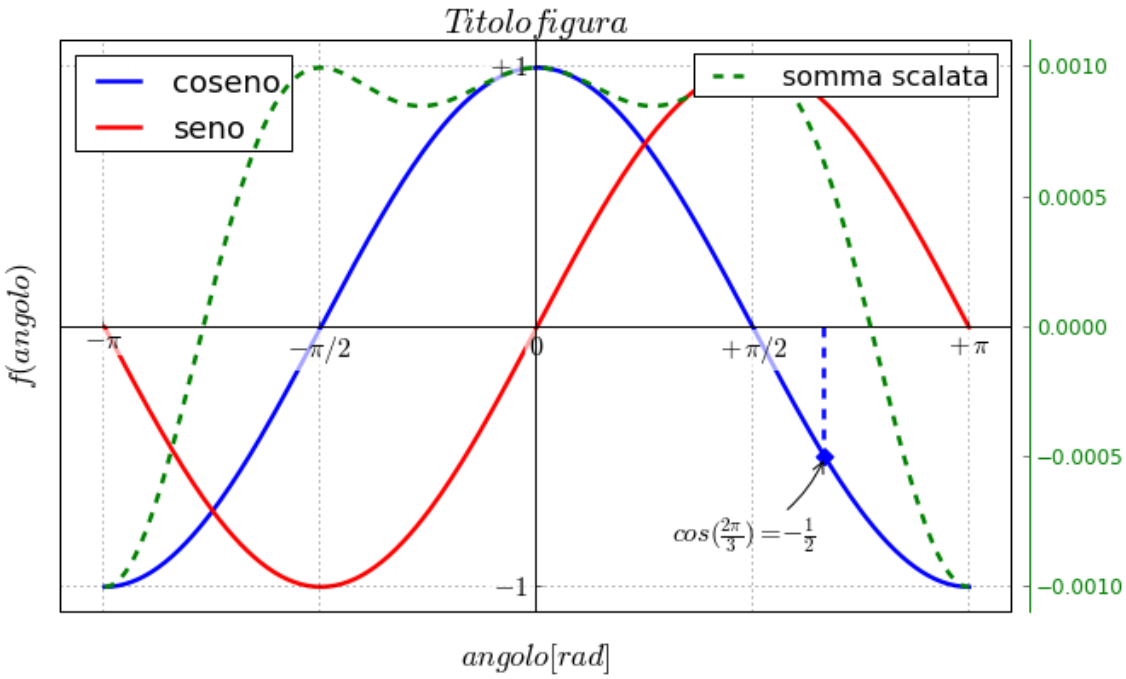
uso quindi una scala secondaria per la serie di dati Z ...

versione di base

In [28]:
```
ax=pl.subplot(1,1,1)
ax.plot(X, C,label='coseno')
ax.plot(X, S,label='seno')

#genero un sistema di assi che condivide l'asse x con ax
a2=ax.twinx()
a2.plot(X, Z,'r--',label='somma scalata')
a2.legend(loc='best')
ax.legend(loc='best')
pl.show()
```

versione che riprende l'esempio precedente

voglio creare questo grafico



In [29]:
```
fig=pl.figure(figsize=(10, 6), dpi=80)
#recupero l'axis corrente che mi permette di accedere a ciascuna delle quattro spines
#(approccio orientato agli oggetti)
ax = pl.subplot(111)
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
```

