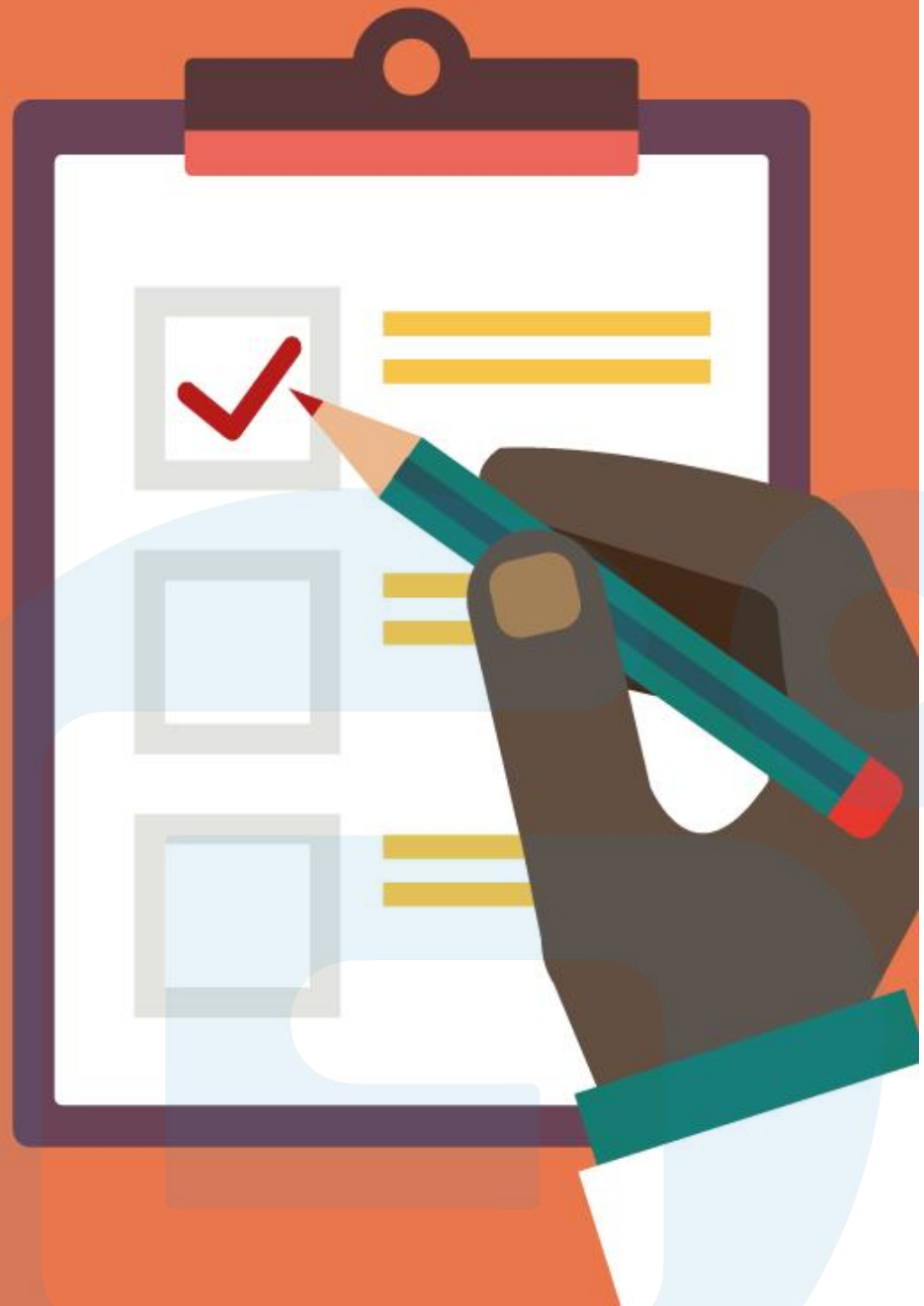
Talk is cheap, show me the code

# 第六课：matplotlib 基础

Python进阶课程系列



# OUTLINE

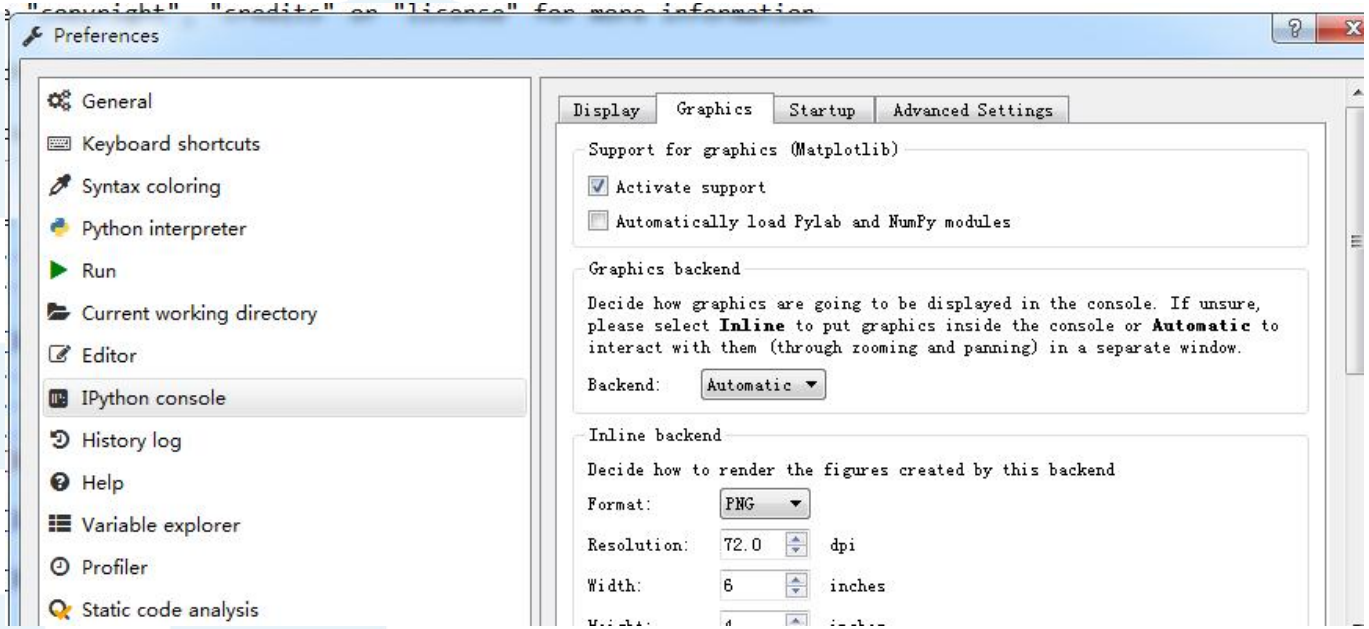
- matplotlib简介
- matplotlib的实例

# 一 matplotlib简介

# matplotlib简介

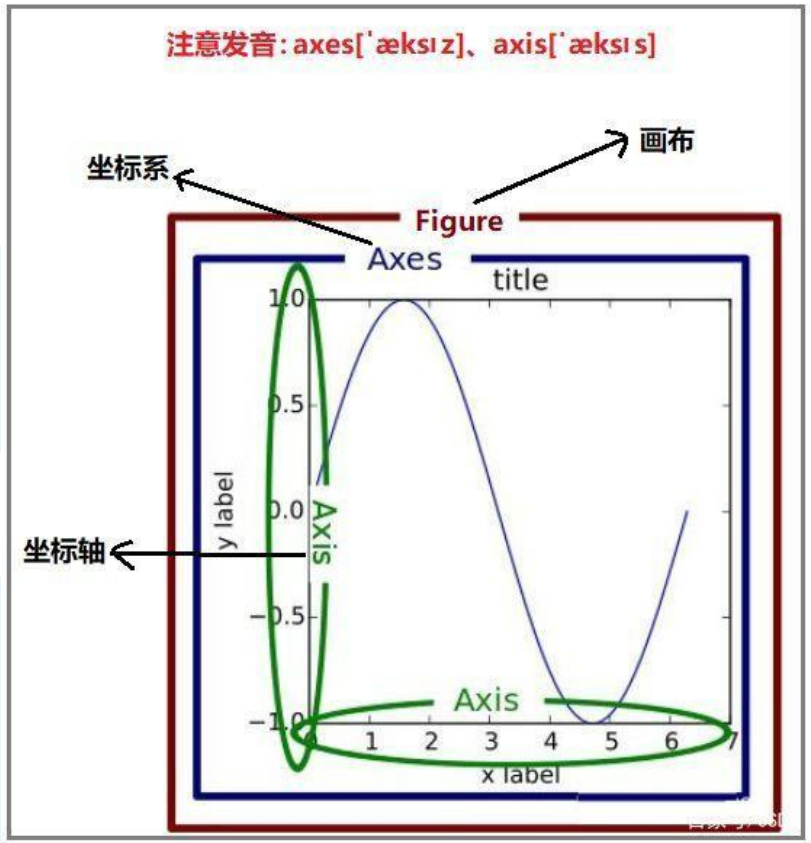
matplotlib是受MATLAB的启发构建的python绘图库。MATLAB是数据绘图领域广泛使用的语言和工具，matplotlib与matlab的很多语法和使用方法是类似的，但是它的开放性更好，而且是免费开源的。

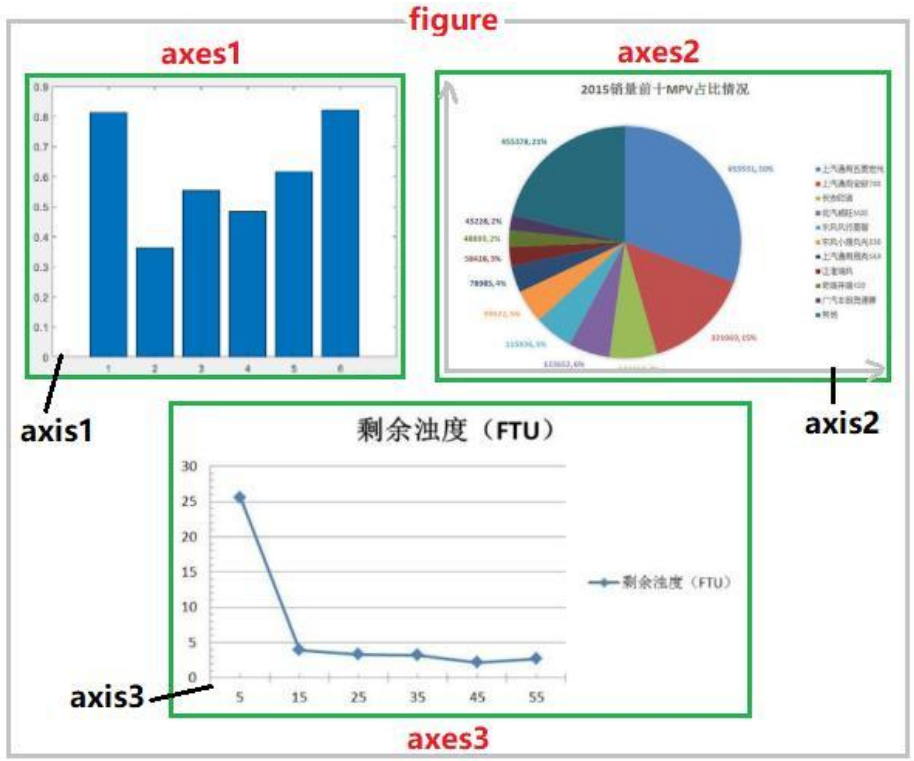
matplotlib和matlab绘图的特点是利用命令行语句来进行绘图，它和excel绘图或者origin绘图的一个主要区别在于可以精确控制图形上的每一个元素，此外它们是与数据处理界面集成的，无需额外打开别的软件。而且也很方便批量化的生成图形，不用一张一张去生成。



