# Visualization in Python

Created by John C.S. Lui, June 3, 2018.

## Data Visualization

- Often, data analysts and scientists want to perform data visualization (e.g., plotting graphs, bar chart,..etc)
- Python provides a **RICH** set of plotting functionalities
- This lecture is meant to push you out of the Excel mindset, and introduce you to the popular Python library, e.g., mathplotlib
- We will (a) download data, (b) parse the data from columns & rows to list of dictionary, (c) then render the data

## Goals

- Run a Python file from the command line
- Make a simple graph

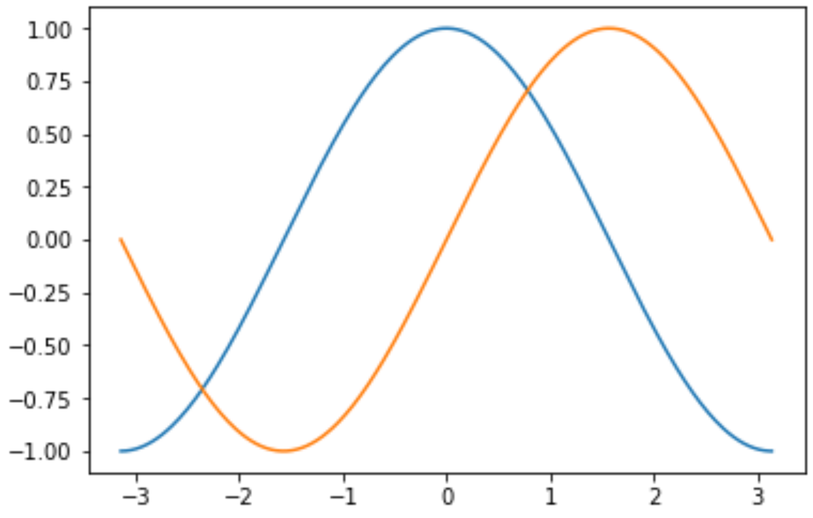
## Matplotlib

- matplotlib is a popular scientific library that gives the developer tools to produce 2D figures
- You need both numpy and matplotlib
- Rich examples: http://matplotlib.org/examples/index.html (http://matplotlib.org/examples/index.html)
- GeoJSON is a derivative of JSON. It's a data format for simple geological feature, including coordinate points.
- GitHub has an awesome feature that allows folks to paste GeoJSON files into Gists, and renders as a map

## Plot1.py

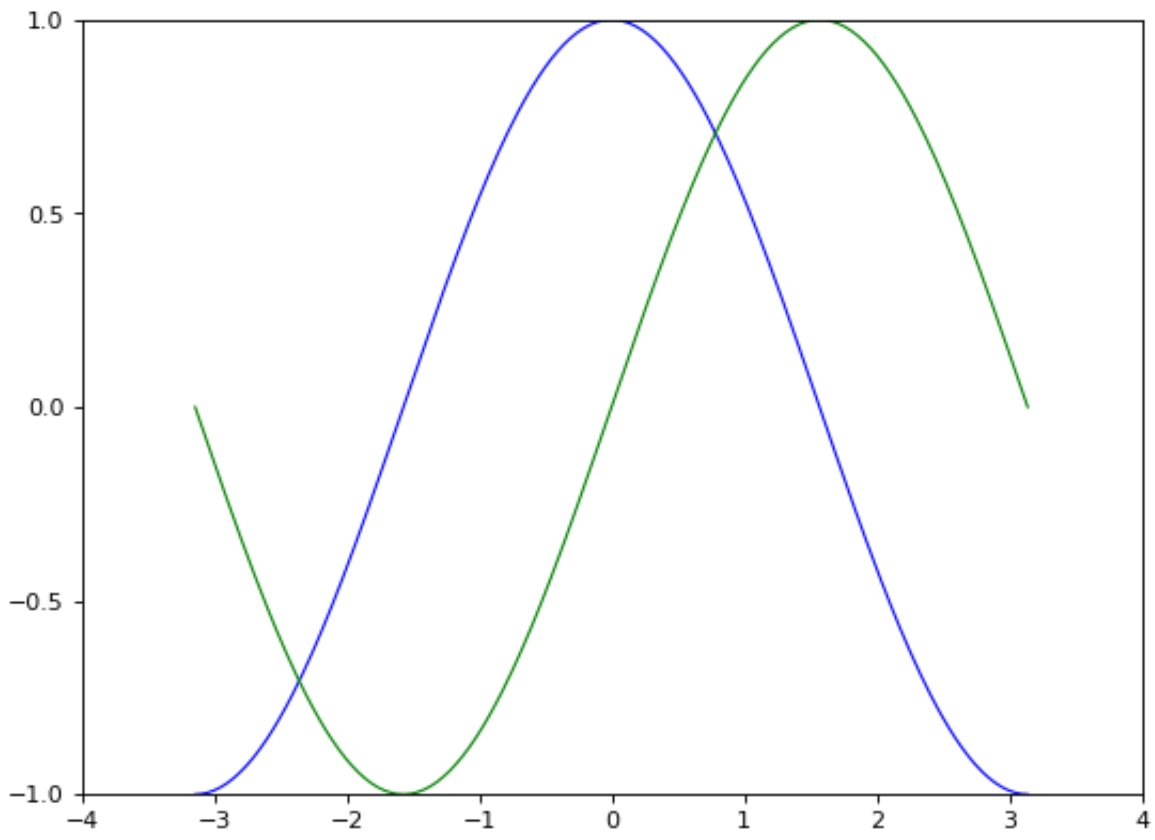
A simple x-y plot

In [2]:
```
# plot1.py
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

#print(X)
#print(C)
#print(S)

plt.plot(X, C)       # plot C vs. X
plt.plot(X, S)       # plot S vs. X

plt.show()    # display all plots
```