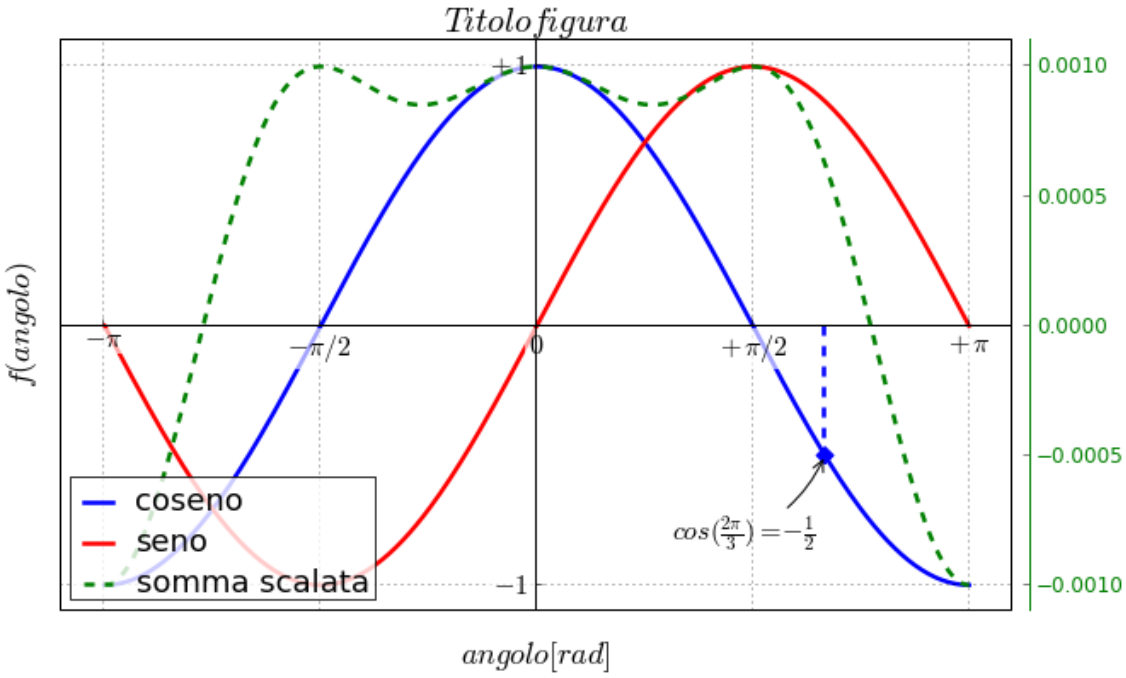
```
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))
ax.patch.set_edgecolor('k')
ax.patch.set_linewidth(1)

ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="coseno")
ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="seno")
ax.legend(loc='upper left', fontsize=18)
ax.grid()
t = 2 * np.pi / 3
ax.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyle="--")
ax.scatter([t, ], [np.cos(t), ], 50, color='blue',marker='D')
ax.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$',#testo
            xy=(t, np.cos(t)), xycoords='data',   #coordinate e sistema di coordinate del punto
'mirato'
            xytext=(-90, -50), textcoords='offset points',# punto di inserimento del testo,
offset in punti rispetto alla punta della freccia
            fontsize=16, # dimensione del testo
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2")) #caratteristiche
del connettore

ax.xaxis.set_label_text('$angolo [rad]$',fontsize=18)
ax.xaxis.set_label_coords( x = 0.5 , y = -0.05)
ax.yaxis.set_label_text('$f(angolo)$',fontsize=18)
ax.yaxis.set_label_coords( x = -0.02 , y = 0.5)
ax.set_title('$Titolo figura$',fontsize=20)


##########################-INIZIO-MODIFICHE-##############################

#genero un sistema di assi che condivide con ax l'asse x e con asse y indipendente
#sa diventa il sistema corrente
sa=ax.twinx()

#vedi http://matplotlib.org/examples/pylab_examples/multiple_yaxis_with_spines.html
#rendo gli assi del sistema di riferimento secondario visibili
sa.set_frame_on(True)
#rendo il riempimento del nuovo sistema di assi invisibile
#altrimenti avrei un rettangolo bianco che si sovrappone alla figura
sa.patch.set_visible(False)
#rendo tutte le spines invisibili, salvo poi rendere visibile solo quella che mi interessa
for sp in sa.spines.values():#sa.spines è un dizionario
        sp.set_visible(False)

#rendo la spine a destra visibile e di colore verde
sa.spines['right'].set_visible(True)
sa.spines['right'].set_color('g')

#sposto la spine a destra rispetto al margine dell'area dati
#http://matplotlib.org/api/spine_api.html#matplotlib.spines.Spine.set_position
sa.spines['right'].set_position( ("axes", 1.02) ) #(tipo_posizione,posizione)

#imposto i limiti per l'asse y secondaria adattandoli alla serie di dati Z
sa.set_ylim(1.1*Z.min(),1.1*Z.max())

#rendo anche tutte le etichette verdi e con fontsize = 12
for label in sa.get_yticklabels():
    label.set_color('green')
    label.set_fontsize(12)



#plotto la serie di dati esplicitamente in sa, ma avrei potuto usare plt.plot
#perche' sa è il sistema di assi corrente
sa.plot(X, Z, color="green", linewidth=2.5, linestyle="--", label="somma scalata")

#aggiungo una nuova legenda per il nuovo sistema di assi
#direttamente con la dimensione del font desiderata
leg2 = sa.legend(fontsize=16)

###########################################################################
```

```
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)

mybb = dict(facecolor='white', edgecolor='None', alpha=0.65)
ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'],fontsize=16 ,
bbox=mybb)
ax.yaxis.set_ticks([-1, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$+1$'],fontsize=16 , bbox=mybb)


pl.savefig("./img/matplotlib12.png")
pl.show()
```

come si può notare ci sono problemi con le legende ...