# matplotlib简介

**import matplotlib.pyplot as plt**
**pyplot是一个子包**

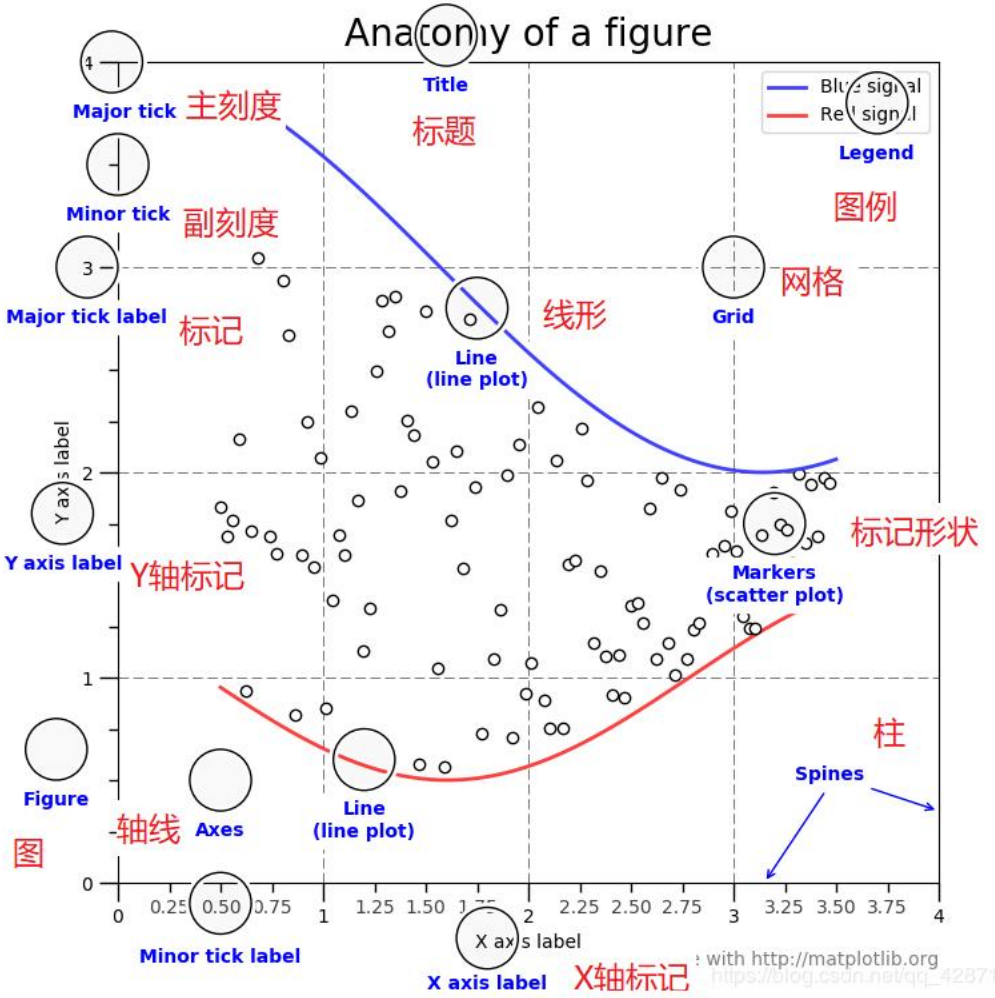



使用matplotlib绘图的原理，主要就是理解figure(画布)、axes(坐标系)、axis(坐标轴)三者之间的关系。

# matplotlib简介

- **axex:** 设置坐标轴边界和表面的颜色、坐标刻度值大小和网格的显示
- **figure:** 控制dpi、边界颜色、图形大小、和子区( subplot)设置
- **font:** 字体集（font family）、字体大小和样式设置
- **grid:** 设置网格颜色和线性
- **legend:** 设置图例和其中的文本的显示line: 设置线条（颜色、线型、宽度等）和标记
- **patch:** 是填充2D空间的图形对象，如多边形和圆。控制线宽、颜色和抗锯齿设置等。
- **savefig:** 可以对保存的图形进行单独设置。例如，设置渲染的文件的背景为白色。
- **verbose:** 设置matplotlib在执行期间信息输出，如silent、helpful、debug和debug-annoying。
- **xticks和yticks:** 为x,y轴的主刻度和次刻度设置颜色、大小、方向，以及标签大小。



# matplotlib简介

## 完整的绘图步骤

**① 导库**

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

**② 创建figure画布对象**

如果绘制一个简单的小图形，我们可以不设置figure对象，使用默认创建的figure对象，当然我们也可以显示创建figure对象。如果一张figure画布上，需要绘制多个图形。那么就必须显示的创建figure对象，然后得到每个位置上的axes对象，进行对应位置上的图形绘制。

**③ 根据figure对象进行布局设置**

1*11*22*12*2...

**④ 获取对应位置的axes坐标系对象**

```
figure = plt.figure()
axes1 = figure.add_subplot(2,1,1)
axes2 = figure.add_subplot(2,1,1)
```

**⑤ 调用axes对象，进行对应位置的图形绘制**

这一步，是我们传入数据，进行绘图的一步。对于图形的一些细节设置，都可以在这一步进行。

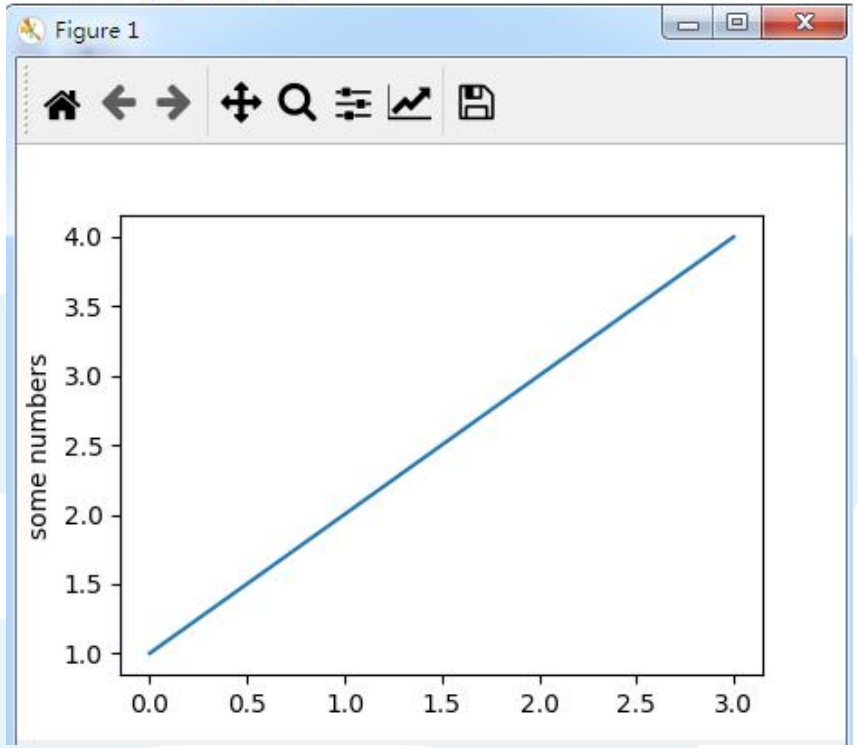
**⑥ 显示图形**

plt.show()或figure.show()

# 二 matplotlib的实例

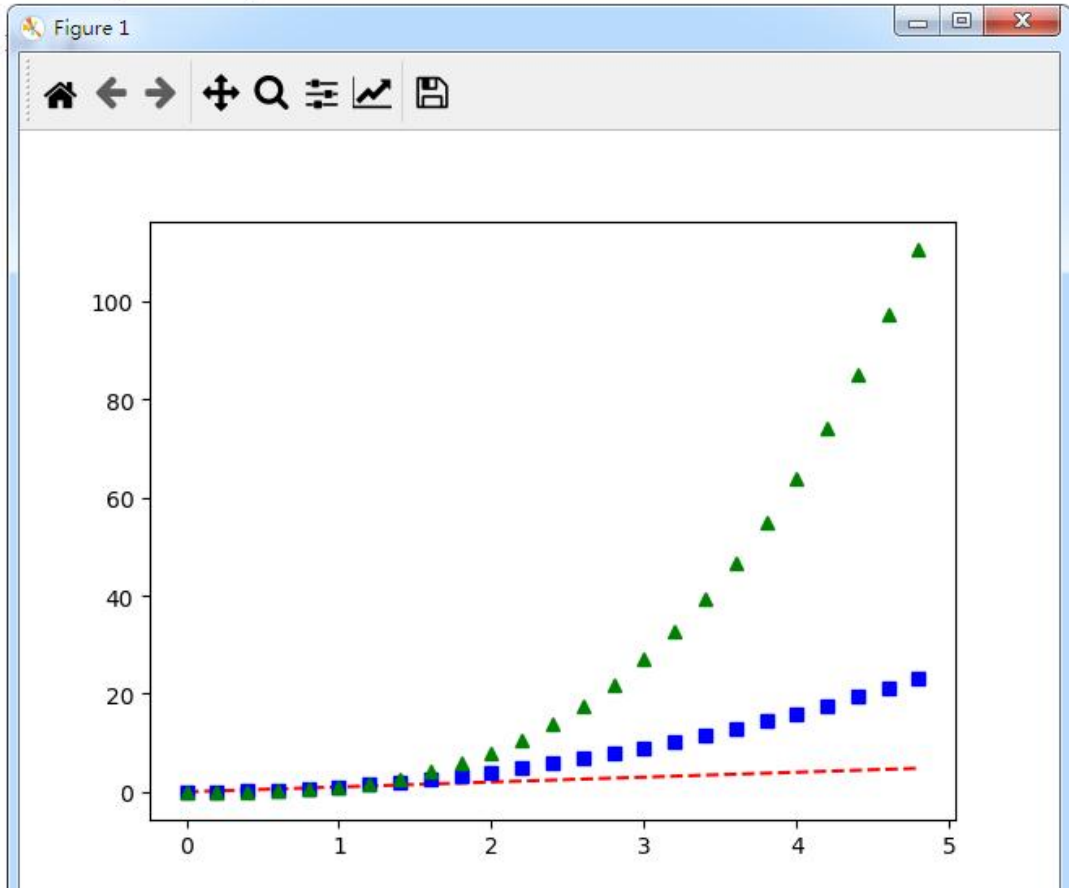
# matplotlib的实例

```
In [3]: import matplotlib.pyplot as plt
   ...: plt.plot([1,2,3,4])
   ...: plt.ylabel('some numbers')
   ...: plt.show()
```




如果向plot()命令提供单个列表或数组，则matplotlib假定它是一个y值序列，并自动生成x值。 由于 python 范围从 0 开始，默认x向量具有与y相同的长度，但从 0 开始。因此x数据是[0,1,2,3]。

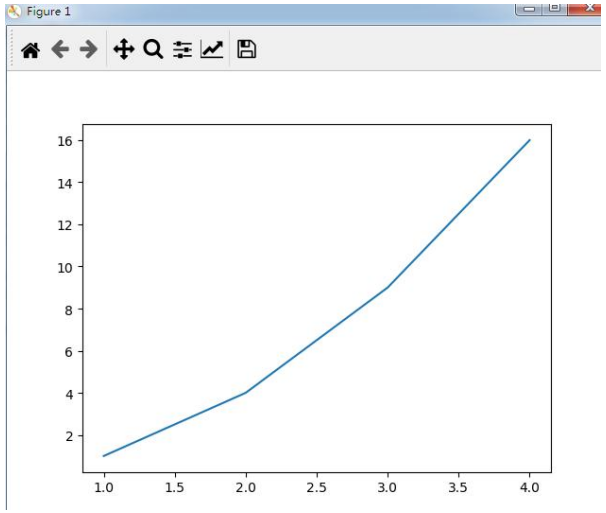
```
In [2]: import numpy as np
   ...: from matplotlib import pyplot as plt
   ...: t = np.arange(0., 5., 0.2)
   ...: plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
   ...: plt.show()
```