In [3]:
```
# plot2.py
import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(8, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

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

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

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

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

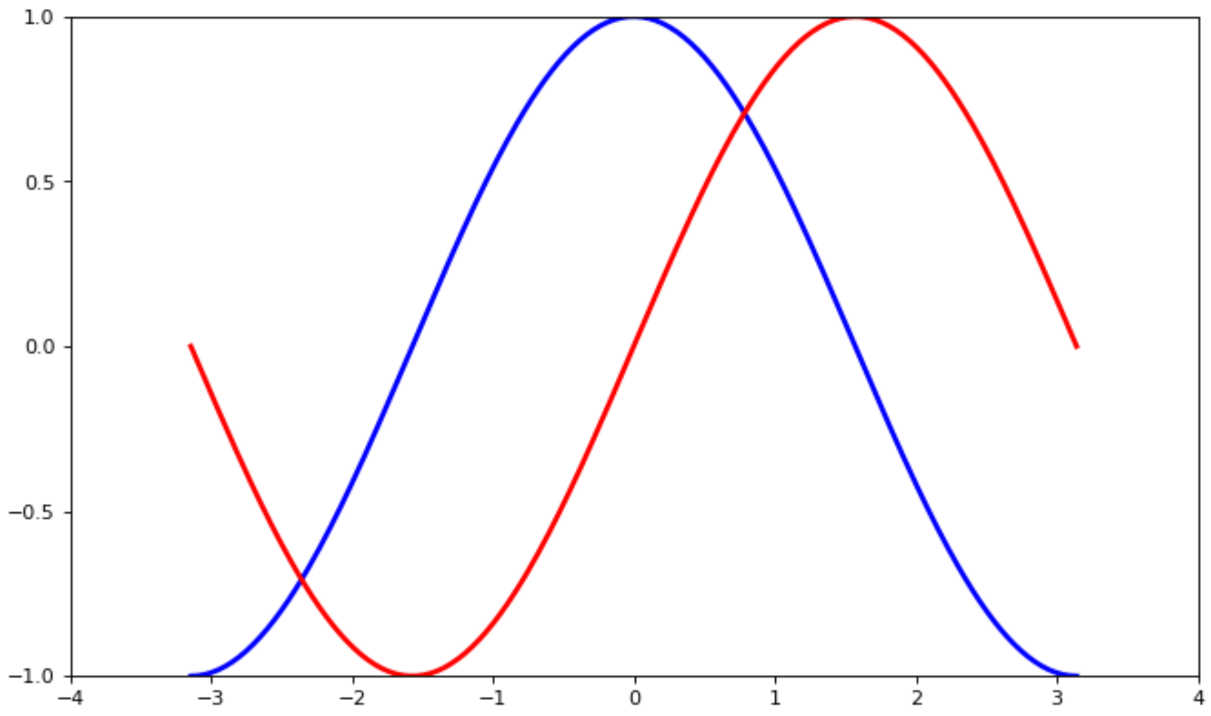
```
# Set y limits
plt.ylim(-1.0, 1.0)

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

# Save figure using 72 dots per inch
plt.savefig("exercice_2.png", dpi=72)

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



In [4]:

```
# ploy3.py


import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

# Plot cosine with a blue continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")

# Plot sine with a green continuous line of width 1 (pixels)
plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")
```

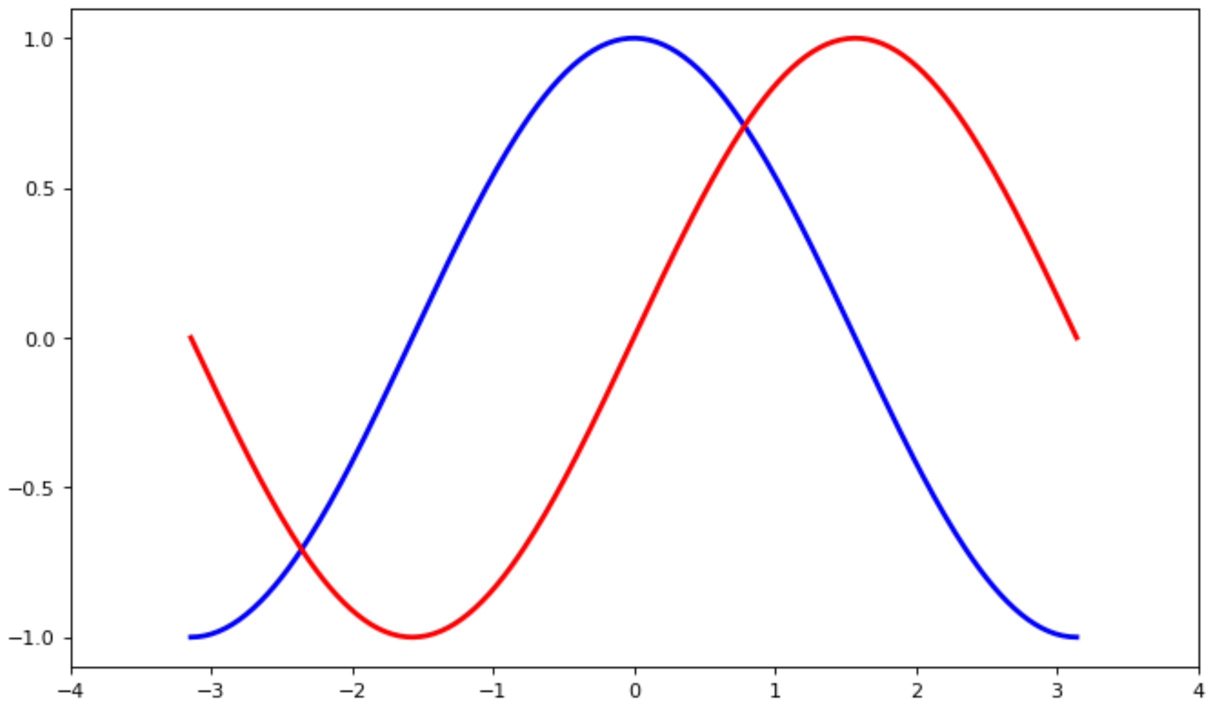
```
# Set x limits
plt.xlim(-4.0, 4.0)

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

# Set y limits
plt.ylim(-1.0, 1.0)

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

# Save figure using 72 dots per inch
plt.savefig("exercice_2.png", dpi=72)

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



In [5]:

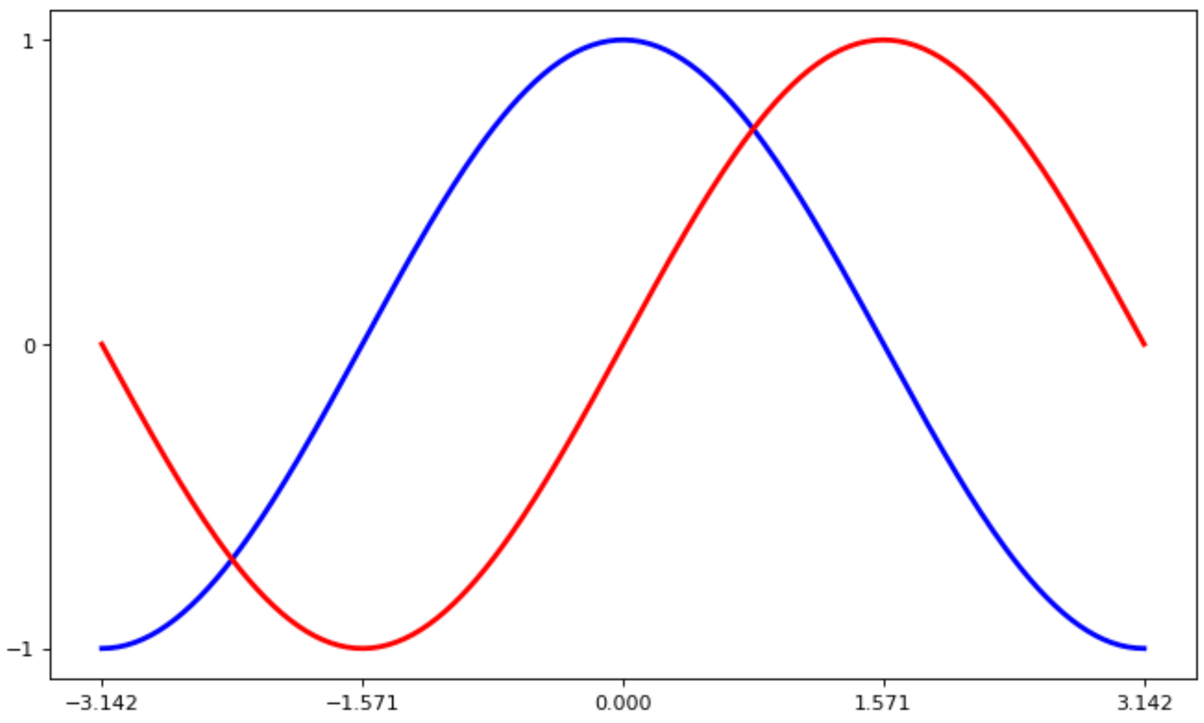
```
# plot4.py