Unificazione delle legende di due axes

creo questo grafico



versione di base

In [30]:
```
ax=pl.subplot(1,1,1)
#conservo i riferimenti ad ogni linea creata (plot restituisce una sequenza di linee ...)
l1, = ax.plot(X, C,label='coseno')
l2, = ax.plot(X, S,label='seno')
#genero un sistema di assi che condivide l'asse x con ax
ax2=ax.twinx()
l3, = ax2.plot(X , Z, 'r--' , label='somma scalata')

#creo una lista di linee da passare alla legenda
linee = [l1 , l2 , l3]
#equivalente:
#linee = ax.get_lines() + ax2.get_lines()

#creo una lista di etichette per la legenda
#attingo alle etichette associate a ciascuna linea
```

```
leglab=[]
for l in linee:
    leglab.append(l.get_label())

#creo la legenda passando esplicitamente linee ed etichette
#uso la legenda dell'asse secondario altrimenti solo le linee l1 ed l2
#sarebbero state coperte dalla legenda
ax2.legend(linee,leglab,loc='best')


pl.show()
```

versione che riprende l'esempio precedente

In [31]:
```
fig=pl.figure(figsize=(10, 6), dpi=80)
#recupero l'axis corrente che mi permette di accedere a ciascuna delle quattro spines
#(approccio orientato agli oggetti)
ax = pl.subplot(111)
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))
ax.patch.set_edgecolor('k')
ax.patch.set_linewidth(1)

ax.xaxis.set_ticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
ax.xaxis.set_ticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
ax.yaxis.set_ticks([-1, +1])
ax.yaxis.set_ticklabels([r'$-1$', r'$+1$'])
#ax.legend(loc='upper left')
ax.grid()
t = 2 * np.pi / 3
ax.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyle="--")
ax.scatter([t, ], [np.cos(t), ], 50, color='blue',marker='D')
ax.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$',#testo
            xy=(t, np.cos(t)), xycoords='data',   #coordinate e sistema di coordinate del punto
'mirato'
            xytext=(-90, -50), textcoords='offset points',# punto di inserimento del testo,
offset in punti rispetto alla punta della freccia
            fontsize=16, # dimensione del testo
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2")) #caratteristiche
del connettore

ax.xaxis.set_label_text('$angolo [rad]$',fontsize=18)
ax.xaxis.set_label_coords( x = 0.5 , y = -0.05)
ax.yaxis.set_label_text('$f(angolo)$',fontsize=18)
ax.yaxis.set_label_coords( x = -0.02 , y = 0.5)
ax.set_title('$Titolo figura$',fontsize=20)

#genero un sistema di assi che condivide con ax l'asse x e con asse y indipendente
#sa diventa il sistema corrente
sa=ax.twinx()
#vedi http://matplotlib.org/examples/pylab_examples/multiple_yaxis_with_spines.html
sa.set_frame_on(True)
#rendo il riempimento del nuovo sistema di assi invisibile
#altrimenti avrei un rettangolo bianco che si sovrappone alla figura
sa.patch.set_visible(False)
#rendo tutte le spines invisibili, salvo poi rendere visibile solo quella che mi interessa
for sp in sa.spines.values():#sa.spines è un dizionario
        sp.set_visible(False)
#rendo la spine a destra visibile e di colore verde
sa.spines['right'].set_visible(True)
sa.spines['right'].set_color('g')
#sposto la spine a destra rispetto al margine dell'area dati
```

```
sa.spines['right'].set_position(("axes", 1.02))
#rendo anche tutte le etichette verdi
for label in sa.get_yticklabels():
    label.set_color('green')
    label.set_fontsize(12)
#imposto i limiti per l'asse y secondaria adattandoli alla serie di dati Z
sa.set_ylim(1.1*Z.min(),1.1*Z.max())

################################################################################
l1,=ax.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="coseno")
l2,=ax.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="seno")
l3,=sa.plot(X, Z, color="green", linewidth=2.5, linestyle="--", label="somma scalata")


#linee = ax.get_lines() + sa.get_lines()#NON VA BENE, altrimenti avrei creato la legenda anche
per la linea verticale tratteggiata
linee=[l1,l2,l3]

#creo una lista di etichette per la legenda
leglab=[]
for l in linee:
    leglab.append(l.get_label())


#creo la legenda passando esplicitamente linee ed etichette
#uso la legenda dell'axes secondaria altrimenti la linea tratteggiata
#sta comunque sopra la legenda
leg=sa.legend(linee,leglab,loc='lower left')

#rendo la legenda semitrasparente
leg.get_frame().set_alpha(0.8)

################################################################################
ax.set_xlim(X.min() * 1.1, X.max() * 1.1)
ax.set_ylim(C.min() * 1.1, C.max() * 1.1)

#cambio font alle legende
for label in leg.get_texts() :
    label.set_fontsize(18)
#cambio font alle etichette
for label in ax.get_xticklabels() + ax.get_yticklabels():
    label.set_fontsize(16)
    label.set_bbox(dict(facecolor='white', edgecolor='None', alpha=0.65))

pl.savefig("./img/matplotlib13.png")
pl.show()
```