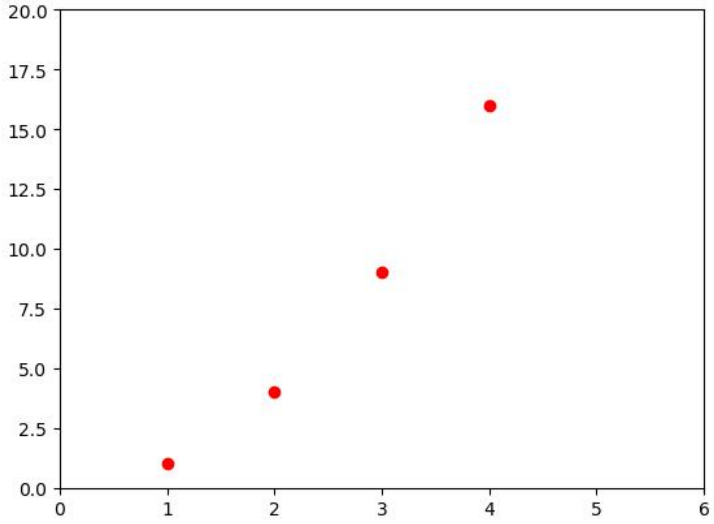
# matplotlib的实例

plot()是一个通用命令，并且可接受任意数量的参数。 例如，要绘制x和y，可以执行命令：

plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
对于每个x,y参数对，有一个可选的第三个参数，它是指示图形颜色和线条类型的格式字符串。 格式字符串的字母和符号来自 MATLAB，并且将颜色字符串与线型字符串连接在一起。 默认格式字符串为"b-"，它是一条蓝色实线。 例如，要绘制上面的红色圆圈，需要执行：




```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20]) #axis()命令接收[xmin，xmax，ymin，ymax]的列表，并指定轴域的可视区域。
plt.show()
```




注意，如果不关闭界面直接执行命令，则是在原图上增加圆点。

# 显示中文属性

默认的matplotlib不支持中文字体，matplotlib 默认使用的 font.family 是 sans-serif，即无衬线字体，可以看到在font.sans-serif中设置的全部为西文字体，如果要使用中文标题，需要做一些改动。

可以使用 rc 配置（rcParams）来自定义图形的各种默认属性

| 字体 | 代码 |
| --- | --- |
| 黑体 | SimHei |
| 仿宋 | FangSong |
| 楷体 | KaiTi |
| 微软雅黑体 | Microsoft YaHei |
| 宋体 | SimSun |

如果装了 office，那么还支持以下字体：

| 字体 | 代码 |
| --- | --- |
| 隶书 | LiSu |
| 幼圆 | YouYuan |
| 华文细黑 | STXihei |
| 华文楷体 | STKaiti |
| 华文宋体 | STSong |
| 华文中宋 | STZhongsong |
| 华文仿宋 | STFangsong |
| 方正舒体 | FZShuTi |
| 方正姚体 | FZYaoti |
| 华文彩云 | STCaiyun |
| 华文琥珀 | STHupo |
| 华文隶书 | STLiti |
| 华文行楷 | STXingkai |
| 华文新魏 | STXinwei |

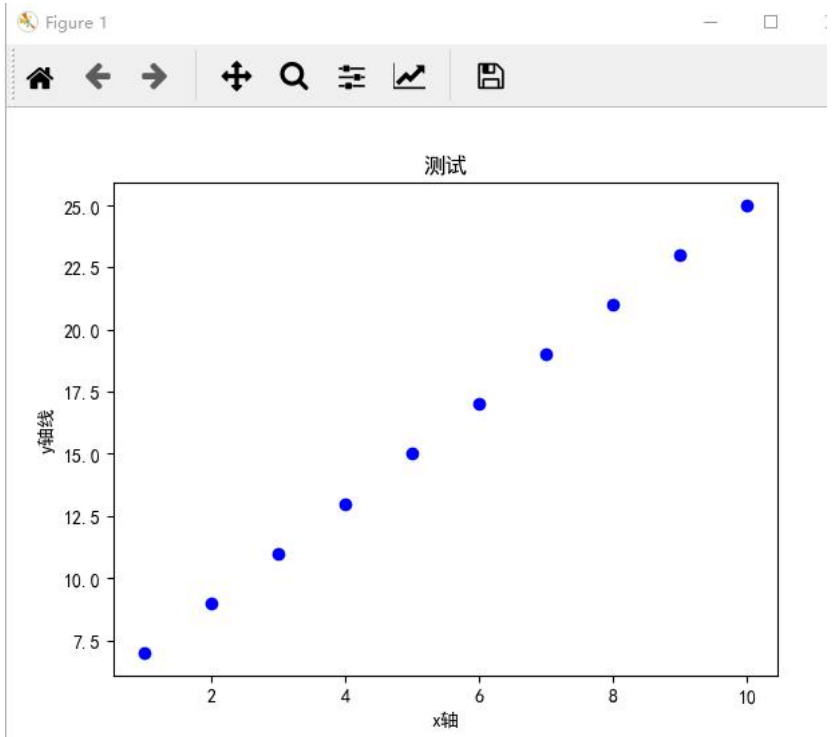
# matplotlib的实例

## 显示中文属性

重载配置文件

mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['font.serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
# 解决保存图像是负号'-'显示为方块的问题,或者转换负号为字符串

```
In [4]:

In [4]: import numpy as np
   ...: import matplotlib as mpl
   ...: from matplotlib import pyplot as plt
   ...:
   ...: mpl.rcParams['font.sans-serif'] = ['SimHei']
   ...: mpl.rcParams['font.serif'] = ['SimHei']
   ...: mpl.rcParams['axes.unicode_minus'] = False
   ...:
   ...: x = np.arange(1,11)
   ...: y =  2  * x +  5
   ...: plt.title("测试")
   ...: plt.xlabel("x轴")
   ...: plt.ylabel("y轴线")
   ...: plt.plot(x,y,"ob")
   ...: plt.show()

In [5]:
```



# 显示中文属性

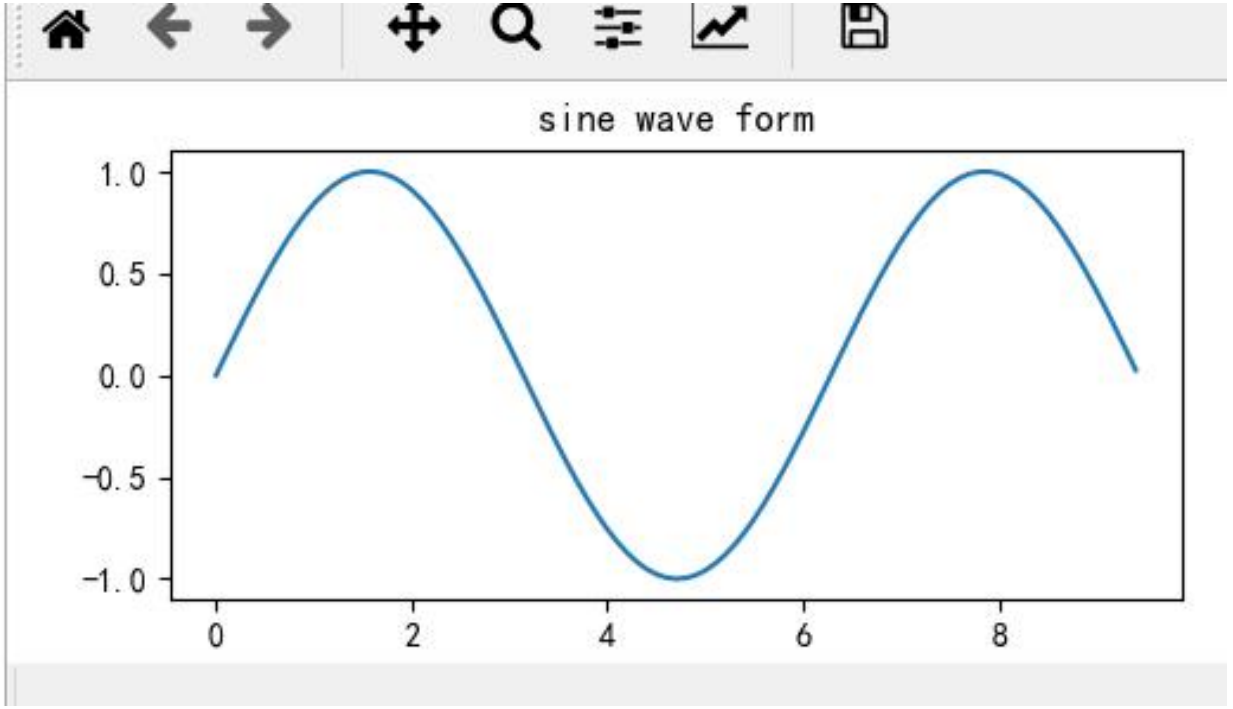
自定义字体

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

# fname 为 你下载的字体库路径，注意字体的路径
zhfont1 = matplotlib.font_manager.FontProperties(fname=r'D:\Fonts\simkai.ttf')

x = np.arange(1,11)
y =  2  * x +  5
plt.title("测试", fontproperties=zhfont1)

# fontproperties 设置中文显示，fontsize 设置字体大小
plt.xlabel("x 轴", fontproperties=zhfont1,fontsize=24)
plt.ylabel("y 轴", fontproperties=zhfont1,fontsize=24)
plt.plot(x,y)
plt.show()
```

# matplotlib的实例

## 生成正弦图

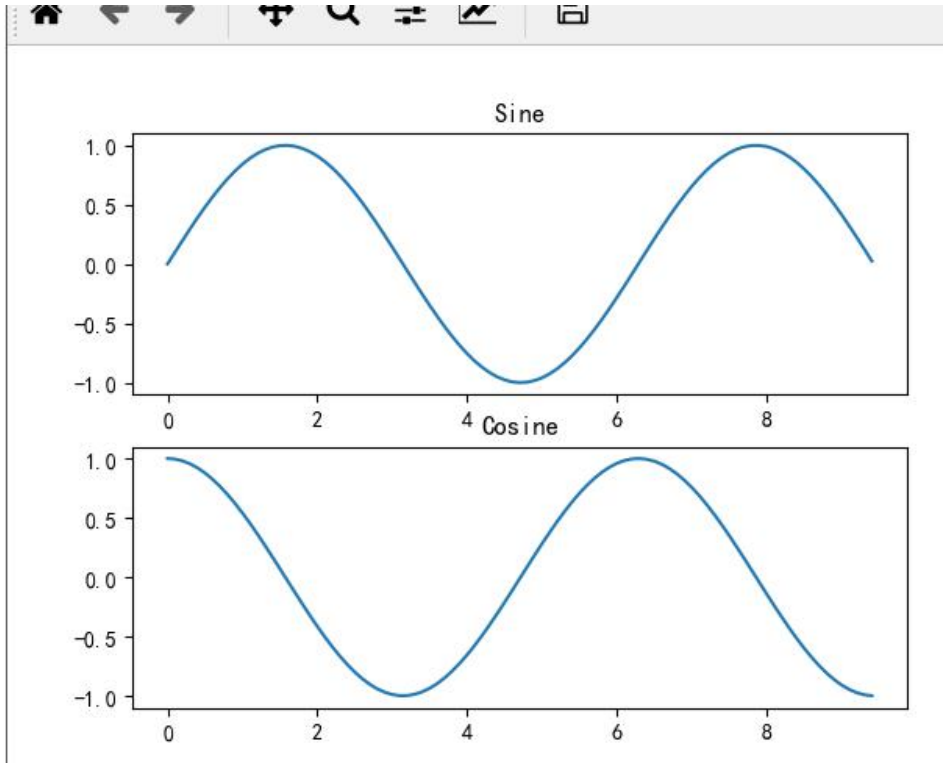
```
In [6]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: # 计算正弦曲线上点的 x 和 y 坐标
   ...: x = np.arange(0,3*np.pi,0.1)
   ...: y = np.sin(x)
   ...: plt.title("sine wave form")
   ...: # 使用 matplotlib 来绘制点
   ...: plt.plot(x, y)
   ...: plt.show()

In [7]:
```



# 生成正弦图+余弦图

subplot() 函数允许在同一图中绘制不同的东西，subplot(2,3,1)是指一个2行3列的图中从左到右从上到下的第一个位置

```
In [7]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: # 计算正弦和余弦曲线上的点的 x 和 y 坐标
   ...: x = np.arange(0,  3  * np.pi,  0.1)
   ...: y_sin = np.sin(x)
   ...: y_cos = np.cos(x)
   ...: # 建立 subplot 网格，高为 2，宽为 1
   ...: # 激活第一个 subplot
   ...: plt.subplot(2,  1,  1)
   ...: # 绘制第一个图像
   ...: plt.plot(x, y_sin)
   ...: plt.title('Sine')
   ...: # 将第二个 subplot 激活，并绘制第二个图像
   ...: plt.subplot(2,  1,  2)
   ...: plt.plot(x, y_cos)
   ...: plt.title('Cosine')
   ...: # 展示图像
   ...: plt.show()

