Matplotlib in python

一、简介

Matplotlib是最常用的独立2维绘图库. 它能够快速将数据可视化，并输出到多种格式的，高质量的图片。 下面我们将会介绍一些常用的例子，它们将会涵盖大多数使用场景

首先，在 **IPython notebook** 中插入 **%matplotlib inline**，以便直接在当页输出图像

In [1]:

```
%matplotlib inline
```

matplotlib的**pyplot子库**提供了和matlab类似的绘图API，方便用户快速绘制2D图表。

In [2]:

```
from matplotlib import pyplot as plt
```

二、绘图基础

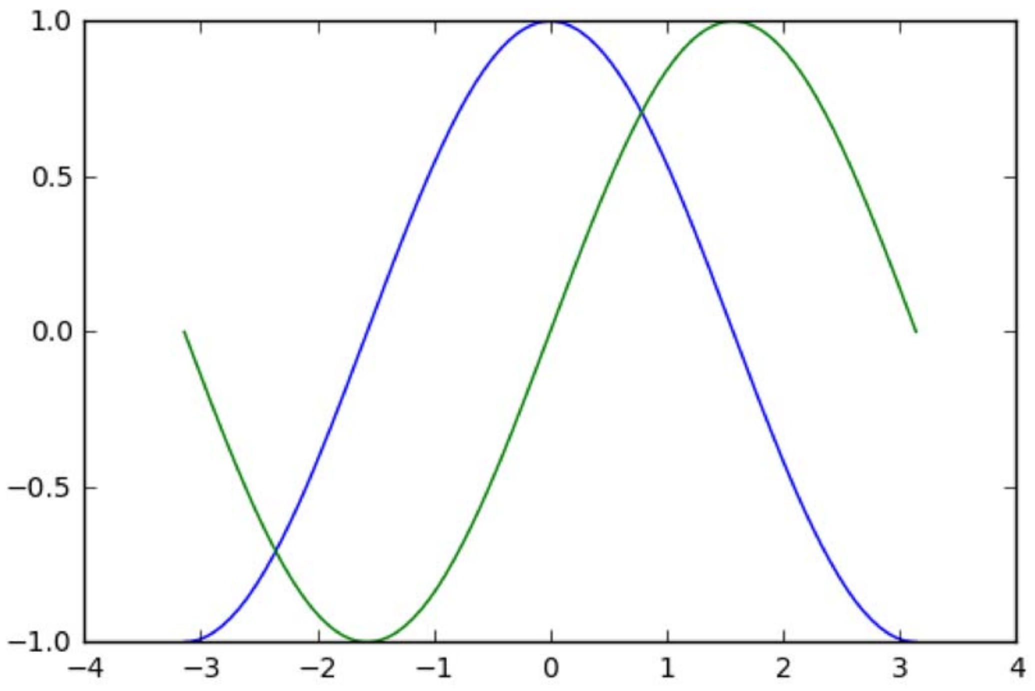
这一节，我们将会在同一画布上绘制正余弦函数的图像。并且在之后逐步添加细节。

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
```

In [41]:

```
X = np.linspace(-np.pi, np.pi, 256, endpoint = True) # X是一个numpy数组，包含从-π到+π的256个数据点(包含端点)
C, S = np.cos(X), np.sin(X) # C和S分别是相应的正余弦函数值
plt.plot(X, C)
plt.plot(X, S)
plt.show()
```



Matplotlib 图像高度可定制， 你可以更改 matplotlib 中几乎所有的设置: 图像大小和分辨率，线的宽度，颜色和样式, 坐标轴, 插入文字和字体等等。

改变线宽和颜色

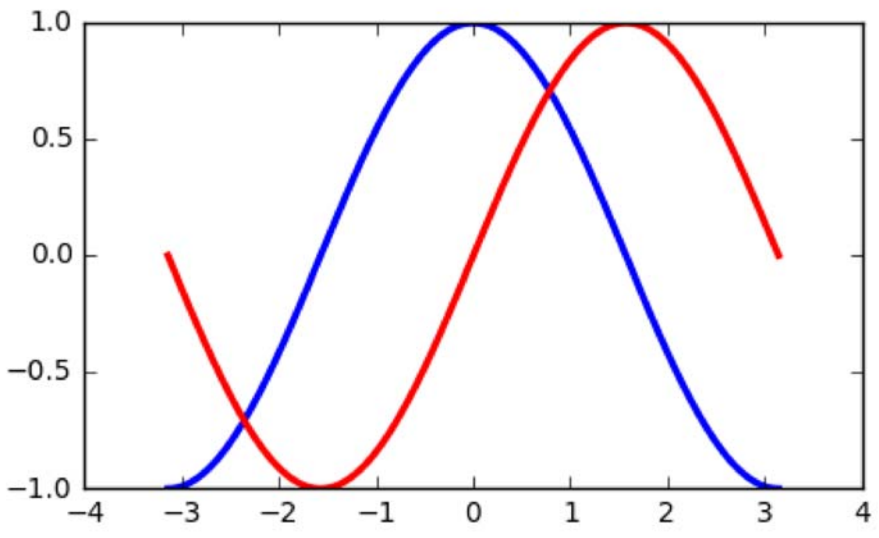
把余弦曲线改为蓝色，正弦曲线改为红色，并加粗。

In [42]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x8eabdd8>]
```



设置坐标范围

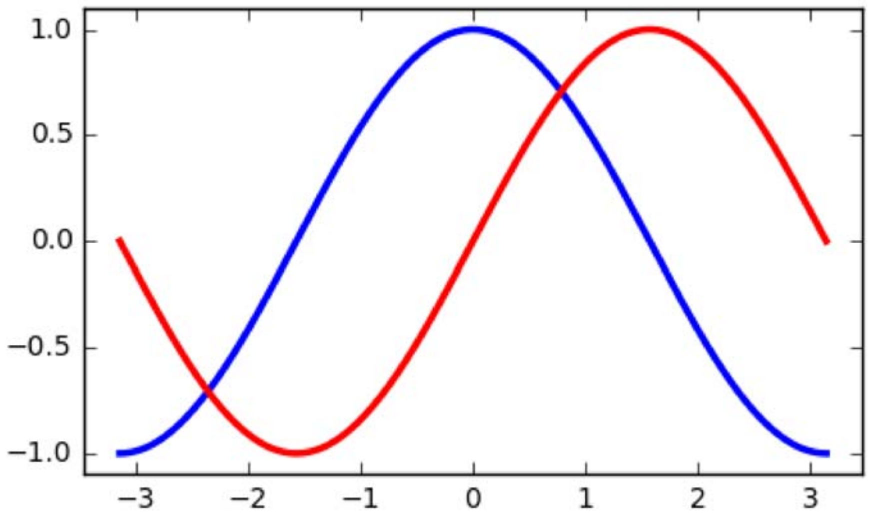
默认的坐标范围较小。为了清晰地呈现所有数据点，我们将坐标范围设置大一点。

In [43]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
```

Out[43]:

```
(-1.1000000000000001, 1.0999165211263138)
```



设置坐标轴刻度

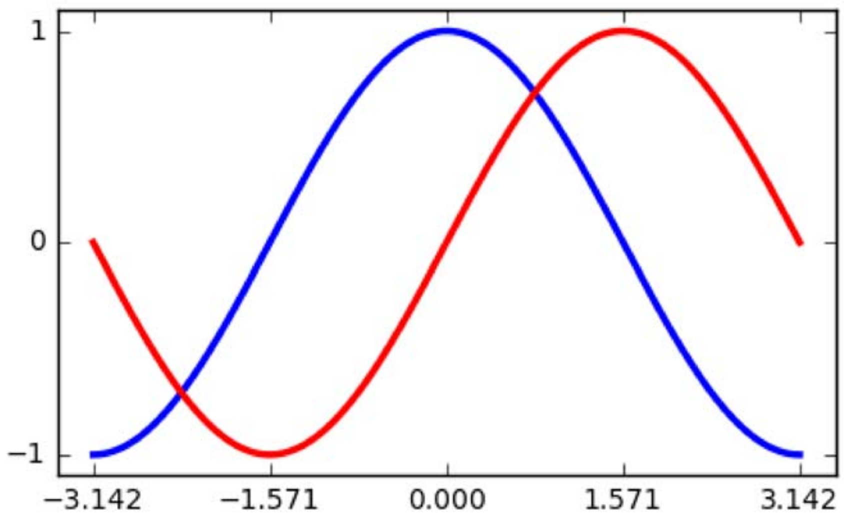
现有的坐标轴没有（+/-π,+/-π/2) 刻度，通过以下代码设置这些坐标点：

In [45]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, +1])
```

Out[45]:

```
([<matplotlib.axis.YTick at 0xb28efd0>,
  <matplotlib.axis.YTick at 0xb28eb38>,
  <matplotlib.axis.YTick at 0xc3e62b0>],
 <a list of 3 Text yticklabel objects>)
```



设置刻度标签

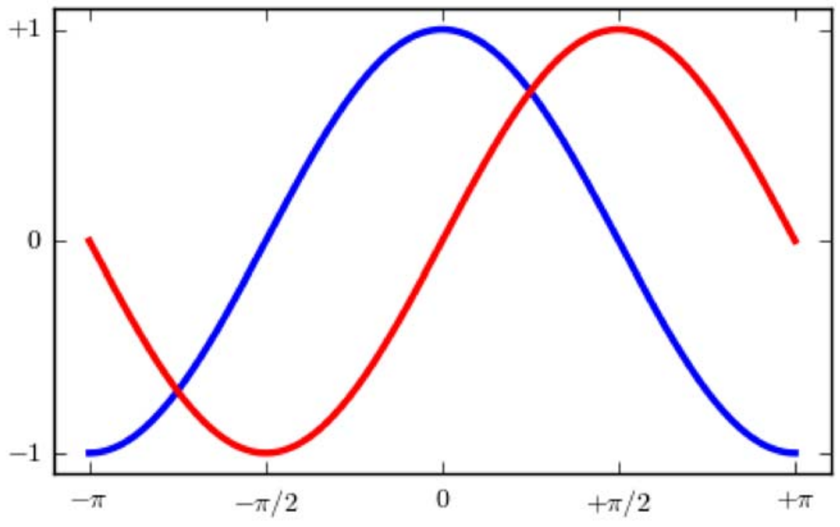
现在刻度已经成功设置好了，但是我们想把3.142显式设置为 π 。 为了做到这一点，在 ‘xticks()’ 和 ‘yticks()’ 中传入第二个参数列表.（这里使用了latex公式，以便更加美观。 ）

In [46]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])
```

Out[46]:

```
([<matplotlib.axis.YTick at 0xbf52f98>,
  <matplotlib.axis.YTick at 0xbf52b00>,
  <matplotlib.axis.YTick at 0xc0d2320>],
 <a list of 3 Text yticklabel objects>)
```

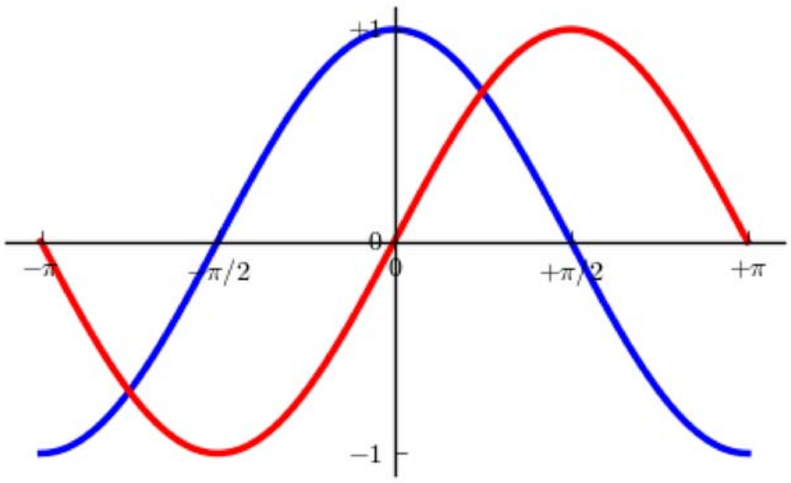


画坐标轴

Spines是连接坐标刻度和标记数据区域的线条. 它们可以被置于图形任意位置. 我们现在把它们移动到图形中央位置。因为总共有4根线条(top/bottom/left/right), 我们 top 和 right 两线条设置为无色，把 bottom 和 left 移动 0 坐标处。

In [47]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])
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))
```



添加图例

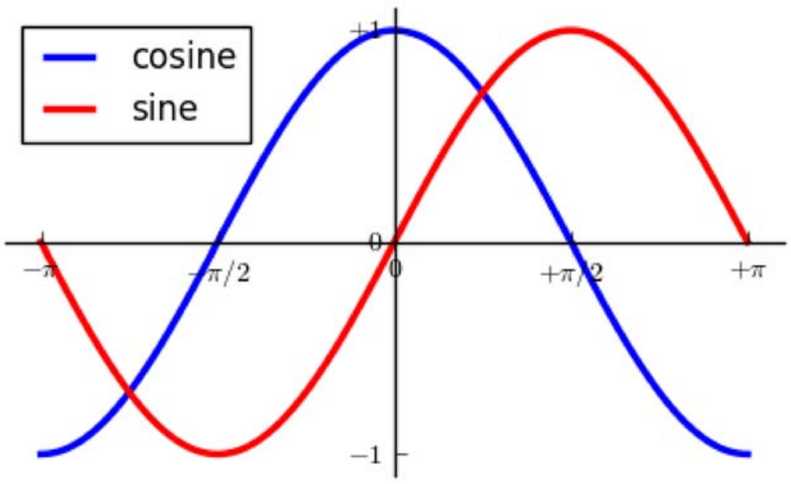
通过在plot()中添加label参数，并设置legend(),在图形左上角图例。

In [48]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine") #add label
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine") #add label
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])
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))
plt.legend(loc='upper left') #show legend
```

Out[48]:

```
<matplotlib.legend.Legend at 0xc897208>
```



标注数据点

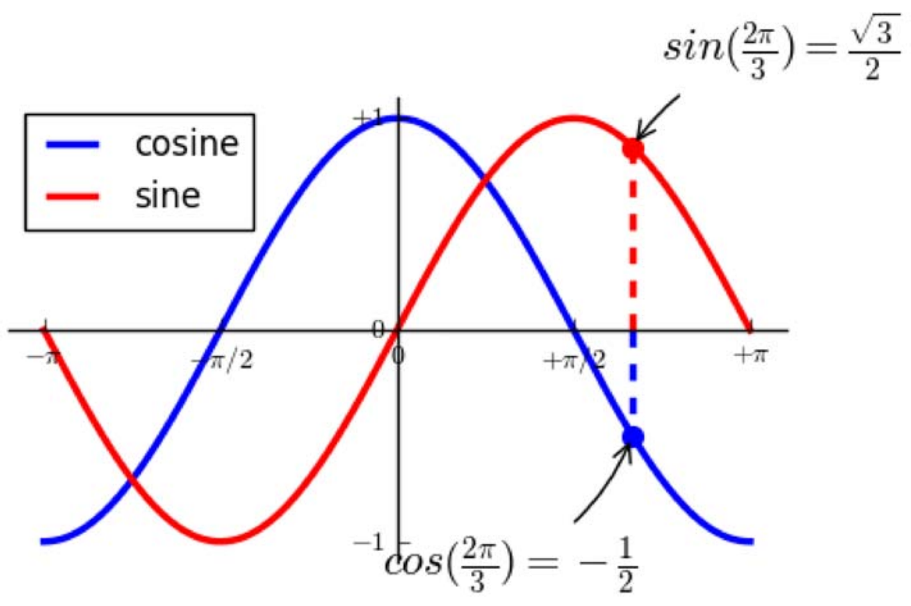
通过 annotate() 在图形中添加注释。在正余弦曲线的 2π/3 处添加 标注，首先在曲线相应位置打上记号，并记号点与坐标轴之间添加一条竖直虚线。 接下来，使用 annotate() 添加带箭头的文字标注。

In [49]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine") #add label
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine") #add label
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])
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))
plt.legend(loc='upper left') #show legend

t = 2 * np.pi / 3
plt.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyle="--")
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=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

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=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))
```

Out[49]:

```
<matplotlib.text.Annotation at 0xc20c7b8>
```

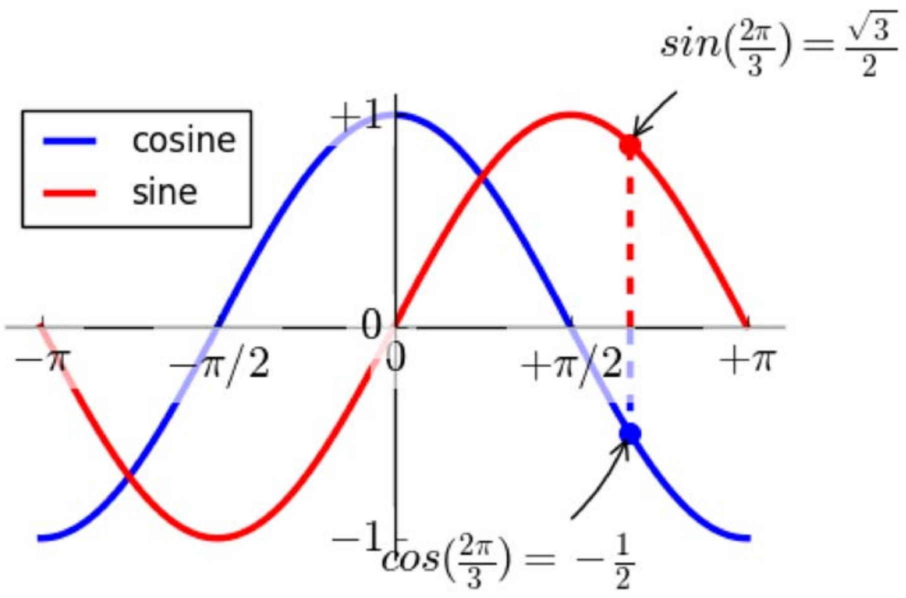


细节决定成败

刻度标签因为线条的遮挡不易看清，通过改变字体大小和背景透明度可以 线条和标签同时可见。

In [50]:

```
plt.figure(figsize=(5, 3), dpi=80)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine") #add label
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine") #add label
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, +1],
          [r'$-1$', r'$0$', r'$+1$'])
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))
plt.legend(loc='upper left') #show legend

t = 2 * np.pi / 3
plt.plot([t, t], [0, np.cos(t)], color='blue', linewidth=2.5, linestyle="--")
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=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

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=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

for label in ax.get_xticklabels() + ax.get_yticklabels():
    label.set_fontsize(16)
    label.set_bbox(dict(facecolor='white', edgecolor='None', alpha=0.65))
```

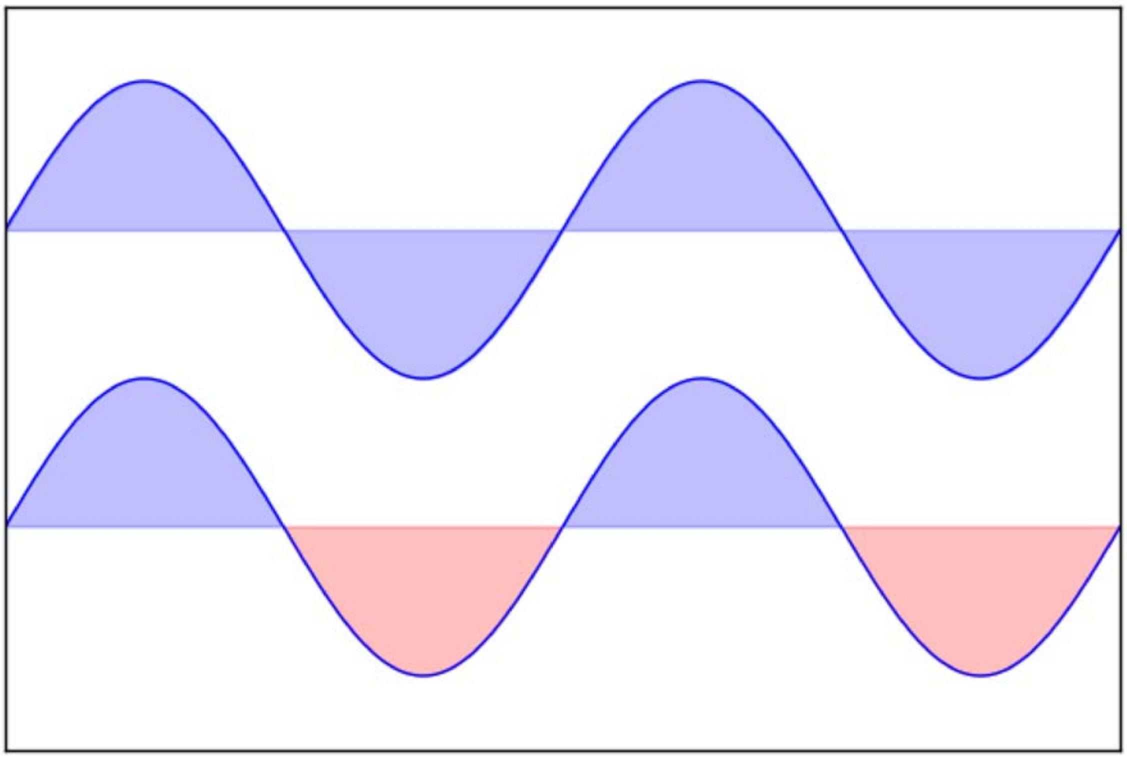


三、常见图像

正余弦曲线

In [28]:

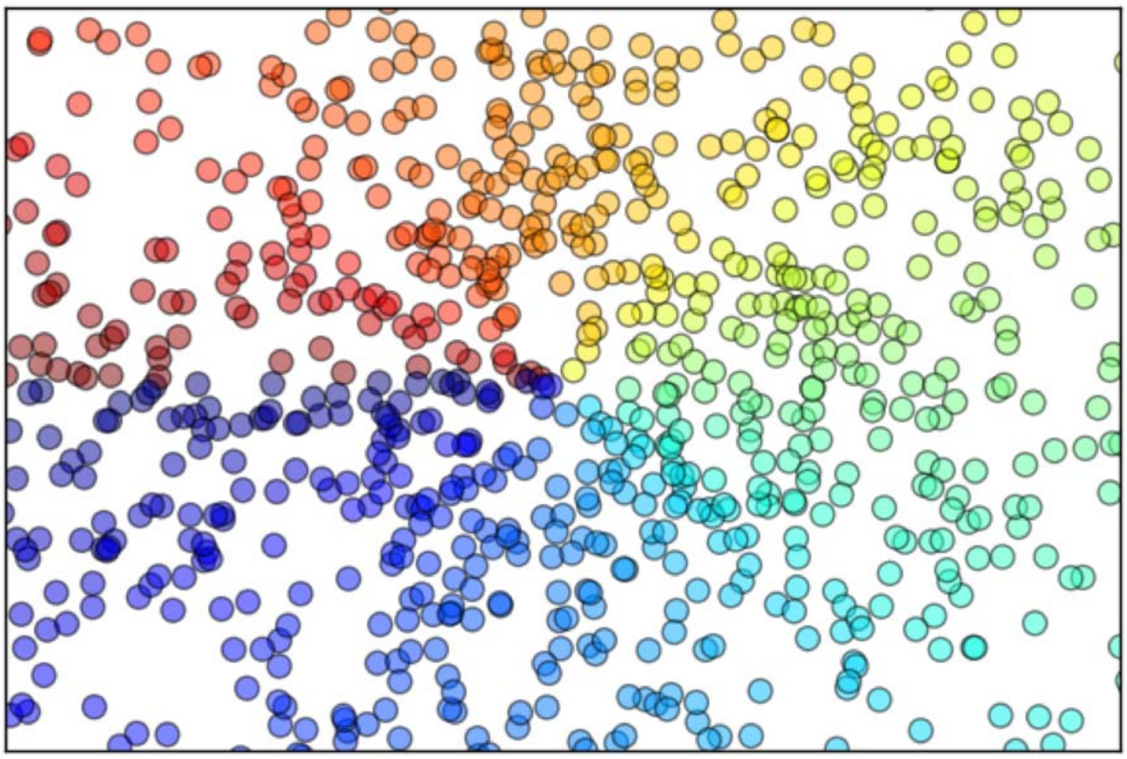
```
import numpy as np
import matplotlib.pyplot as plt

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()
```



散点图

In [29]:

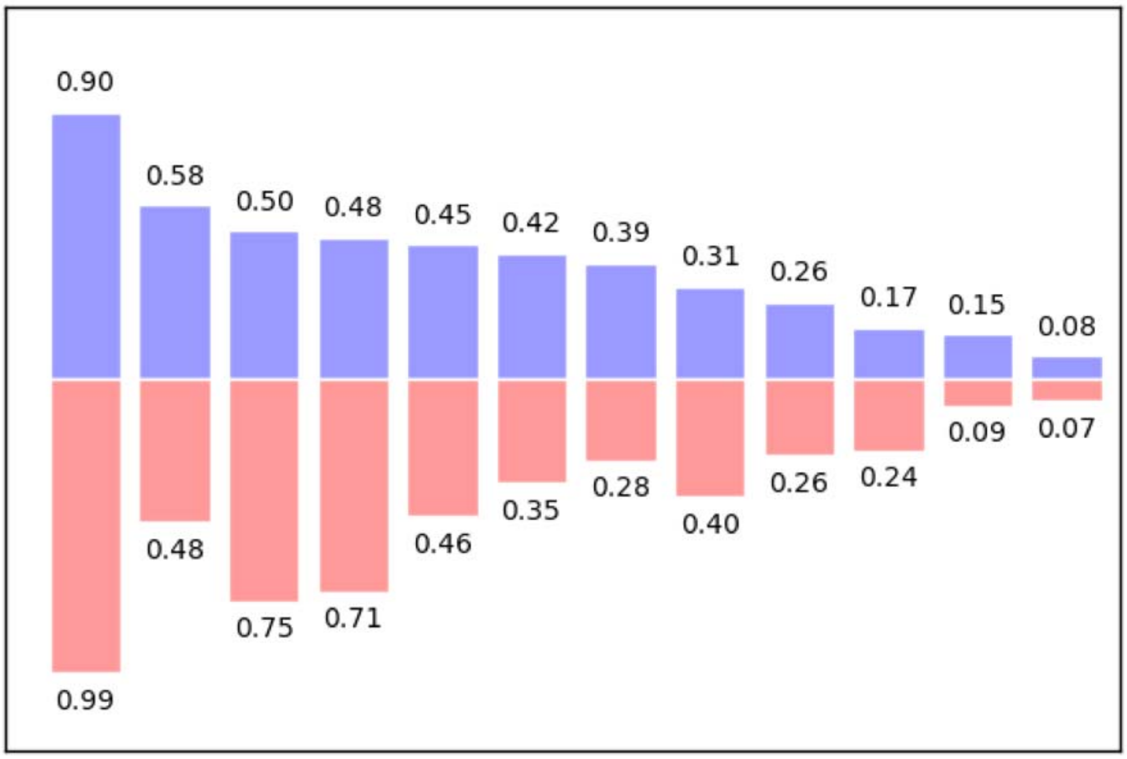
```
import numpy as np
import matplotlib.pyplot as plt

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()
```



条形图

In [30]:

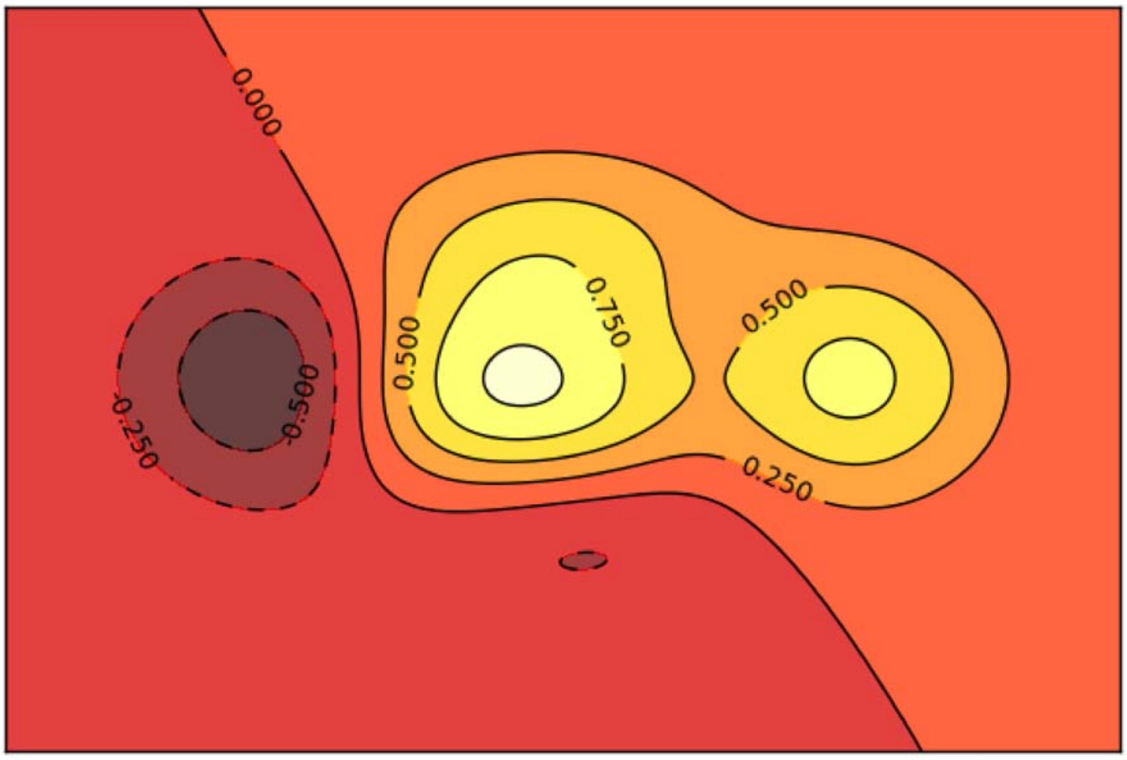
```
import numpy as np
import matplotlib.pyplot as plt

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()
```



等高线

In [31]:

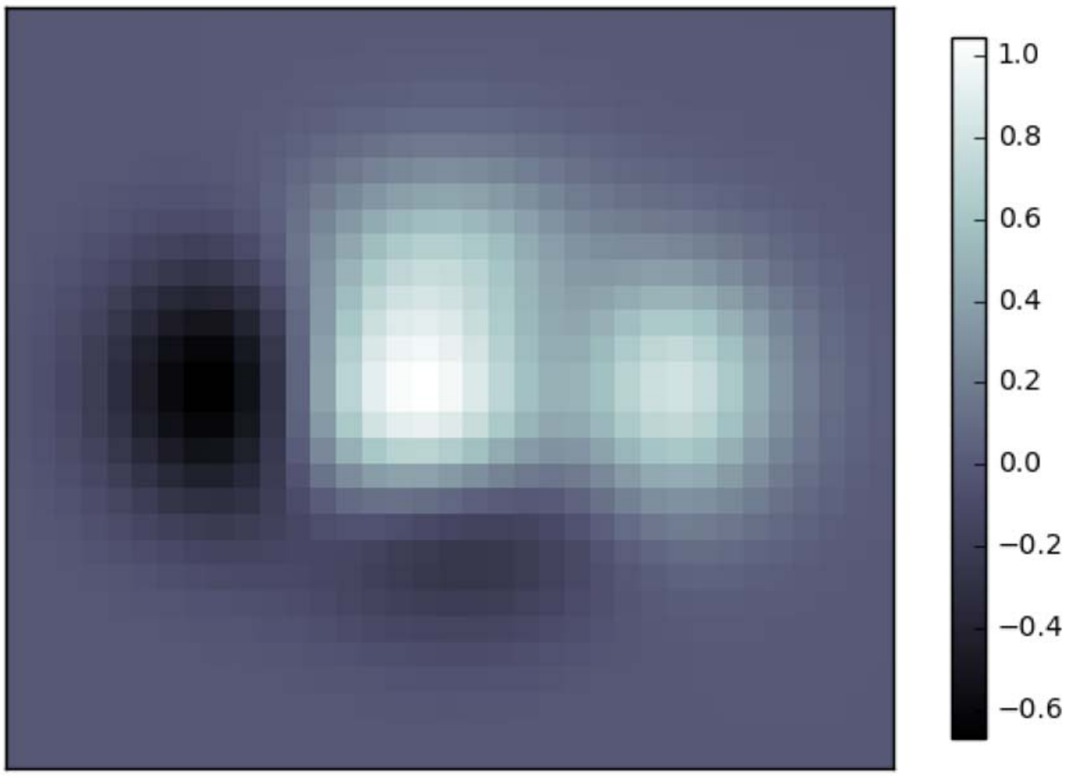
```
import numpy as np
import matplotlib.pyplot as plt

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()
```



Imshow

In [32]:

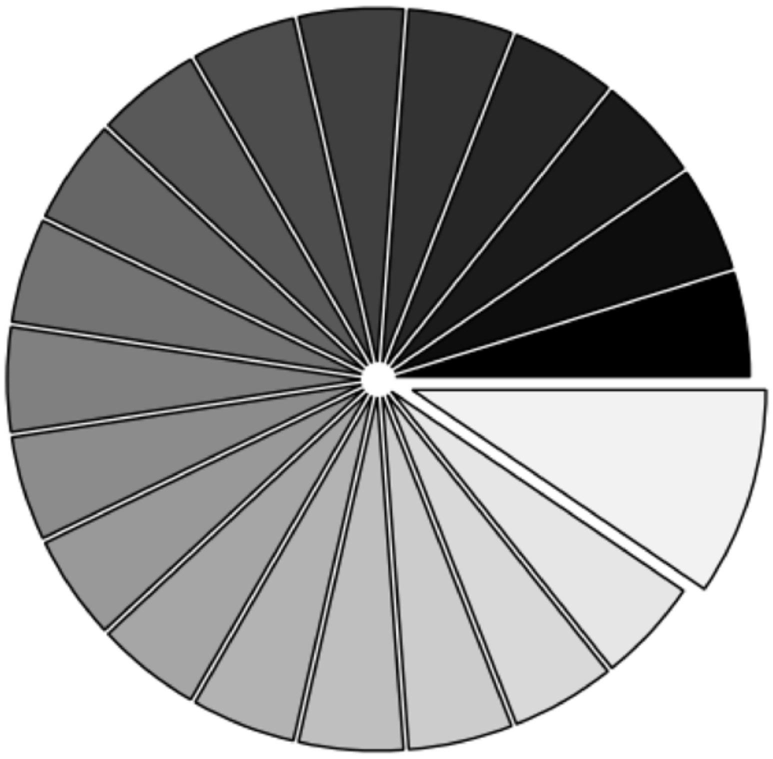
```
import numpy as np
import matplotlib.pyplot as plt

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

n = 10
x = np.linspace(-3, 3, 3.5 * n)
y = np.linspace(-3, 3, 3.0 * n)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.imshow(Z, interpolation='nearest', cmap='bone', origin='lower')
plt.colorbar(shrink=.92)

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



饼图

In [33]:

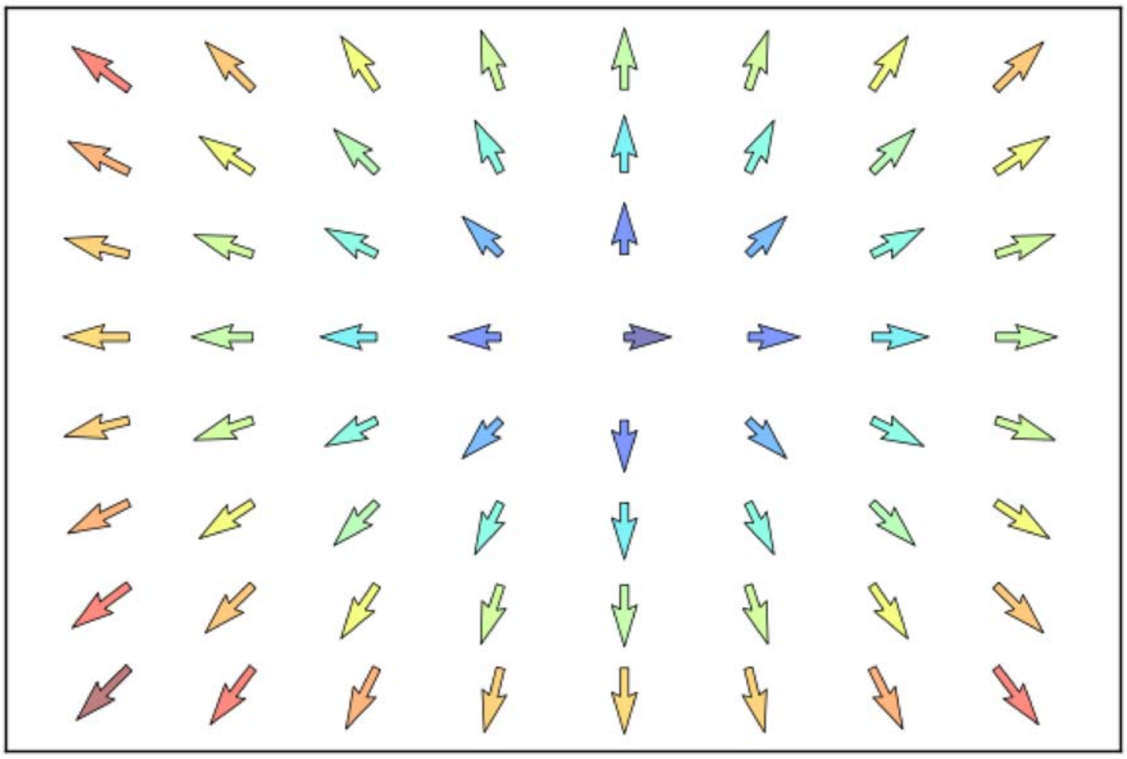
```
import numpy as np
import matplotlib.pyplot as plt

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(n)])
plt.axis('equal')
plt.xticks(())
plt.yticks()

plt.show()
```



箭头图

In [34]:

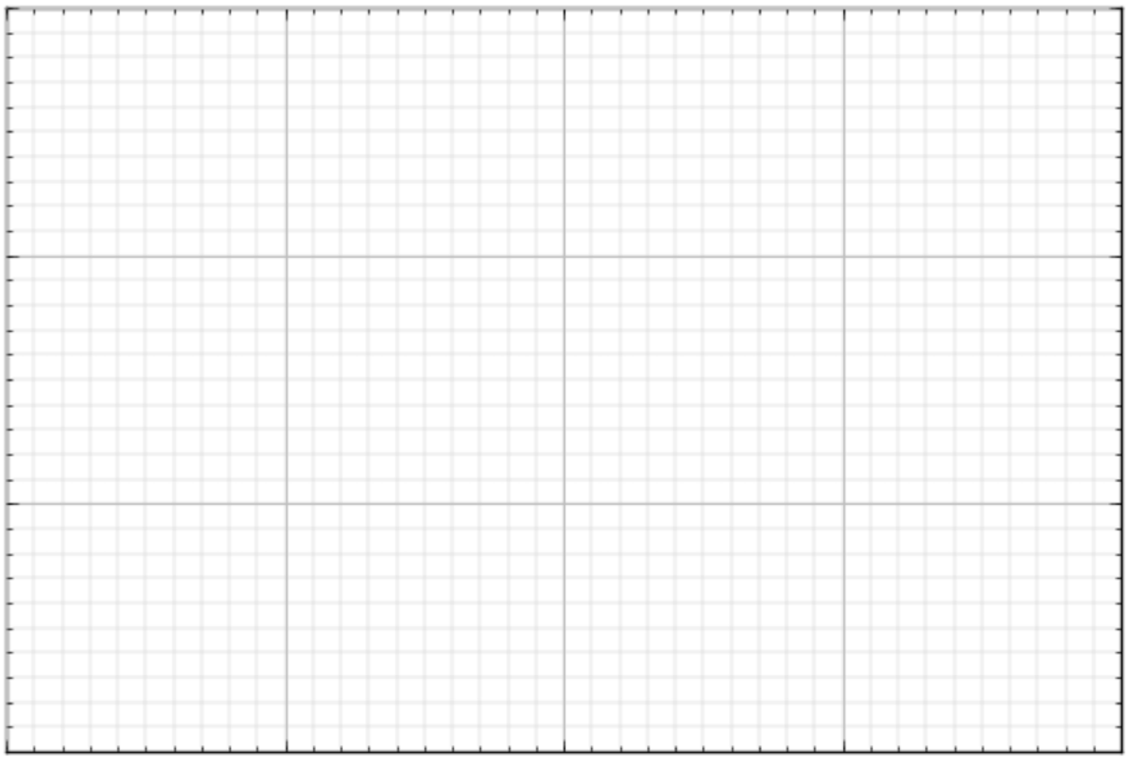
```
import numpy as np
import matplotlib.pyplot as plt

n = 8
X, Y = np.mgrid[0:n, 0:n]
T = np.arctan2(Y - n / 2., X - n/2.)
R = 10 + np.sqrt((Y - n / 2.0) ** 2 + (X - n / 2.0) ** 2)
U, V = R * np.cos(T), R * np.sin(T)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.quiver(X, Y, U, V, R, alpha=.5)
plt.quiver(X, Y, U, V, edgecolor='k', facecolor='None', linewidth=.5)

plt.xlim(-1, n)
plt.xticks(())
plt.ylim(-1, n)
plt.yticks(())

plt.show()
```



坐标网格

In [35]:

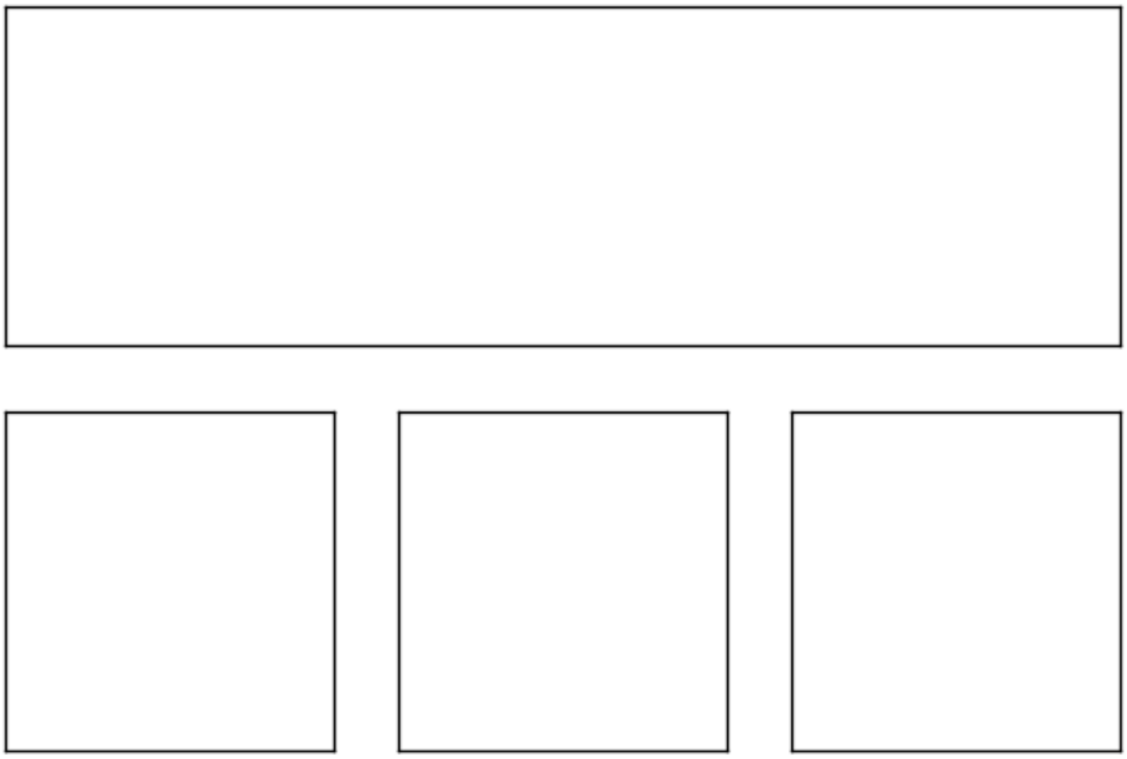
```
import matplotlib.pyplot as plt

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

ax.set_xlim(0,4)
ax.set_ylim(0,3)
ax.xaxis.set_major_locator(plt.MultipleLocator(1.0))
ax.xaxis.set_minor_locator(plt.MultipleLocator(0.1))
ax.yaxis.set_major_locator(plt.MultipleLocator(1.0))
ax.yaxis.set_minor_locator(plt.MultipleLocator(0.1))
ax.grid(which='major', axis='x', linewidth=0.75, linestyle='-', color='0.75')
ax.grid(which='minor', axis='x', linewidth=0.25, linestyle='-', color='0.75')
ax.grid(which='major', axis='y', linewidth=0.75, linestyle='-', color='0.75')
ax.grid(which='minor', axis='y', linewidth=0.25, linestyle='-', color='0.75')
ax.set_xticklabels([])
ax.set_yticklabels([])

plt.show()
```



分面多图

In [36]:

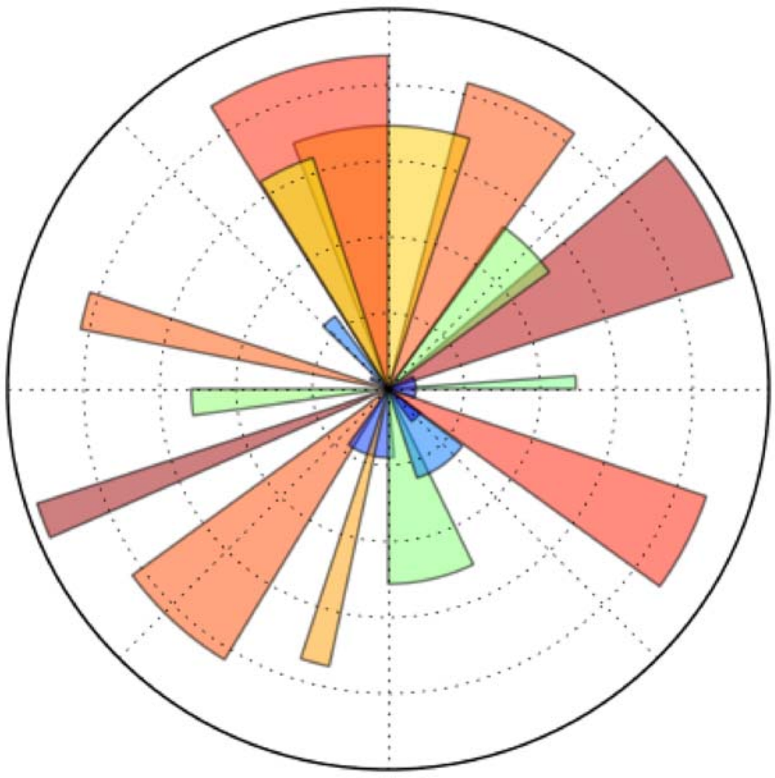
```
import matplotlib.pyplot as plt

fig = plt.figure()
fig.subplots_adjust(bottom=0.025, left=0.025, top = 0.975, right=0.975)

plt.subplot(2, 1, 1)
plt.xticks(()), plt.yticks(())

plt.subplot(2, 3, 4)
plt.xticks(())
plt.yticks(())

plt.subplot(2, 3, 5)
plt.xticks(())
plt.yticks(())

plt.subplot(2, 3, 6)
plt.xticks(())
plt.yticks(())

plt.show()
```



极坐标

In [37]:

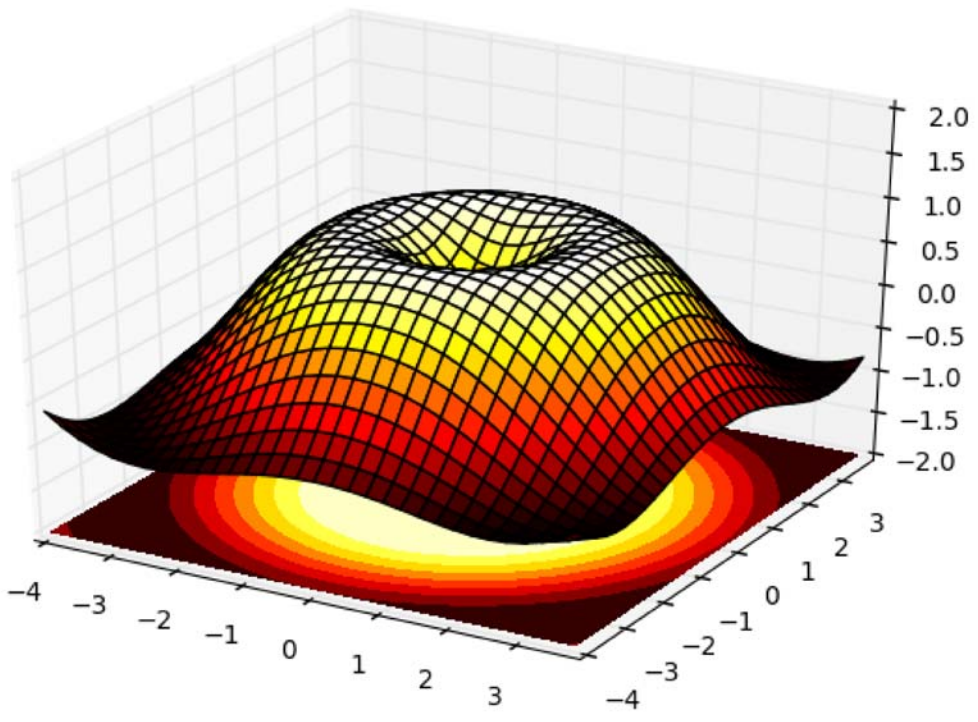
```
import numpy as np
import matplotlib.pyplot as plt

ax = plt.axes([0.025, 0.025, 0.95, 0.95], polar=True)

N = 20
theta = np.arange(0.0, 2 * np.pi, 2 * np.pi / N)
radii = 10 * np.random.rand(N)
width = np.pi / 4 * np.random.rand(N)
bars = plt.bar(theta, radii, width=width, bottom=0.0)

for r,bar in zip(radii, bars):
    bar.set_facecolor(plt.cm.jet(r/10.))
    bar.set_alpha(0.5)

ax.set_xticklabels([])
ax.set_yticklabels([])
plt.show()
```



三维图

In [38]:

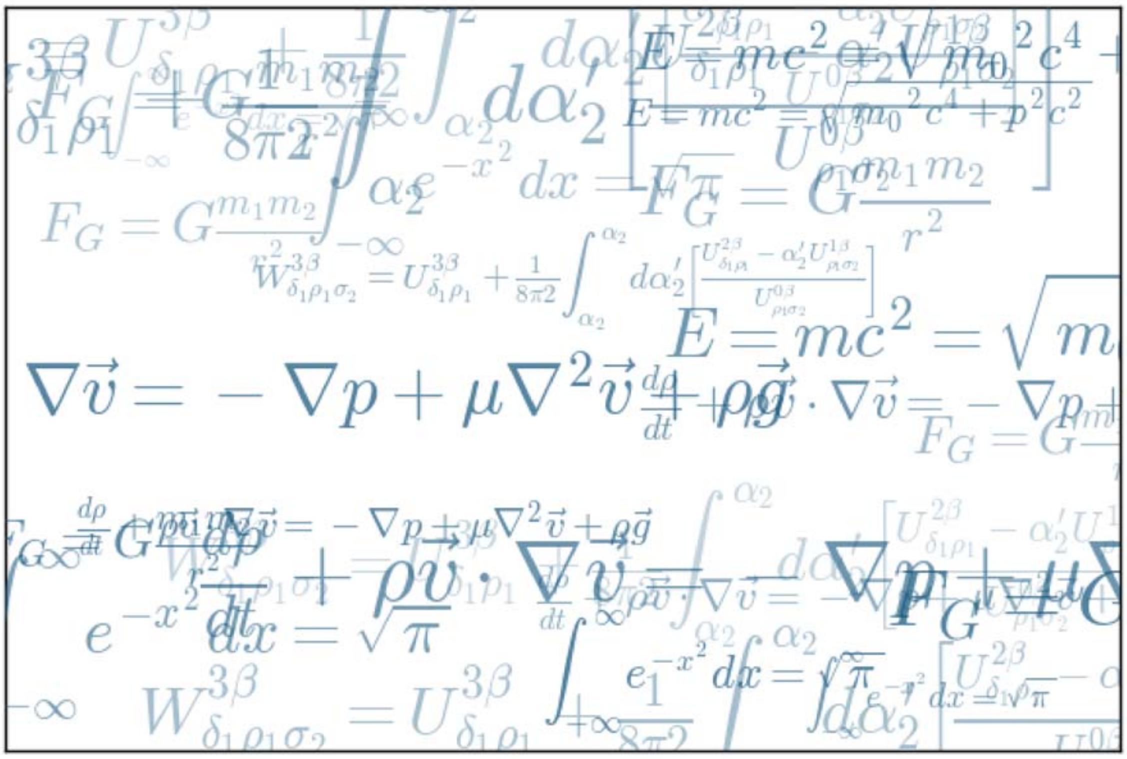
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

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()
```



文本

In [39]:

```
import numpy as np
import matplotlib.pyplot as plt

eqs = []
eqs.append((r"$W^{3\beta}_{\delta_1 \rho_1 \sigma_2} = U^{3\beta}_{\delta_1 \rho_1} + \frac{1}{8
 \pi 2} \int^{\alpha_2}_{\alpha_2} d \alpha^\prime_2 \left[\frac{ U^{2\beta}_{\delta_1 \rho_1} -
 \alpha^\prime_2U^{1\beta}_{\rho_1 \sigma_2} }{U^{0\beta}_{\rho_1 \sigma_2}}\right]$"))
eqs.append((r"$\frac{d\rho}{d t} + \rho \vec{v}\cdot\nabla\vec{v} = -\nabla p + \mu\nabla^2 \vec
{v} + \rho \vec{g}$"))
eqs.append((r"$\int_{-\infty}^\infty e^{-x^2}dx=\sqrt{\pi}$"))
eqs.append((r"$E = mc^2 = \sqrt{{m_0}^2c^4 + p^2c^2}$"))
eqs.append((r"$F_G = G\frac{m_1m_2}{r^2}$"))

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

for i in range(24):
    index = np.random.randint(0, len(eqs))
    eq = eqs[index]
    size = np.random.uniform(12, 32)
    x,y = np.random.uniform(0, 1, 2)
    alpha = np.random.uniform(0.25, .75)
    plt.text(x, y, eq, ha='center', va='center', color="#11557c", alpha=alpha,
         transform=plt.gca().transAxes, fontsize=size, clip_on=True)
plt.xticks(())
plt.yticks(())

plt.show()
```



以上就是Matplotlib绘图库的基本内容，个人认为应对一般的画图足够，但是与R的ggplot2相比，我还是更偏向于使用后者。但是，无论用哪个软件，用哪个包，做出好看的图形才是王道，一个1分钟可以用Photoshop或者windows自带画图软件解决的问题，个人认为没必要花半天时间浪费在ggplot2或者Matplotlib上。