import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

# Plot cosine with a blue continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
```

```
# Plot sine with a green continuous line of width 1 (pixels)
plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")

# Set x limits
plt.xlim(X.min() * 1.1, X.max() * 1.1)

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

# Set y limits
plt.ylim(C.min() * 1.1, C.max() * 1.1)

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

# Save figure using 72 dots per inch
plt.savefig("exercice_2.png", dpi=72)

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



In [6]:
```
# plot5.py



import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)
```

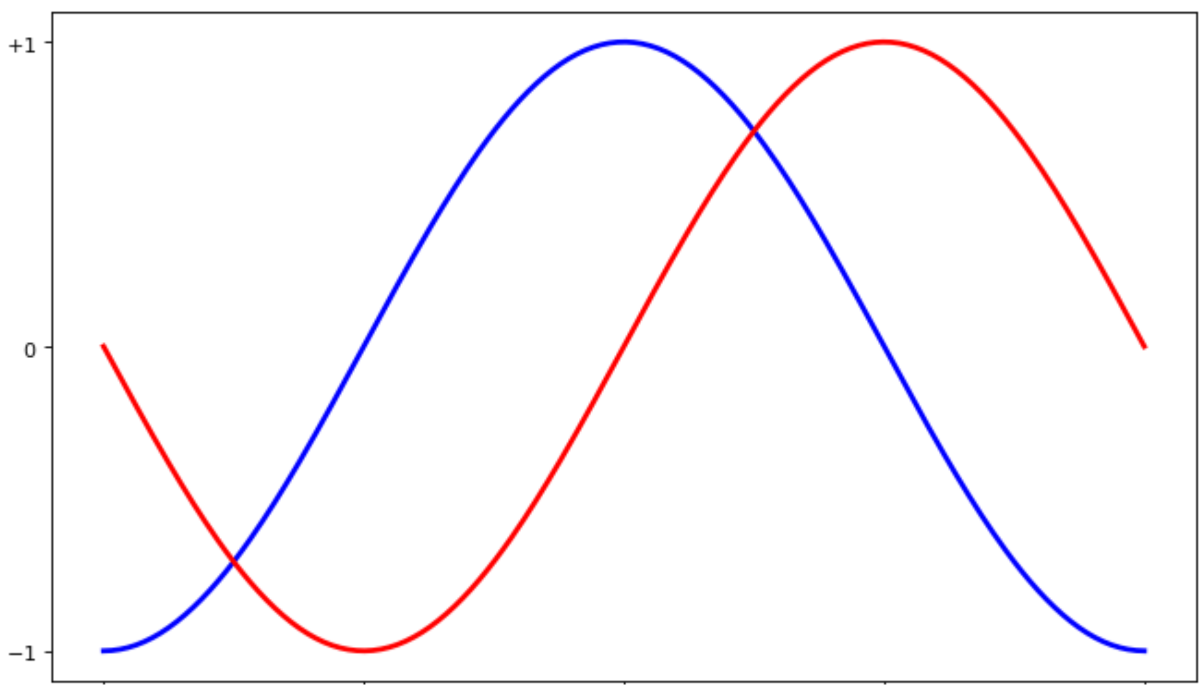
```python
X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

# Plot cosine with a blue continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")

# Plot sine with a green continuous line of width 1 (pixels)
plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")

# Set x limits
plt.xlim(X.min() * 1.1, X.max() * 1.1)

# Set x ticks
#plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])

# Set y limits
plt.ylim(C.min() * 1.1, C.max() * 1.1)

# Set y ticks
#plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
plt.yticks([-1, 0, +1])

# Save figure using 72 dots per inch
plt.savefig("exercice_2.png", dpi=72)

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

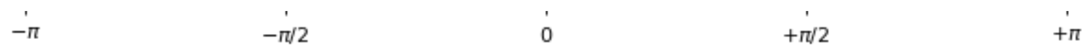


In [8]:

```python
# plot6.py

import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
```

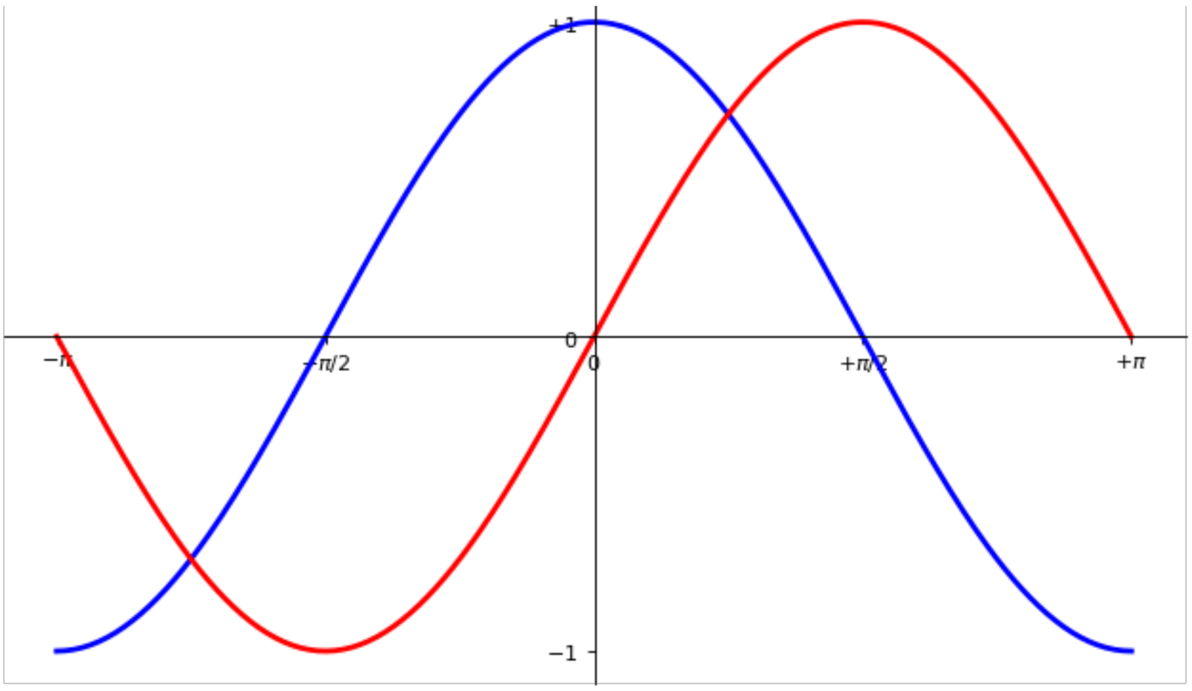
```python
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

# Plot cosine with a blue continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")

# Plot sine with a green continuous line of width 1 (pixels)
plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")

# Set x limits
plt.xlim(X.min() * 1.1, X.max() * 1.1)

# Set x ticks
#plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
#plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

# Set y limits
plt.ylim(C.min() * 1.1, C.max() * 1.1)

# Set y ticks
#plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
#plt.yticks([-1, 0, +1])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])