In [8]:
```

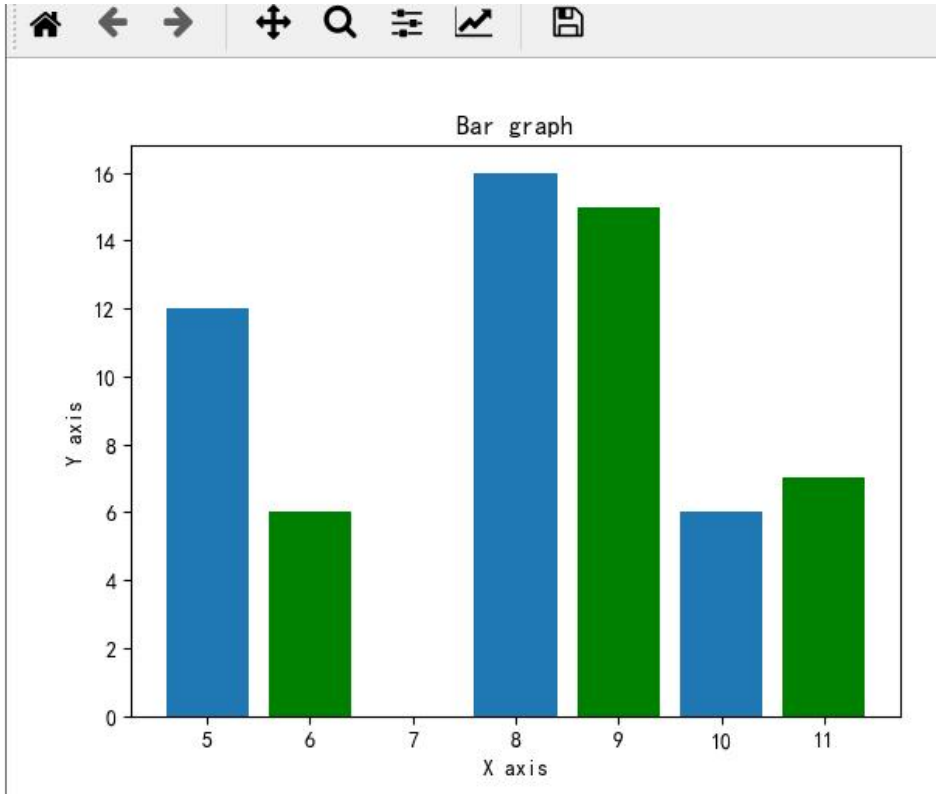


# matplotlib的实例

## 生成条状图

subplot() 函数允许在同一图中绘制不同的东西，subplot(2,3,1)是指一个2行3列的图中从左到右从上到下的第一个位置

```
...: # 展示图像
...: plt.show()

In [8]: from matplotlib import pyplot as plt
...: x =  [5,8,10]
...: y =  [12,16,6]
...: x2 =  [6,9,11]
...: y2 =  [6,15,7]
...: plt.bar(x, y, align =  'center')
...: plt.bar(x2, y2, color =  'g', align =  'center')
...: plt.title('Bar graph')
...: plt.ylabel('Y axis')
...: plt.xlabel('X axis')
...: plt.show()

In [9]:
```

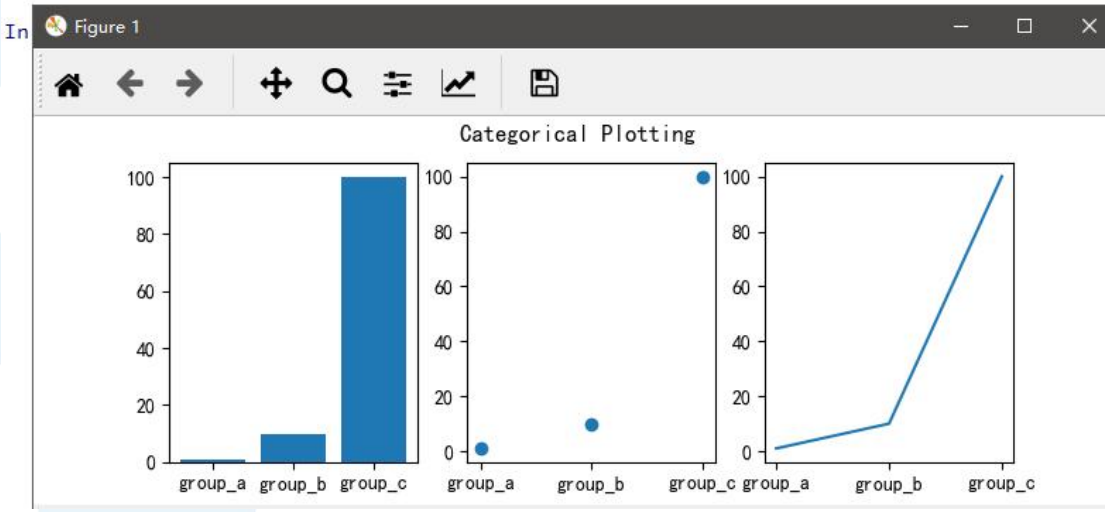


# matplotlib的实例

## 生成条状图

也可以利用字符作为横坐标

```
In [11]: names = ['group_a', 'group_b', 'group_c']
    ...: values = [1, 10, 100]
    ...:
    ...: plt.figure(1, figsize=(9, 3))
    ...:
    ...: plt.subplot(131)
    ...: plt.bar(names, values)
    ...: plt.subplot(132)
    ...: plt.scatter(names, values)
    ...: plt.subplot(133)
    ...: plt.plot(names, values)
    ...: plt.suptitle('Categorical Plotting')
    ...: plt.show()
```



# 式样美化

使用matplotlib自带的几种美化样式，就可以很轻松的对生成的图形进行美化。

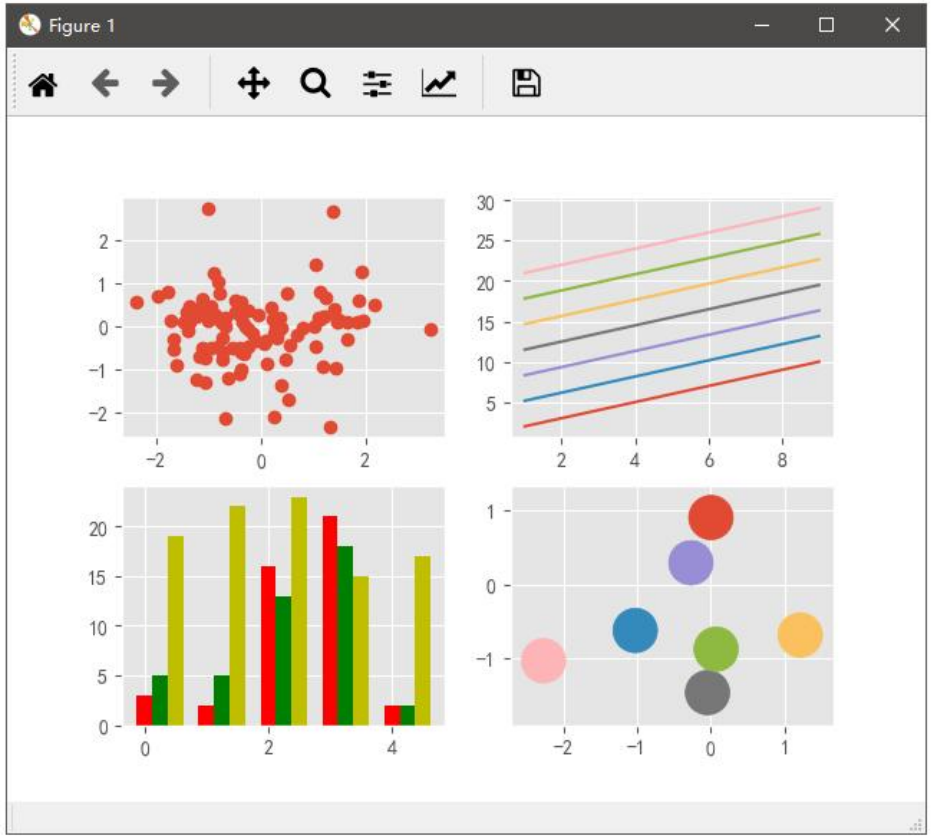
可以使用matplotlib.pyplot.style.available获取所有的美化样式

```
In [14]: print (plt.style.available)
['bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-
bright', 'seaborn-colorblind', 'seaborn-dark-palette', 'seaborn-dark', 'seaborn-darkgrid', 'seaborn-
deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster',
'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'seaborn', 'Solarize_Light2',
'tableau-colorblind10', '_classic_test']
```

# matplotlib的实例

ggplot，（模仿R语言的一个的流行绘图包）

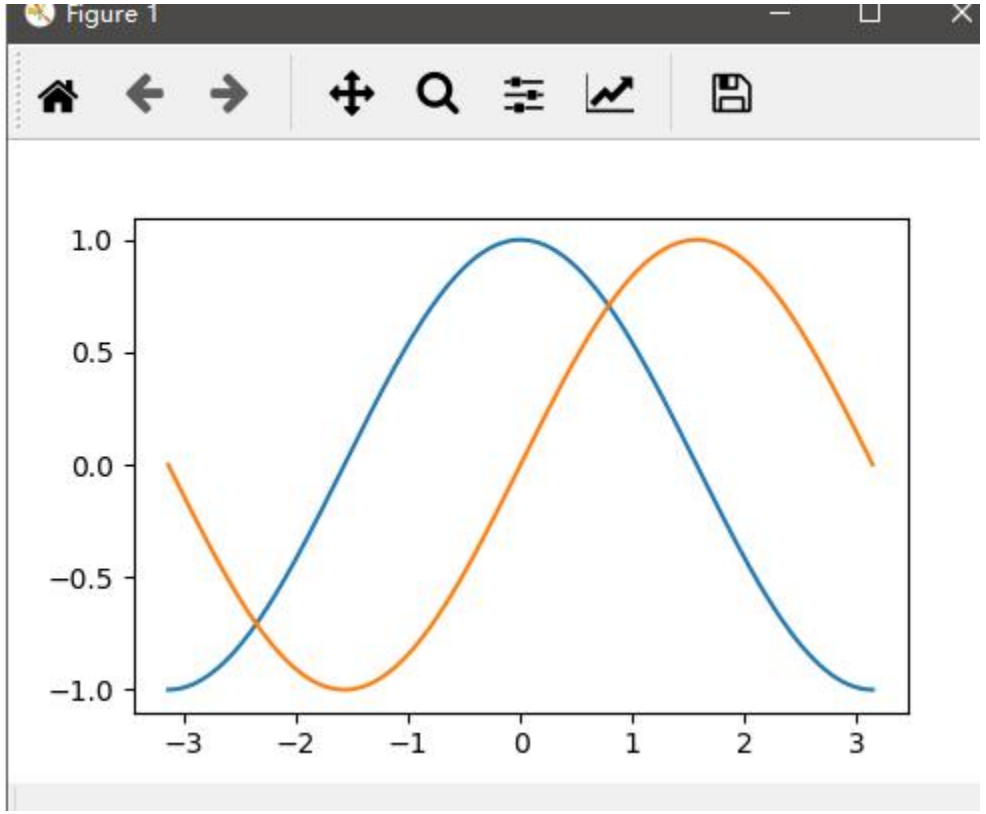
```
plt.style.use("ggplot")# 使用自带的样式进行美化
fig, axes = plt.subplots(ncols = 2, nrows = 2)

ax1, ax2, ax3, ax4 = axes.ravel() # 四个子图的坐标轴赋予四个对象

x, y = np.random.normal(size = (2, 100))
ax1.plot(x, y, "o")

x = np.arange(1, 10)
y = np.arange(1, 10)

# plt.rcParams['axes.prop_cycle']获取颜色的字典
ncolors = len(plt.rcParams['axes.prop_cycle'])
shift = np.linspace(1, 20, ncolors)
for s in shift:
    # print s
    ax2.plot(x, y + s, "-")

x = np.arange(5)
y1, y2, y3 = np.random.randint(1, 25, size = (3, 5))
width = 0.25