# Save figure using 72 dots per inch
plt.savefig("exercice_2.png", dpi=72)

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





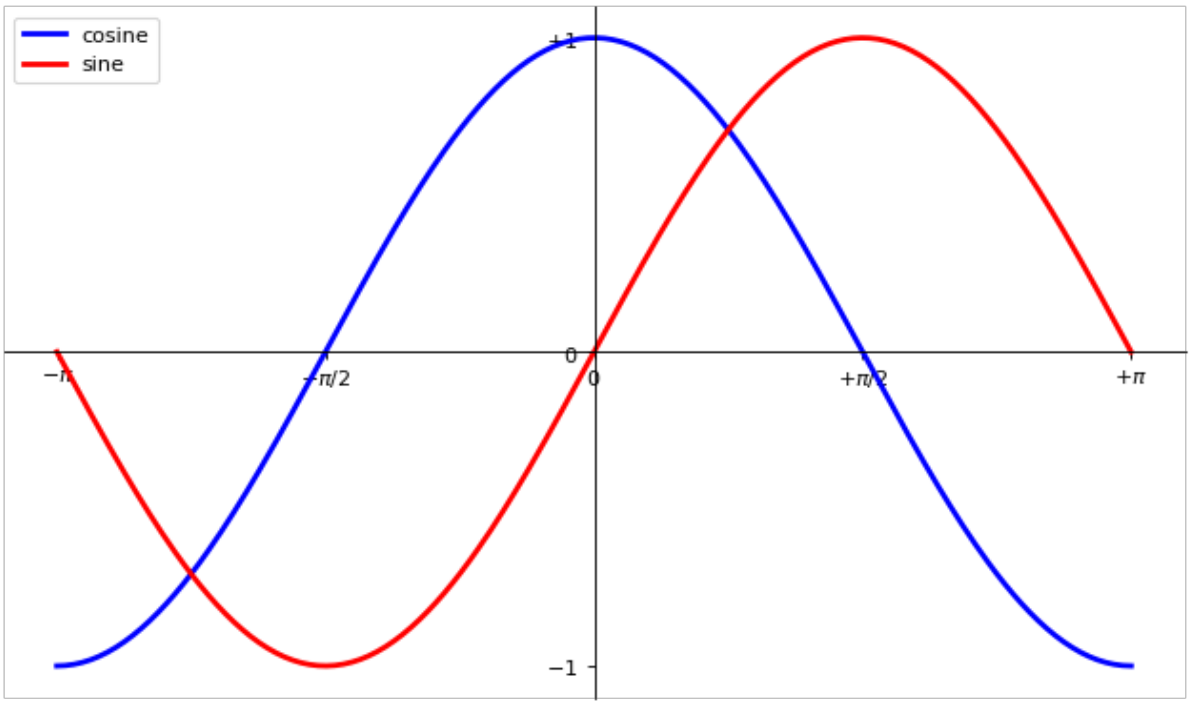
In [9]:

```python
# plot7.py

import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

# Plot cosine with a blue continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")

# Plot sine with a green continuous line of width 1 (pixels)
plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")

# Set x limits
plt.xlim(X.min() * 1.1, X.max() * 1.1)

# Set x ticks
#plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
#plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

# Set y limits
plt.ylim(C.min() * 1.1, C.max() * 1.1)

# Set y ticks
#plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
#plt.yticks([-1, 0, +1])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])

# Save figure using 72 dots per inch
#plt.savefig("exercice_2.png", dpi=72)

ax = plt.gca()  # gca stands for 'get current axis'
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))



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



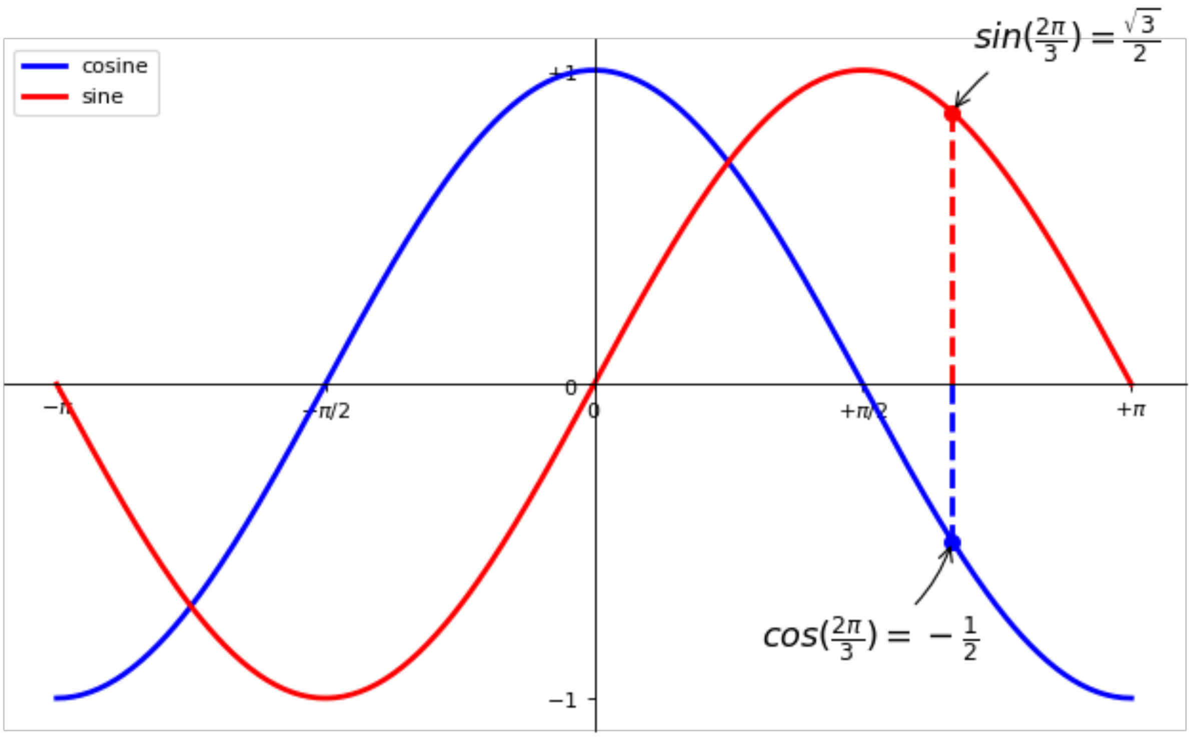
In [10]:
```
# plot8.py

import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

# Plot cosine with a blue continuous line of width 1 (pixels)
#plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosi

# Plot sine with a green continuous line of width 1 (pixels)
#plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine


# Set x limits
plt.xlim(X.min() * 1.1, X.max() * 1.1)

# Set x ticks
#plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
#plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

# Set y limits
plt.ylim(C.min() * 1.1, C.max() * 1.1)

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

```
#plt.yticks([-1, 0, +1])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])

# Save figure using 72 dots per inch
#plt.savefig("exercice_2.png", dpi=72)

ax = plt.gca()  # gca stands for 'get current axis'
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

# Print legend
plt.legend(loc='upper left')

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



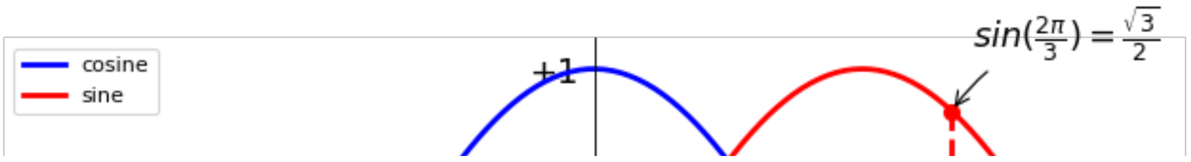
In [11]:
```
# plot9.py

import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)
```

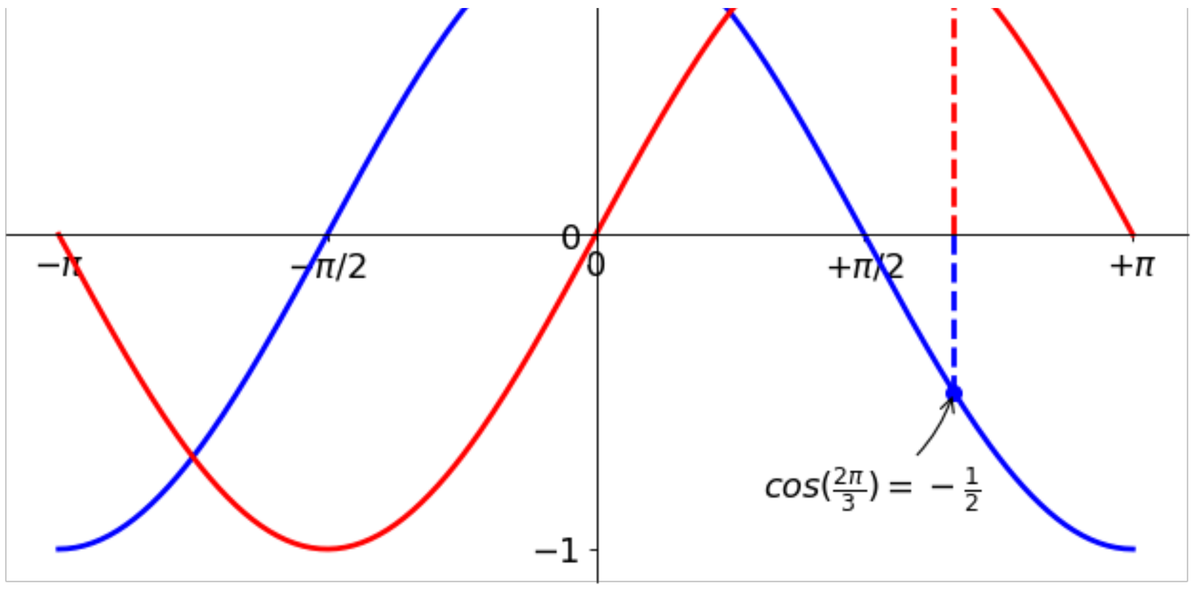
```
# Plot cosine with a blue continuous line of width 1 (pixels)
#plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosi

# Plot sine with a green continuous line of width 1 (pixels)
#plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine


# Set x limits
plt.xlim(X.min() * 1.1, X.max() * 1.1)

# Set x ticks
#plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
#plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

# Set y limits
plt.ylim(C.min() * 1.1, C.max() * 1.1)

# Set y ticks
#plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
#plt.yticks([-1, 0, +1])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])

# Save figure using 72 dots per inch
#plt.savefig("exercice_2.png", dpi=72)

ax = plt.gca()  # gca stands for 'get current axis'
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

# annotate some points
t = 2 * np.pi / 3
plt.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyl
plt.scatter([t, ], [np.cos(t), ], 50, color='blue')

plt.annotate(r'$sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
             xy=(t, np.sin(t)), xycoords='data',
             xytext=(+10, +30), textcoords='offset points', fontsize=1
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,ra

plt.plot([t, t],[0, np.sin(t)], color='red', linewidth=2.5, linestyle=
plt.scatter([t, ],[np.sin(t), ], 50, color='red')

plt.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$',
             xy=(t, np.cos(t)), xycoords='data',
             xytext=(-90, -50), textcoords='offset points', fontsize=1
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,ra

# Print legend
plt.legend(loc='upper left')
plt.legend(loc='upper left')

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



In [12]:
```
# plot10.py



import numpy as np
import matplotlib.pyplot as plt

# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(10, 6), dpi=80)

# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

# Plot cosine with a blue continuous line of width 1 (pixels)
#plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosi

# Plot sine with a green continuous line of width 1 (pixels)
#plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine


# Set x limits
plt.xlim(X.min() * 1.1, X.max() * 1.1)

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

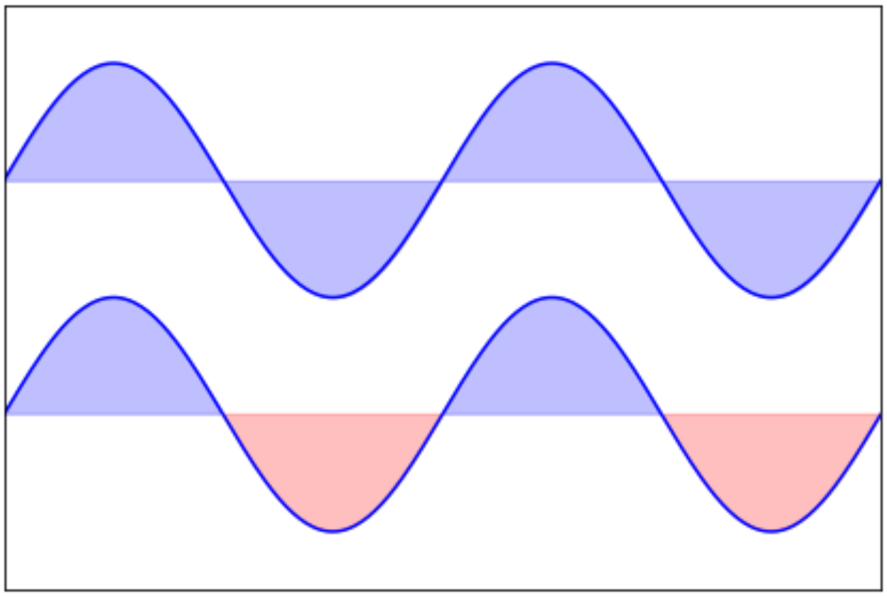
```
#plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

# Set y limits
plt.ylim(C.min() * 1.1, C.max() * 1.1)

# Set y ticks
#plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
#plt.yticks([-1, 0, +1])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])

# Save figure using 72 dots per inch
#plt.savefig("exercice_2.png", dpi=72)

ax = plt.gca()  # gca stands for 'get current axis'
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

# annotate some points
t = 2 * np.pi / 3
plt.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyl
plt.scatter([t, ], [np.cos(t), ], 50, color='blue')

plt.annotate(r'$sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
             xy=(t, np.sin(t)), xycoords='data',
             xytext=(+10, +30), textcoords='offset points', fontsize=1
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,ra

plt.plot([t, t],[0, np.sin(t)], color='red', linewidth=2.5, linestyle=
plt.scatter([t, ],[np.sin(t), ], 50, color='red')

plt.annotate(r'$cos(\frac{2\pi}{3})=-\frac{1}{2}$',
             xy=(t, np.cos(t)), xycoords='data',
             xytext=(-90, -50), textcoords='offset points', fontsize=1
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,ra