# 柱状图中要显式的指定颜色
ax3.bar(x, y1, width, color = "r")
ax3.bar(x + width, y2, width, color = "g")
ax3.bar(x + 2 * width, y3, width, color = "y")

for i, color in enumerate(plt.rcParams['axes.prop_cycle']):
    xy = np.random.normal(size= 2)
    for c in color.values():
        ax4.add_patch(plt.Circle(xy, radius = 0.3, color= c))

ax4.axis("equal")
plt.show()
```



# matplotlib的实例

## 一个完整的范例

```
In [1]: 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)
   ...:
   ...: plt.plot(X,C)
   ...: plt.plot(X,S)
   ...:
   ...: plt.show()

In [2]:
```



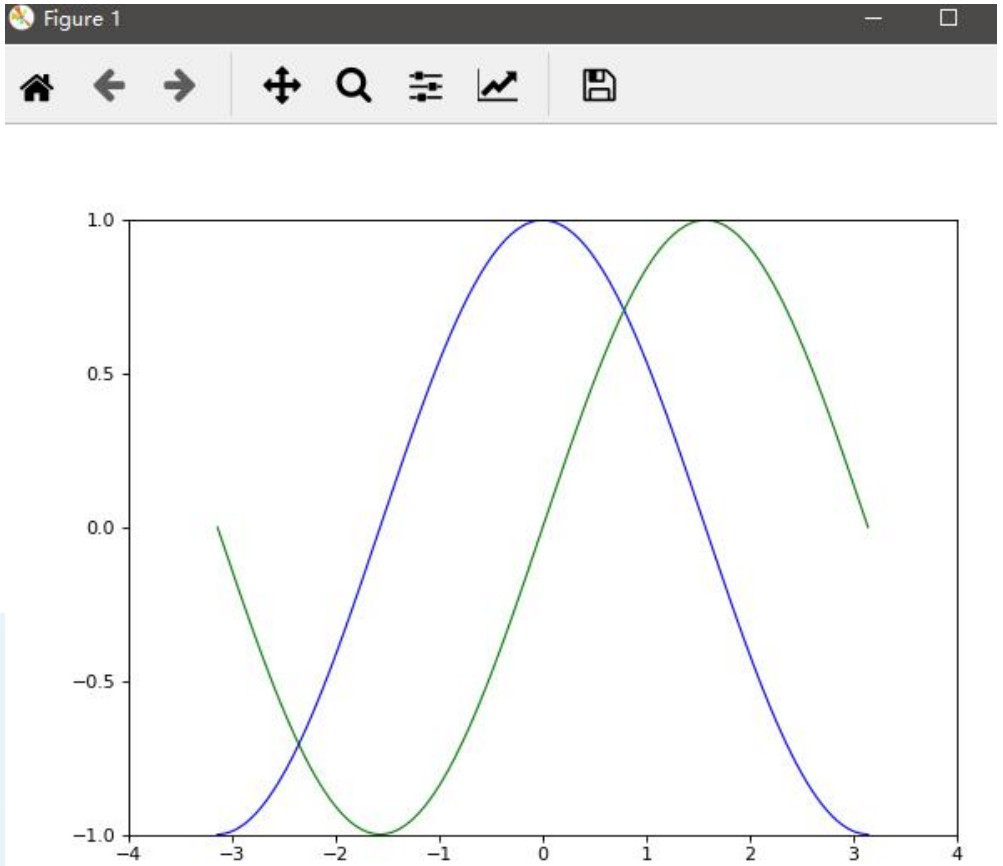
# matplotlib的实例

下面的代码中，展现了 matplotlib 的默认配置并辅以注释说明，这部分配置包含了有关绘图样式的所有配置。代码中的配置与默认配置完全相同，你可以在交互模式中修改其中的值来观察效果。

```
# 导入 matplotlib 的所有内容（nympy 可以用 np 这个名字来使用）
from matplotlib.pyplot import *

# 创建一个 8 * 6 点（point）的图，并设置分辨率为 80
figure(figsize=(8,6), dpi=80)

# 创建一个新的 1 * 1 的子图，接下来的图样绘制在其中的第 1 块（也是唯一的一块）
subplot(1,1,1)

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

# 绘制余弦曲线，使用蓝色的、连续的、宽度为 1 （像素）的线条
plot(X, C, color="blue", linewidth=1.0, linestyle="-")

# 绘制正弦曲线，使用绿色的、连续的、宽度为 1 （像素）的线条
plot(X, S, color="green", linewidth=1.0, linestyle="-")

# 设置横轴的上下限
xlim(-4.0,4.0)

# 设置横轴记号
xticks(np.linspace(-4,4,9,endpoint=True))

# 设置纵轴的上下限
ylim(-1.0,1.0)

# 设置纵轴记号
yticks(np.linspace(-1,1,5,endpoint=True))

# 以分辨率 72 来保存图片
# savefig("exercice_2.png",dpi=72)

# 在屏幕上显示
show()
```

# matplotlib的实例

下面的代码中，展现了 matplotlib 的默认配置并辅以注释说明，这部分配置包含了有关绘图样式的所有配置。代码中的配置与默认配置完全相同，你可以在交互模式中修改其中的值来观察效果。

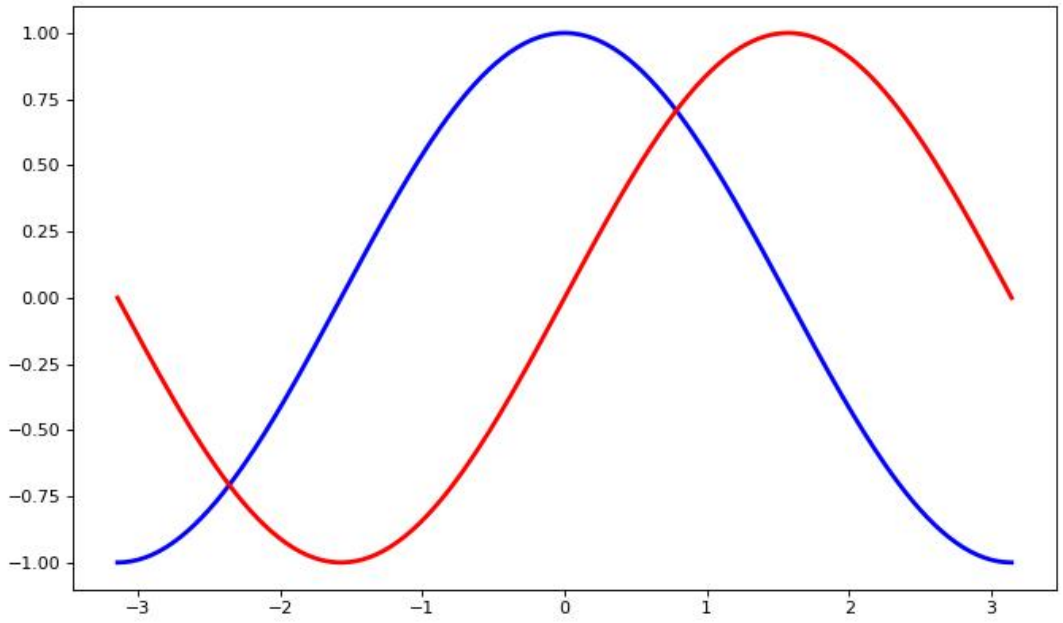


# matplotlib的实例

## 改变线条的颜色和粗细

首先，我们以蓝色和红色分别表示余弦和正弦函数，而后将线条变粗一点。接下来，我们在水平方向拉伸一下整个图。

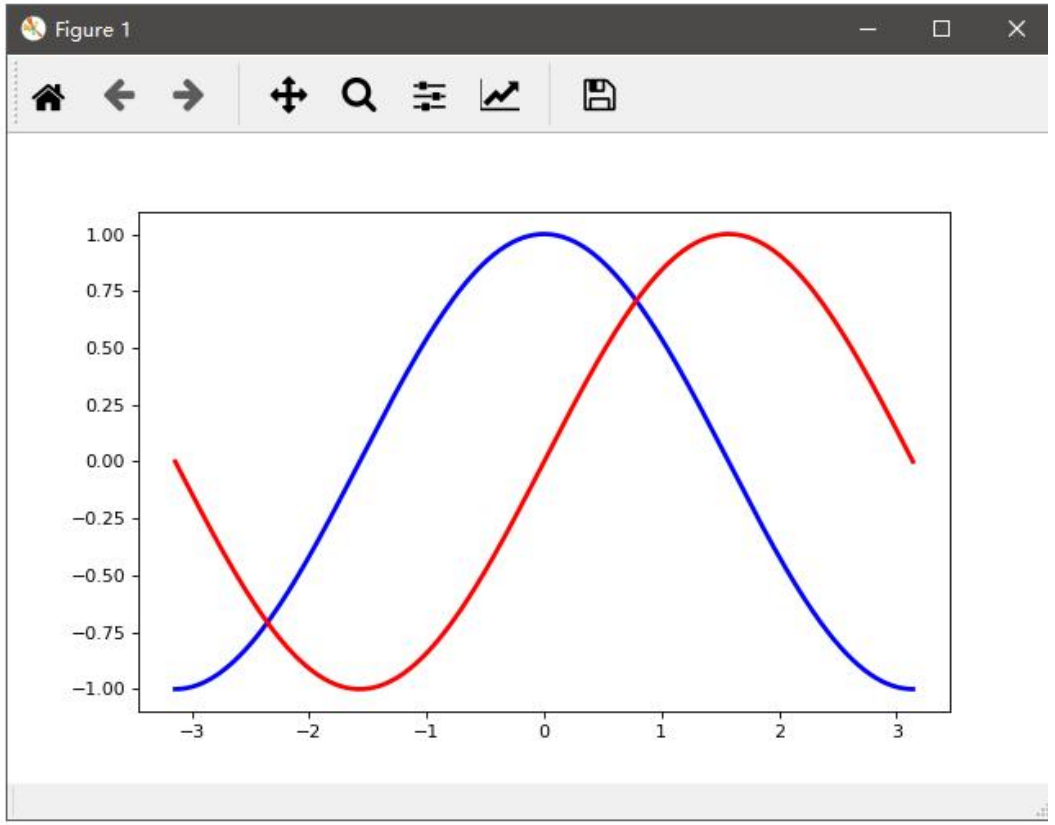
```
figure(figsize=(10,6), dpi=80)
plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plot(X, S, color="red",  linewidth=2.5, linestyle="-")
```



# matplotlib的实例

## 设置图片边界

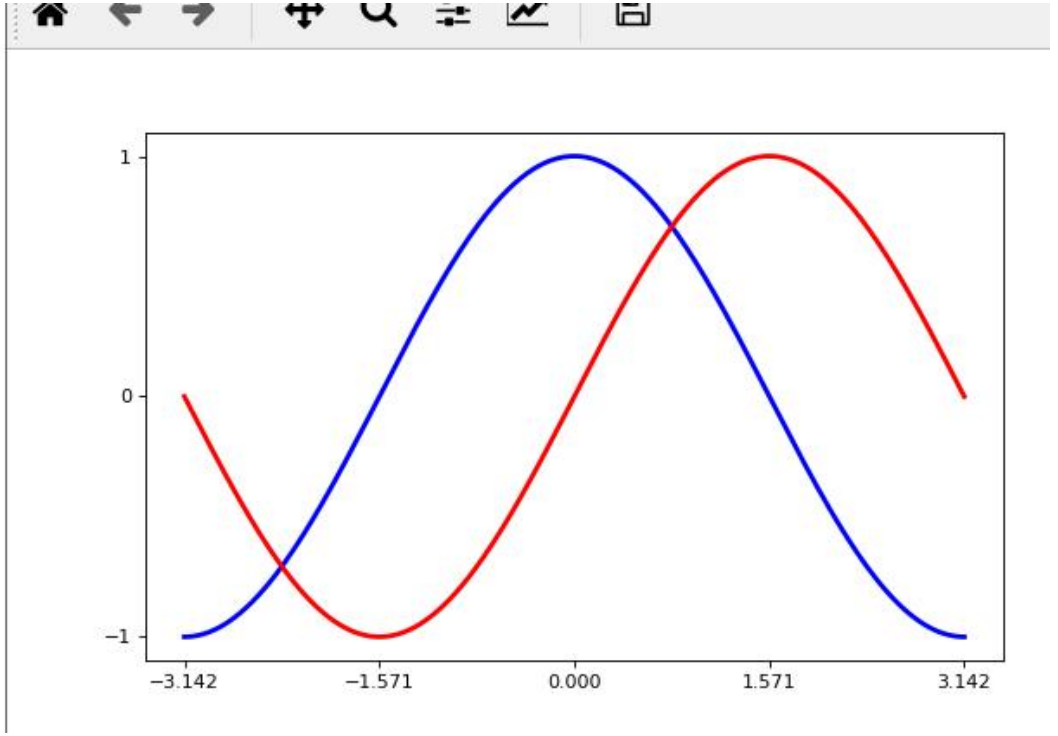
```
In [4]:

In [4]:
   ...: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...:
   ...: plt.figure(figsize=(8,5), dpi=80)
   ...: plt.subplot(111)
   ...:
   ...: X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
   ...: C,S = np.cos(X), np.sin(X)
   ...:
   ...: 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.show()

In [5]:
```



# matplotlib的实例

## 设置坐标轴上的记号

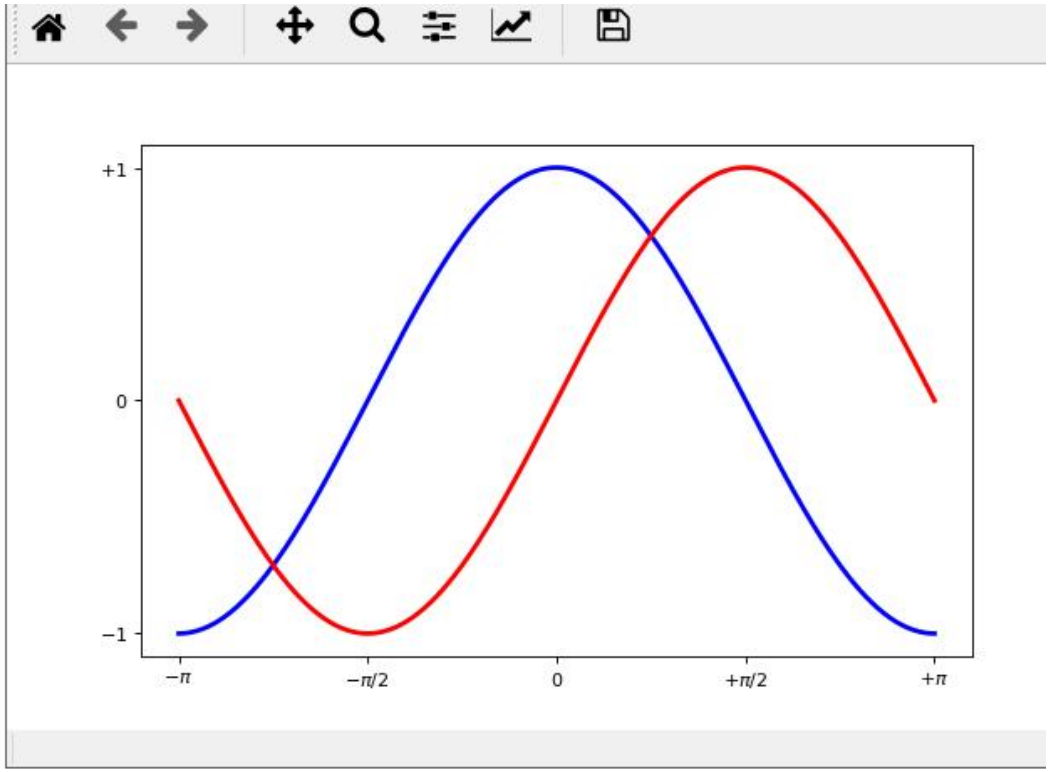
```
In [5]:
   ...: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...: plt.figure(figsize=(8,5), dpi=80)
   ...: plt.subplot(111)
   ...:
   ...: X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
   ...: C,S = np.cos(X), np.sin(X)
   ...:
   ...: plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
   ...: plt.plot(X, S, color="red", linewidth=2.5, linestyle="-")
   ...:
   ...: plt.xlim(X.min()*1.1, X.max()*1.1)
   ...: plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
   ...:
   ...: plt.ylim(C.min()*1.1,C.max()*1.1)
   ...: plt.yticks([-1, 0, +1])
   ...:
   ...: plt.show()

In [6]:
```



# matplotlib的实例

## 设置坐标轴上的记号标签

```
   ...: plt.show()

In [6]:
   ...: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...: plt.figure(figsize=(8,5), dpi=80)
   ...: plt.subplot(111)
   ...:
   ...: X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
   ...: C,S = np.cos(X), np.sin(X)
   ...:
   ...: 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.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.ylim(C.min()*1.1,C.max()*1.1)
   ...: plt.yticks([-1, 0, +1],
   ...:        [r'$-1$', r'$0$', r'$+1$'])
   ...:
   ...: plt.show()
```



# matplotlib的实例

## 设置坐标轴上的记号标签

Matplotlib中支持LaTex语法，输入格式为：r'$\Delta$' #其中的Delta对应于希腊字母的Δ
r'$\Delta$rv' #对应于Δrv

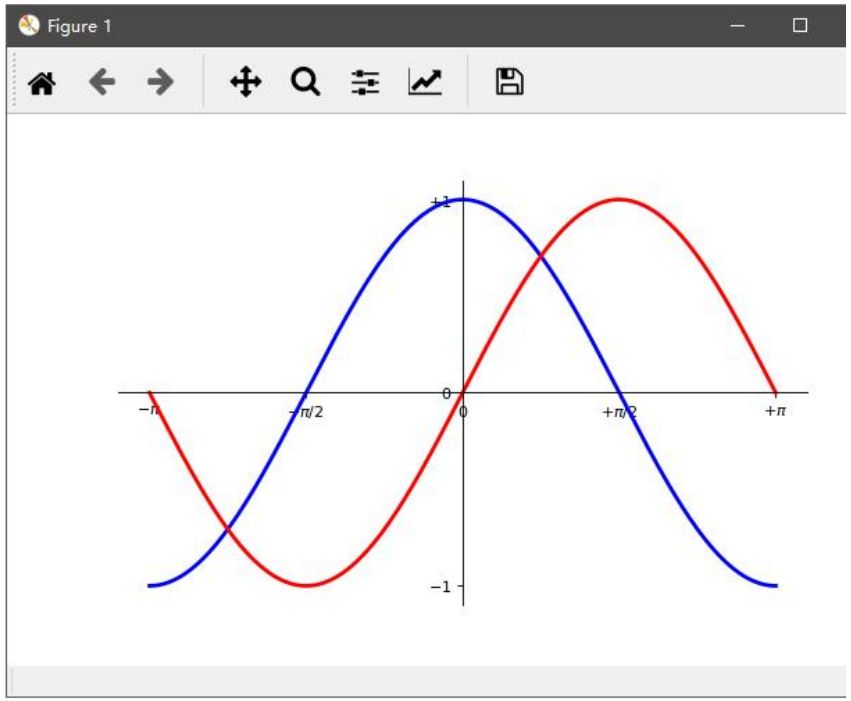
| 希腊字母小写、大写 | LaTeX形式 | 希腊字母小写、大写 | LaTeX形式 |
|---|---|---|---|
| α A | \alpha A | μ N | \nu N |
| β B | \beta B | ξ | \xi \Xi |
| γ Γ | \gamma \Gamma | o O | o O |
| δ Δ | \delta \ Delta | π Π | \pi \Pi |
| ϵ ε E | \epsilon \varepsilon E | ρ ϱ P | \rho \varrho P |
| ζ Z | \zeta Z | σ Σ | \sigma \Sigma |
| η H | \eta H | τ T | \tau T |
| θ ϑ Θ | \theta \vartheta \Theta | υ Y | \upsilon \Upsilon |
| ι I | \iota I | ϕ φ Φ | \phi \varphi \Phi |
| κ K | \kappa K | χ X | \chi X |
| λ Λ | \lambda \Lambda | ψ Ψ | \psi \Psi |
| μ M | \mu M | ω Ω | \omega \Omega |

# matplotlib的实例

## 移动Spines

坐标轴线和上面的记号连在一起就形成了脊柱（Spines），它记录了数据区域的范围。它们可以放在任意位置，不过至今为止，我们都把它放在图的四边。实际上每幅图有四条脊柱（上下左右），为了将脊柱放在图的中间，我们必须将其中的两条（上和右）设置为无色，然后调整剩下的两条到合适的位置——数据空间的 0 点。

```
In [7]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...: plt.figure(figsize=(8,5), dpi=80)
   ...: ax = plt.subplot(111)
   ...:
   ...: 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))
   ...:
   ...: X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
   ...: C,S = np.cos(X), np.sin(X)
   ...:
   ...: 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.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.ylim(C.min()*1.1,C.max()*1.1)
   ...: plt.yticks([-1, 0, +1],
   ...:        [r'$-1$', r'$0$', r'$+1$'])
   ...:
   ...: plt.show()
```

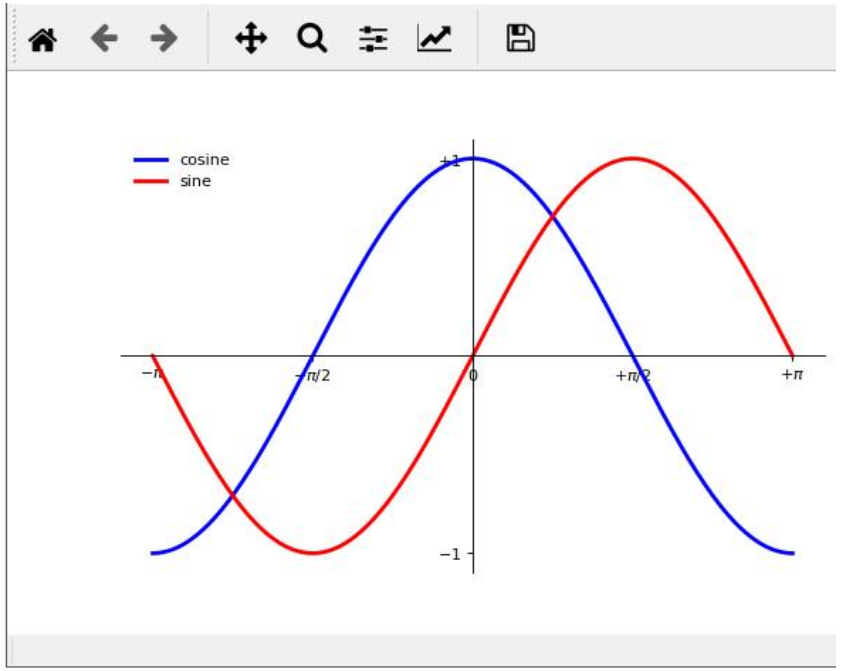


# matplotlib的实例

## 添加图例

只需要在 plot 函数里以「键 - 值」的形式增加一个参数legend

```
: import numpy as np
: import matplotlib.pyplot as plt
:
: plt.figure(figsize=(8,5), dpi=80)
: ax = plt.subplot(111)
: 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))
:
: X = np.linspace(-np.pi, np.pi, 256,endpoint=True)
: C,S = np.cos(X), np.sin(X)
:
: plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")
: plt.plot(X, S, color="red", linewidth=2.5, linestyle="-", label="sine")
:
: plt.xlim(X.min()*1.1, X.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.ylim(C.min()*1.1,C.max()*1.1)
: plt.yticks([-1, +1],
:            [r'$-1$', r'$+1$'])
:
: plt.legend(loc='upper left', frameon=False)
: # plt.savefig("../figures/exercice_8.png",dpi=72)
: plt.show()
```



# matplotlib的实例

## 添加注释

我们希望在 2π/3 的位置给两条函数曲线加上一个注释。
首先，我们在对应的函数图像位置上画一个点；
然后，向横轴引一条垂线，以虚线标记；
最后，写上标签。

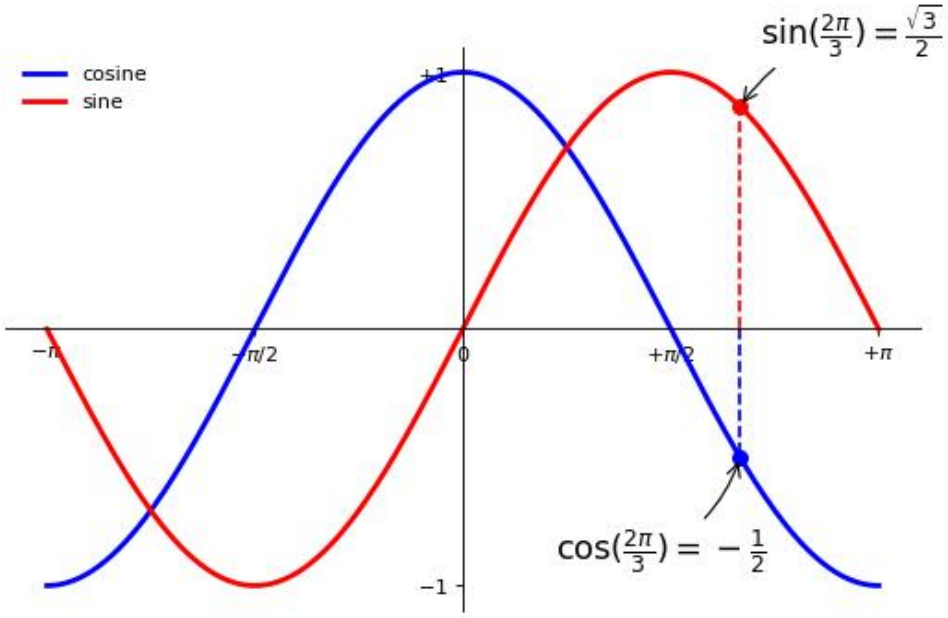
```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5), dpi=80)
ax = plt.subplot(111)
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))

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

plt.plot(X, C, color="blue", linewidth=2.5,
linestyle="-", label="cosine")
plt.plot(X, S, color="red", linewidth=2.5,
linestyle="-",  label="sine")

plt.xlim(X.min()*1.1, X.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.ylim(C.min()*1.1,C.max()*1.1)
plt.yticks([-1, +1],
        [r'$-1$', r'$+1$'])

t = 2*np.pi/3
plt.plot([t,t],[0,np.cos(t)],
      color ='blue',  linewidth=1.5, linestyle="--")
plt.scatter([t,],[np.cos(t),], 50, color ='blue')
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"))

plt.plot([t,t],[0,np.sin(t)],
      color ='red',  linewidth=1.5, linestyle="--")
plt.scatter([t,],[np.sin(t),], 50, color ='red')
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.legend(loc='upper left', frameon=False)
#plt.savefig("../figures/exercice_9.png",dpi=72)
plt.show()
```

# matplotlib的实例

## 添加注释

我们希望在 2π/3 的位置给两条函数曲线加上一个注释。
首先，我们在对应的函数图像位置上画一个点；
然后，向横轴引一条垂线，以虚线标记；最后，写上标签。

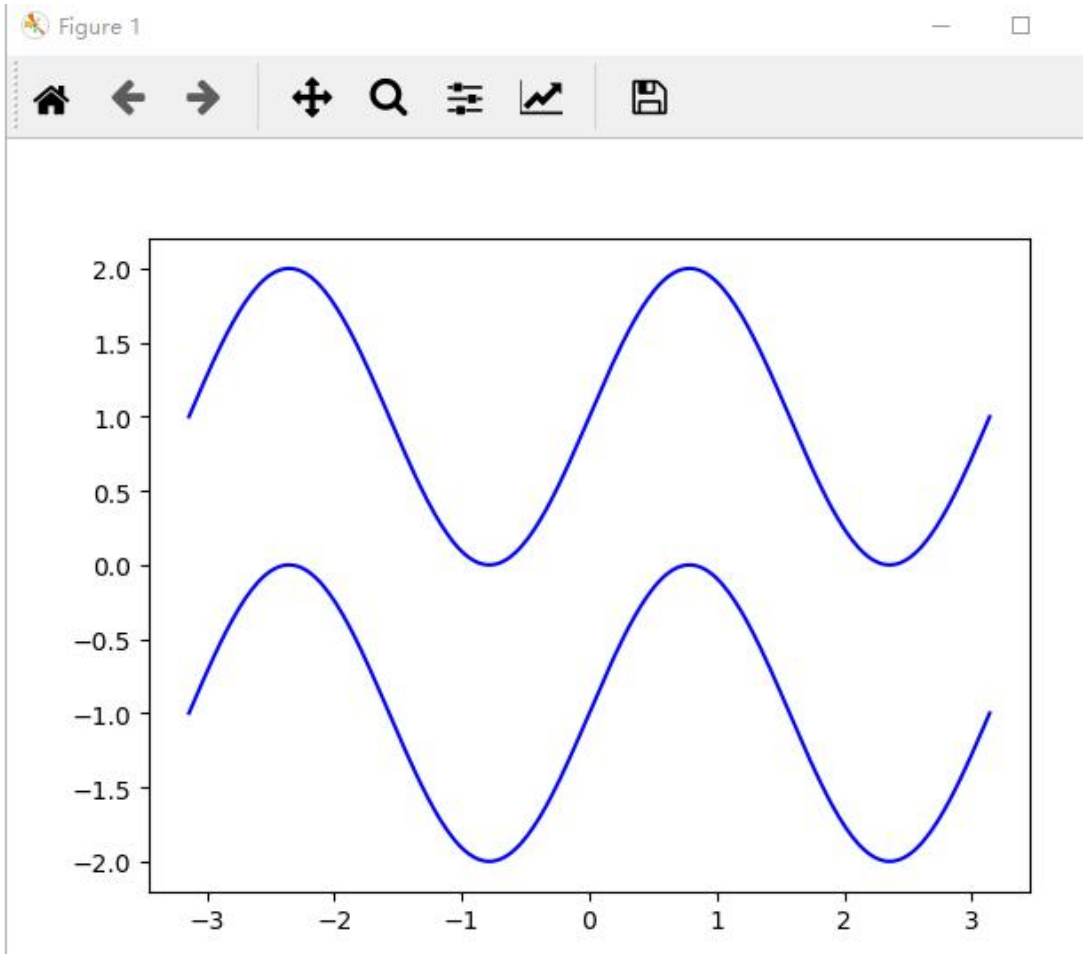
```
t = 2*np.pi/3
plt.plot([t,t],[0,np.cos(t)],
       color ='blue',  linewidth=1.5, linestyle="--")
plt.scatter([t,],[np.cos(t),], 50, color ='blue')
plt.annotate(r'$\cos(\frac{2\pi}{3})=-\frac{1}{2}$',
           xy=(t, np.cos(t)),  xycoords='data',
           xytext=(-90, -50), textcoords='offset points', fontsize=16,
           arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

plt.plot([t,t],[0,np.sin(t)],
       color ='red',  linewidth=1.5, linestyle="--")
plt.scatter([t,],[np.sin(t),], 50, color ='red')
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.legend(loc='upper left', frameon=False)
#plt.savefig("../figures/exercice_9.png",dpi=72)
plt.show()
```



# matplotlib的实例

## 其他实例

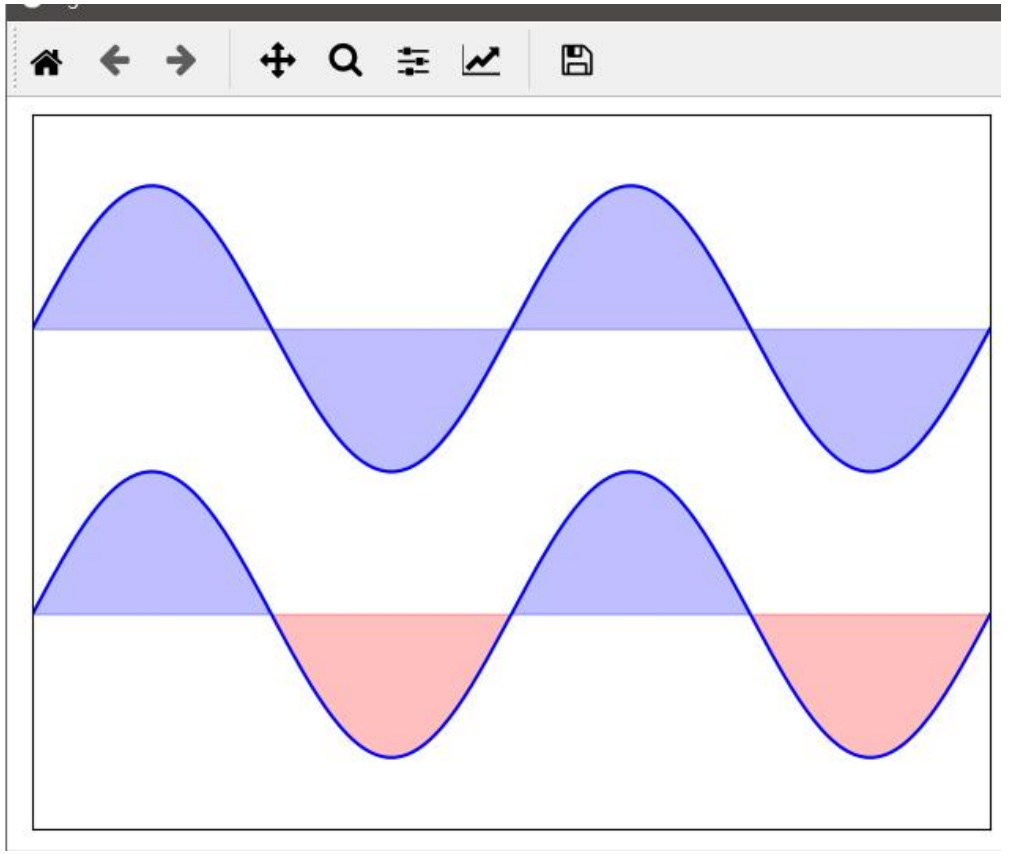
```
In [4]: import numpy as np
   ...: from matplotlib.pyplot import *

In [5]: n = 256
   ...: X = np.linspace(-np.pi,np.pi,n,endpoint=True)
   ...: Y = np.sin(2*X)
   ...:
   ...: plot (X, Y+1, color='blue', alpha=1.00)
   ...: plot (X, Y-1, color='blue', alpha=1.00)
   ...: show()

In [6]:
```



# matplotlib的实例

## 其他实例

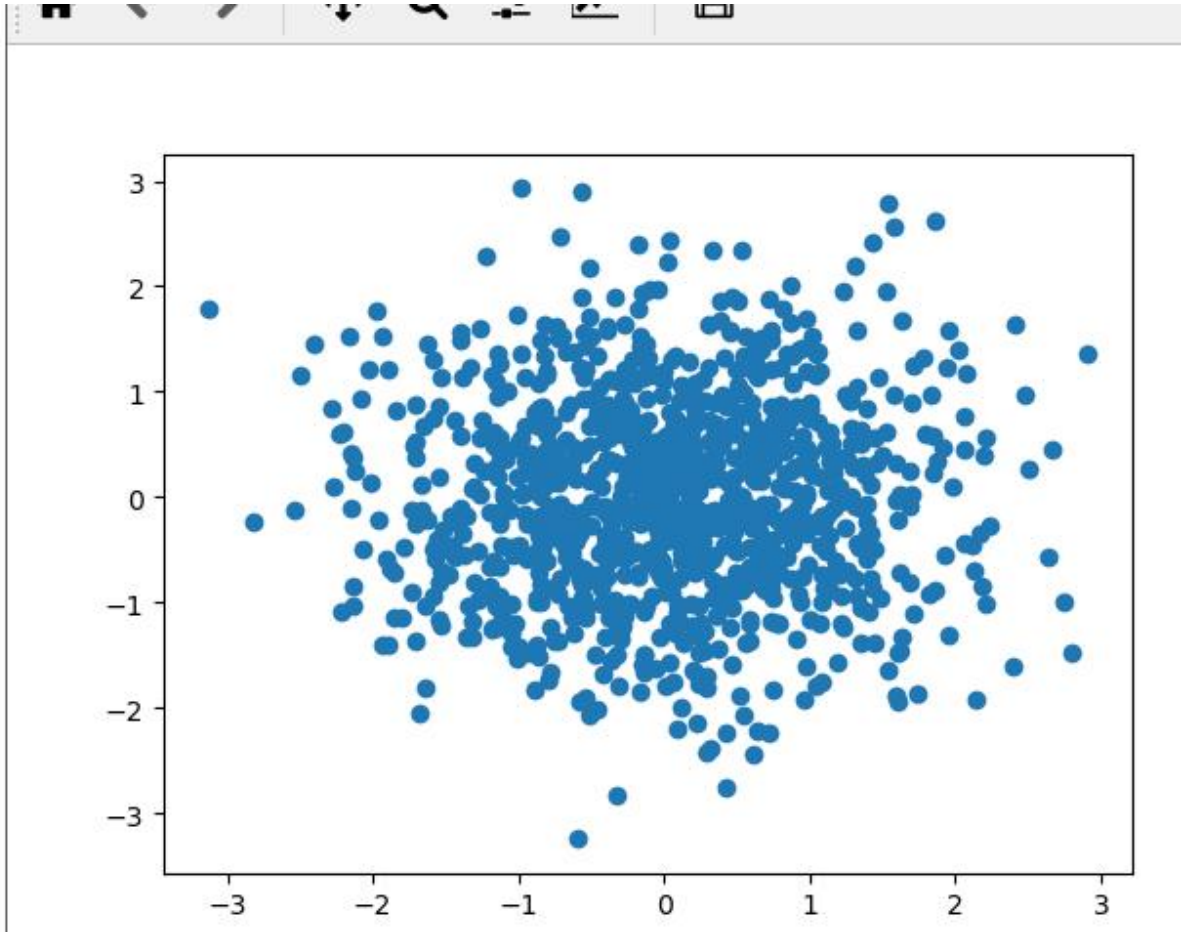
```
In [6]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...: n = 256
   ...: X = np.linspace(-np.pi,np.pi,n,endpoint=True)
   ...: Y = np.sin(2*X)
   ...:
   ...: plt.axes([0.025,0.025,0.95,0.95])
   ...:
   ...: plt.plot (X, Y+1, color='blue', alpha=1.00)
   ...: plt.fill_between(X, 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 [7]:
```



# matplotlib的实例

## 其他实例

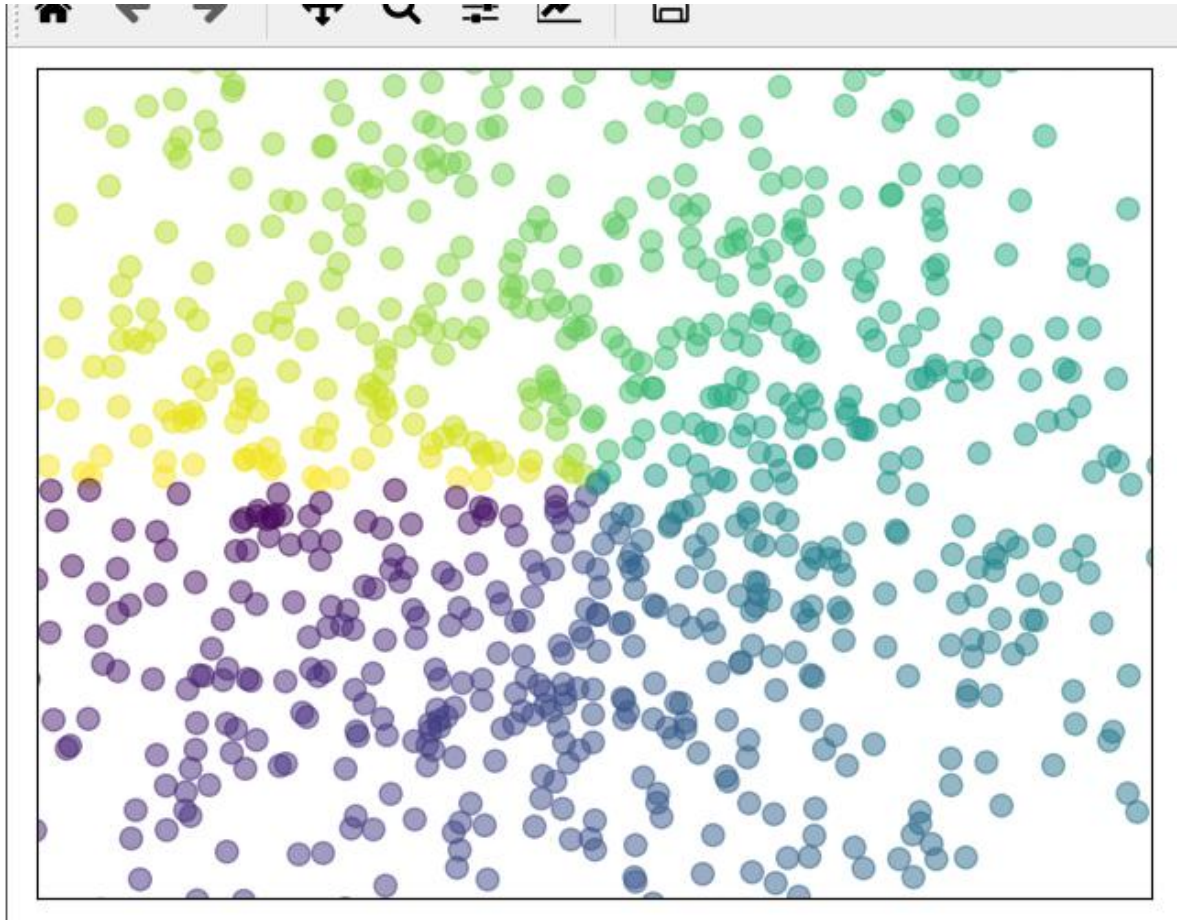
```
In [7]: n = 1024
   ...: X = np.random.normal(0,1,n)
   ...: Y = np.random.normal(0,1,n)
   ...:
   ...: scatter(X,Y)
   ...: show()

In [8]:
```



# matplotlib的实例

## 其他实例

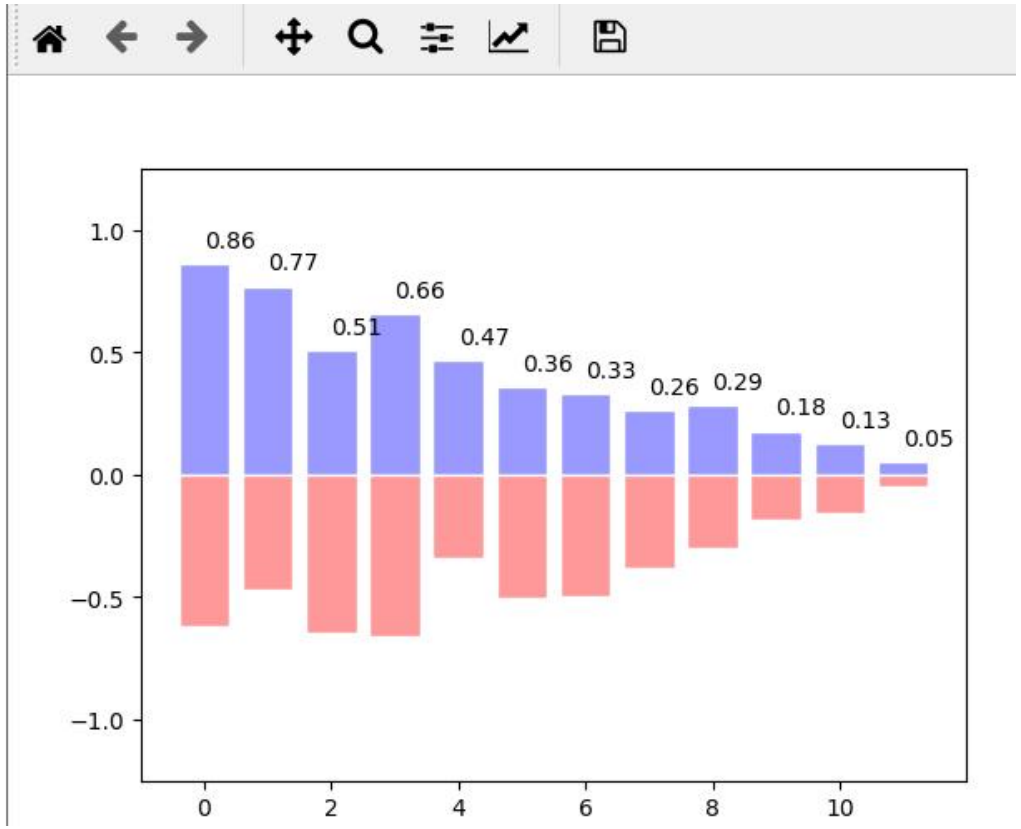
```
n [8]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...:
   ...: n = 1024
   ...: X = np.random.normal(0,1,n)
   ...: Y = np.random.normal(0,1,n)
   ...: T = np.arctan2(Y,X)
   ...:
   ...: plt.axes([0.025,0.025,0.95,0.95])
   ...: plt.scatter(X,Y, s=75, c=T, alpha=.5)
   ...:
   ...: plt.xlim(-1.5,1.5), plt.xticks([])
   ...: plt.ylim(-1.5,1.5), plt.yticks([])
   ...:
   ...: plt.show()

n [9]:
```



# matplotlib的实例

## 其他实例

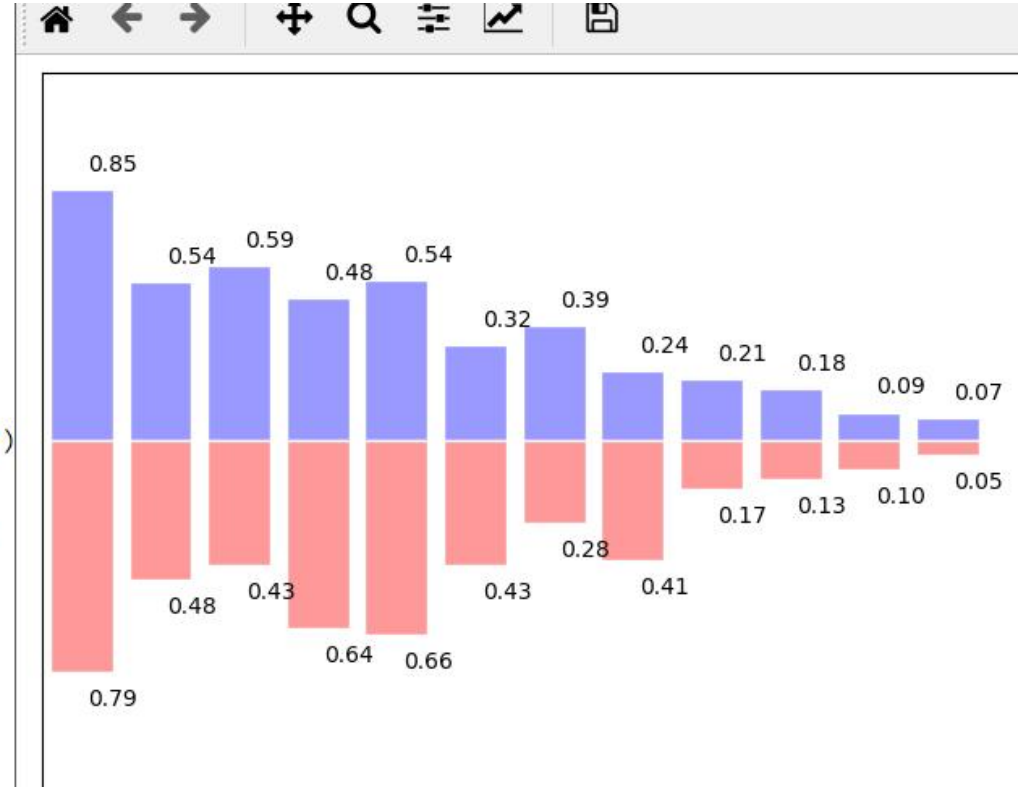
```
In [10]: n = 12
    ...: X = np.arange(n)
    ...: Y1 = (1-X/float(n)) * np.random.uniform(0.5,1.0,n)
    ...: Y2 = (1-X/float(n)) * np.random.uniform(0.5,1.0,n)
    ...:
    ...: bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
    ...: bar(X, -Y2, facecolor='#ff9999', edgecolor='white')
    ...:
    ...: for x,y in zip(X,Y1):
    ...:     text(x+0.4, y+0.05, '%.2f' % y, ha='center', va= 'bottom')
    ...:
    ...: ylim(-1.25,+1.25)
    ...: show()

In [11]:
```



# matplotlib的实例

## 其他实例

```
In [12]: 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 [13]:
```



# 感谢参与 下堂课见