# Print legend
plt.legend(loc='upper left')

# reset x-labels
for label in ax.get_xticklabels() + ax.get_yticklabels():
    label.set_fontsize(16)
    label.set_bbox(dict(facecolor='white', edgecolor='None', alpha=0.6

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



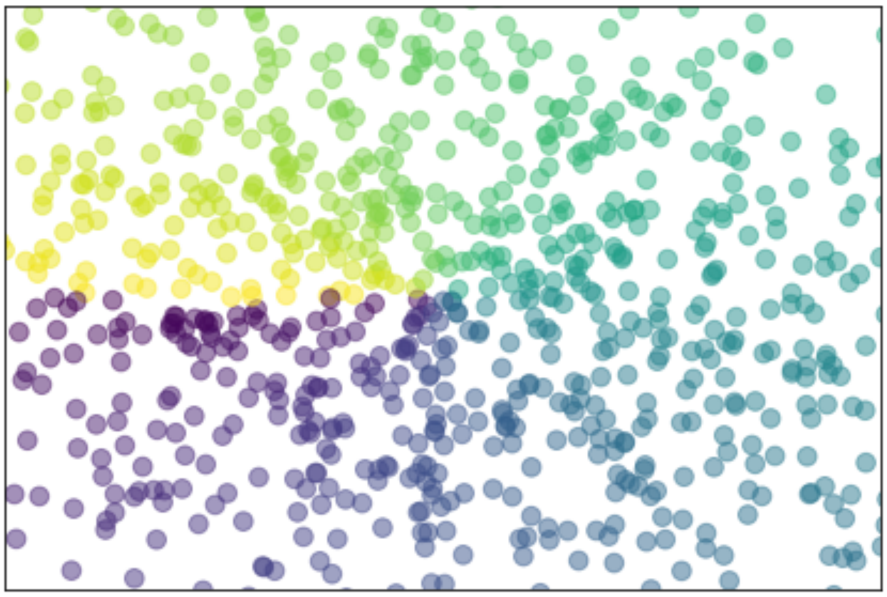


$-\pi$
$-\pi/2$
$0$
$+\pi/2$
$+\pi$
$0$
$-1$
$cos(\frac{2\pi}{3}) = -\frac{1}{2}$

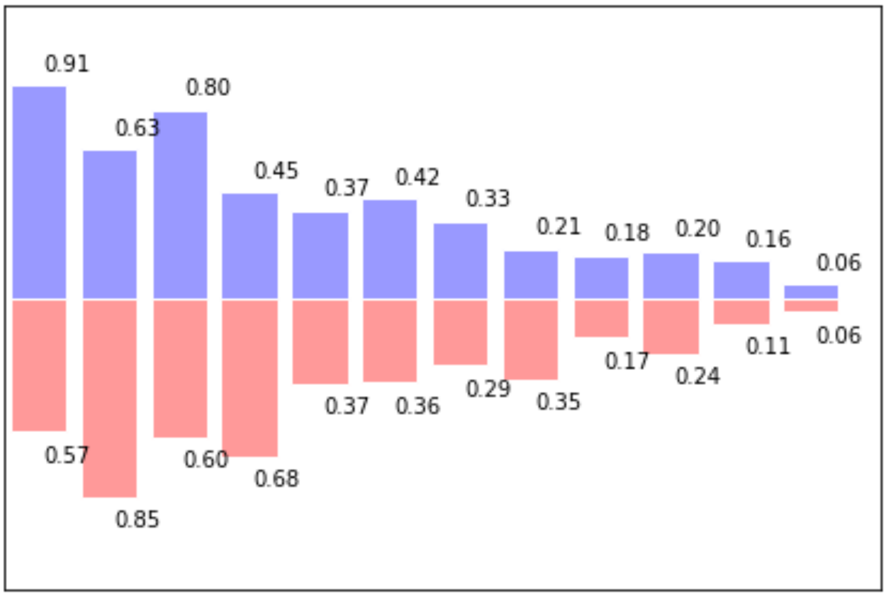
In [13]:
```
# plot11.py

import numpy as np
import matplotlib.pyplot as plt

n = 256
X = np.linspace(-np.pi, np.pi, n, endpoint=True)
Y = np.sin(2 * X)

plt.axes([0.025, 0.025, 0.95, 0.95])

plt.plot(X, Y + 1, color='blue', alpha=1.00)
plt.fill_between(X, 1, Y + 1, color='blue', alpha=.25)

plt.plot(X, Y - 1, color='blue', alpha=1.00)
plt.fill_between(X, -1, Y - 1, (Y - 1) > -1, color='blue', alpha=.25)
plt.fill_between(X, -1, Y - 1, (Y - 1) < -1, color='red',  alpha=.25)

plt.xlim(-np.pi, np.pi)
plt.xticks(())
plt.ylim(-2.5, 2.5)
plt.yticks(())

plt.show()
```



In [14]:
```
# plot12.py


import numpy as np
import matplotlib.pyplot as plt

n = 1024
X = np.random.normal(0, 1, n)
Y = np.random.normal(0, 1, n)
T = np.arctan2(Y, X)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.scatter(X, Y, s=75, c=T, alpha=.5)

plt.xlim(-1.5, 1.5)
plt.xticks(())
plt.ylim(-1.5, 1.5)
plt.yticks(())

plt.show()
```

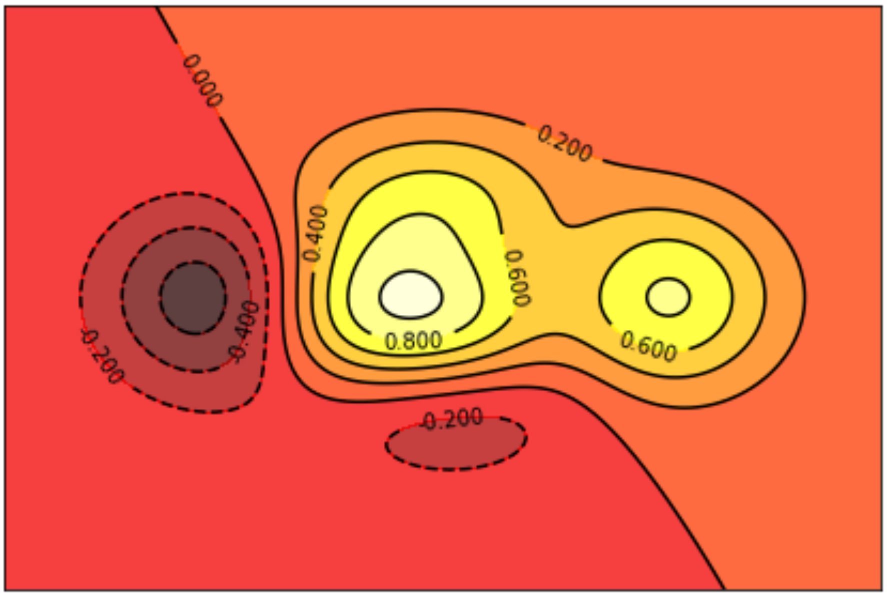


In [15]:
```
# plot13.py


import numpy as np
import matplotlib.pyplot as plt

n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')

for x, y in zip(X, Y1):
    plt.text(x + 0.4, y + 0.05, '%.2f' % y, ha='center', va= 'bottom')

for x, y in zip(X, Y2):
    plt.text(x + 0.4, -y - 0.05, '%.2f' % y, ha='center', va= 'top')

plt.xlim(-.5, n)
plt.xticks(())
plt.ylim(-1.25, 1.25)
plt.yticks(())

plt.show()
```

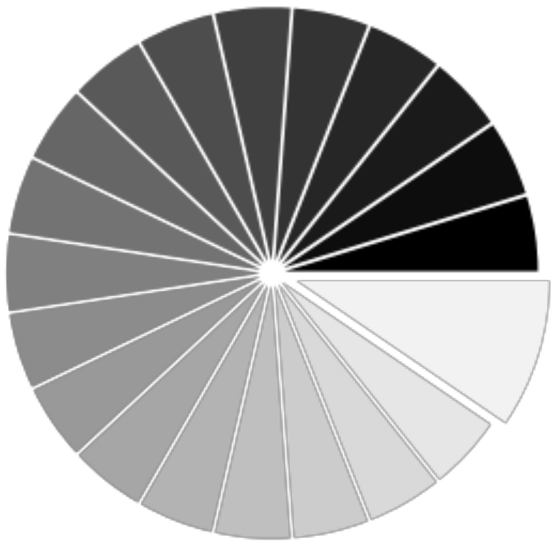


In [16]:
```python
# plot14.py

import numpy as np
import matplotlib.pyplot as plt

def f(x,y):
    return (1 - x / 2 + x**5 + y**3) * np.exp(-x**2 -y**2)

n = 256
x = np.linspace(-3, 3, n)
y = np.linspace(-3, 3, n)
X,Y = np.meshgrid(x, y)

plt.axes([0.025, 0.025, 0.95, 0.95])

plt.contourf(X, Y, f(X, Y), 8, alpha=.75, cmap=plt.cm.hot)
C = plt.contour(X, Y, f(X, Y), 8, colors='black', linewidth=.5)
plt.clabel(C, inline=1, fontsize=10)

plt.xticks(())
plt.yticks(())
plt.show()
```

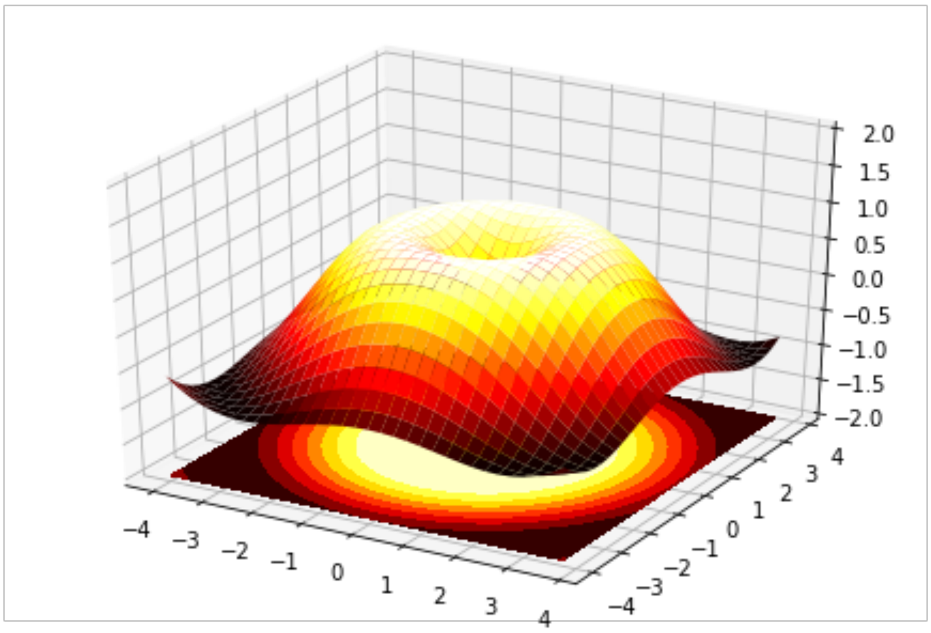
```
/anaconda3/lib/python3.6/site-packages/matplotlib/contour.py:967: Us
erWarning: The following kwargs were not used by contour: 'linewidth
'
  s)
```



```
In [17]: # plot15.py



import numpy as np
import matplotlib.pyplot as plt

n = 20
Z = np.ones(n)
Z[-1] *= 2

plt.axes([0.025, 0.025, 0.95, 0.95])

plt.pie(Z, explode=Z*.05, colors = ['%f' % (i/float(n)) for i in range
plt.axis('equal')
plt.xticks(())
plt.yticks()

plt.show()
```



In [18]:
```python
# plot16.py

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = Axes3D(fig)
X = np.arange(-4, 4, 0.25)
Y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X ** 2 + Y ** 2)
Z = np.sin(R)

ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=plt.cm.hot)
ax.contourf(X, Y, Z, zdir='z', offset=-2, cmap=plt.cm.hot)
ax.set_zlim(-2, 2)

plt.show()
```



In [ ]: