Notepad of <Python for Data Analysis>

Matplotlib

Kitsu Liu
Graduate School of Environment and Information Sciences
Yokohama National University

目录

1 基础（开始） 3

2 折线图 3

2.1 基础指令 3
2.2 设置不同的线条 6
2.3 风格设置（皮肤） 14
2.4 填充图片 17

3 条形图 18

3.1 设置不同的条件下表示不同颜色（比如大于 0 就是 xx 色，小于 0 就是 yy 色） 21

4 盒图 28

5 子图 34

6 给图片增加注释 37

7 绘图细节 39

8 直方图 47

9 散点图 49

10 三维做图 53

11 Pie 图 58

12 设置子图布局 **61**

12.1 嵌套图 62

1 基础（开始）

[26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#把几个必要的库导入进来

%matplotlib inline
#使得像展现的图不需要 plt.show 这个指令就能直接出来
```

2 折线图

2.1 基础指令

1. **plt.plot(array1,array2)** 对 array1,array2 画出折线图，默认谁在前谁就是 x 轴

2. **plt.xlabel('text about x label')** / **plt.ylabel('text about y label')** 设置 x 轴和 y 轴的标签解释

3. **plt.xlabel('text about x label',fontsize = xx)** 设置 x 轴的 label 的字体大小，y 轴同理

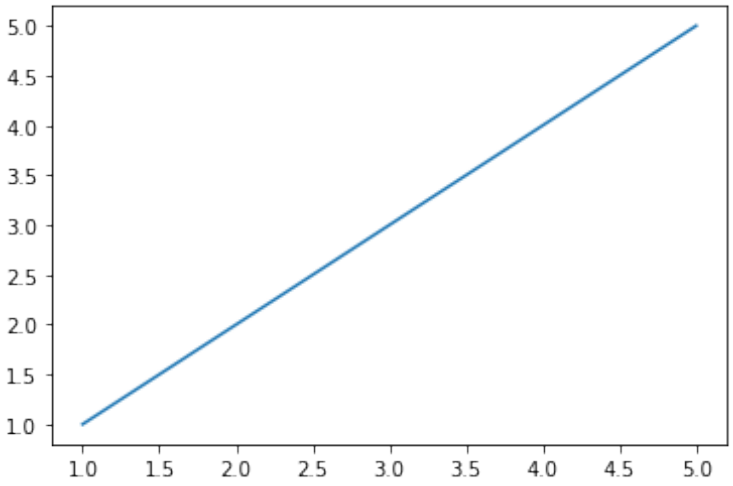
4. **plt.title('text about title',fontsize = xx)** 设置图片的标题，以及字体大小

5. **plt.text(x 坐标,y 坐标,'文字内容',fontsize = xx)** 在图片里的指定坐标的位置添加文本，用于组配注释，x 坐标,y 坐标的位置直接写数字就行

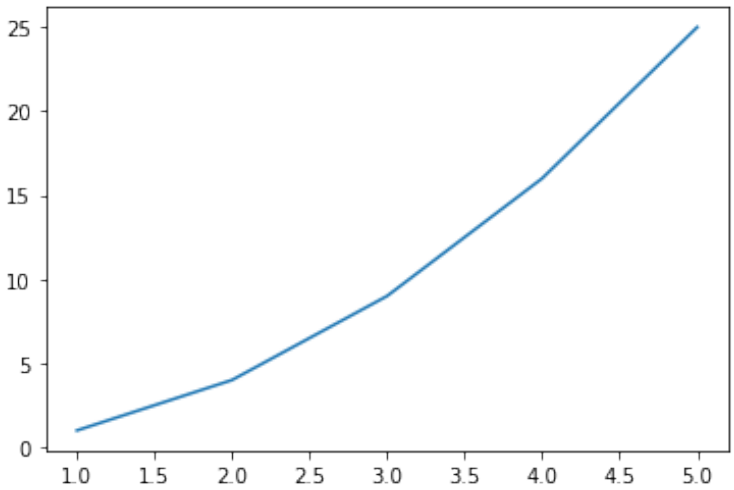
6. **plt.grid(True)** 是否加网格，True 是，False 否

7. **plt.annotate('文本',xy=(x,y),xytext=(2,-0.5),arrowprops = dict(facecolor = 'black',shrink =** 加用于说明某个点的箭头和文本，**xy=** 里面写的是箭头指向哪里，**xytext** 是文本放在哪里，**arrowprops=** 是用来设置箭头格式的，比如颜色和长度 **shrink**，详细参考

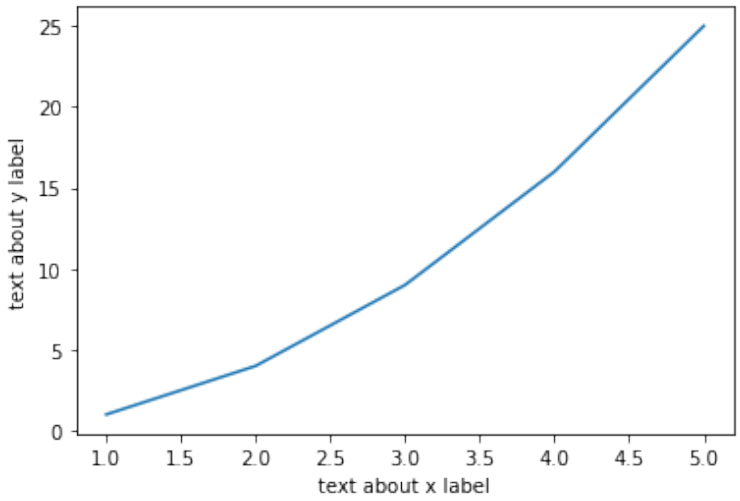
[12]:
```
plt.plot([1,2,3,4,5],[1,2,3,4,5])
#一个例子，让 matplotlib 帮我们自动设置谁是 x 轴，谁是 y 轴
```



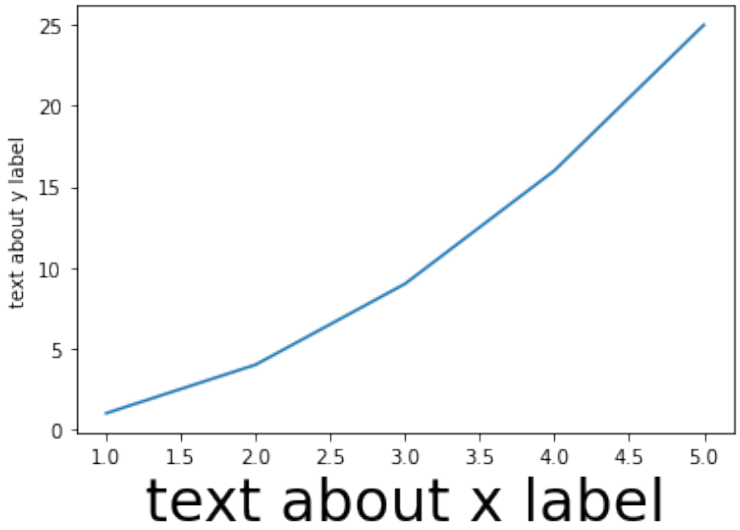
[13]:
```
plt.plot([1,2,3,4,5],[1,4,9,16,25])  #把后面的数字设置成平方
```



[16]:
```
plt.plot([1,2,3,4,5],[1,4,9,16,25])
plt.xlabel('text about x label')
plt.ylabel('text about y label')
#我们加上对于 x 和 y 轴的解释来看看，这样就有 x 和 y 轴的解释了
```



[17]:
```
plt.plot([1,2,3,4,5],[1,4,9,16,25])
plt.xlabel('text about x label',fontsize = 30)
plt.ylabel('text about y label') #肉眼可见的 x 轴的说明文字变大了
```



2.2 设置不同的线条

1. plt.plot([1,2,3,4,5],[1,4,9,16,25],linestyle = 线条类型,color = 线条颜色,linewidth = 线条粗度, marker = 关键节点)

- 线条类型的写法如下：

字符命令	类型	字符命令	类型	字符命令	类型
‘ - ’	实线	’ . ’	点	’ d ’	瘦菱形点
’ -. ’	虚点线	’ o ’	圆点	’ x ’	乘号点
’ ^ ’	上三角形	’ > ’	右三角形	’ H ’	六边形点 2
’ 2 ’	上三叉点	’ 4 ’	右三叉点	’ * ’	星型点
’ p ’	五角点	‘h’	六边形点	’ s ’	正方点
’ + ’	加号点	’ D ’	实习菱形点	’ 3 ’	左三叉点
’ _ ’	横线点	’ ___ ’	虚线	’ 1 ’	下三叉点
’ : ’	点线	’ , ’	像素点	’ < ’	左三角形
’ v ’	下三角形				

- 线条颜色的写法如下：

命令	线条颜色
‘b’	蓝色 blue
‘g’	绿色 green
’r’	红色 red
’c’	青色 cyan
’m’	品红 magenta
’ y’	黄色 yellow
’ k ’	黑色 black
’w’	白色 white

- 线条粗度写法：**plt.plot([1,2,3,4,5],[1,4,9,16,25],linewidth = xx)**
  xx 表示粗读，数字越大就越粗
- 关键节点写法：
  除以下以外还有很多：click this

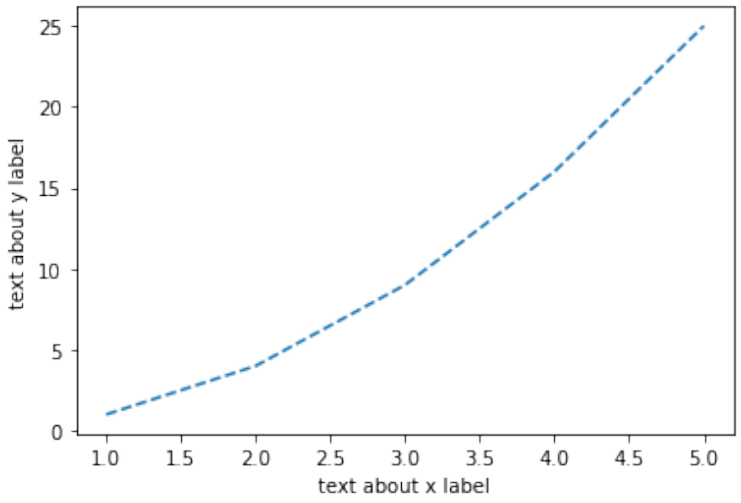
并且还可以通过 **markerfacecolor = 'r'** 以及 **markersize = 10** 来指定 marker 点的大小和颜色

还可以把线条和颜色指定写在同一条命令 '颜色 线条': 这么个写法，但是中间没有空格，空格是为了好分辨
比如 **'r--'**, **'g.'** 等等

2. plt.setp(线条名, 参数 1, 参数 2)
哪怕是画出来的设定好的线条，我们依旧可以改它的参数，**把线条名带进去后，依次写想改的参数就好了**

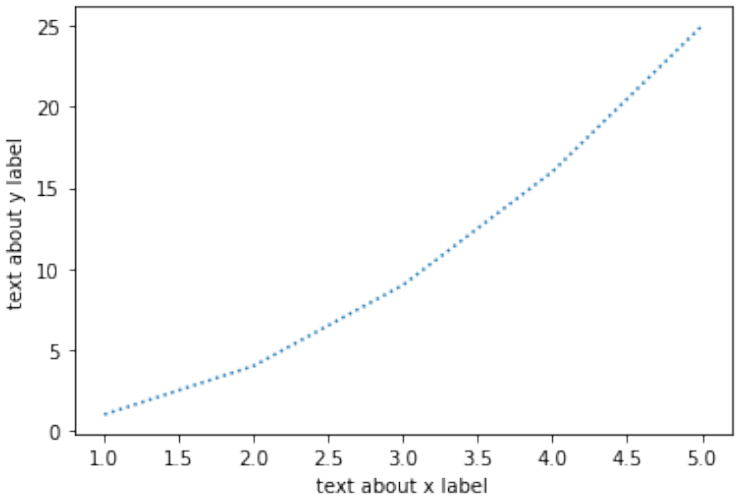
[20]:
```
plt.plot([1,2,3,4,5],[1,4,9,16,25],linestyle ='--')  #加一个‘--’看看  变成虚线
plt.xlabel('text about x label')
plt.ylabel('text about y label')
```

[20]: Text(0, 0.5, 'text about y label')



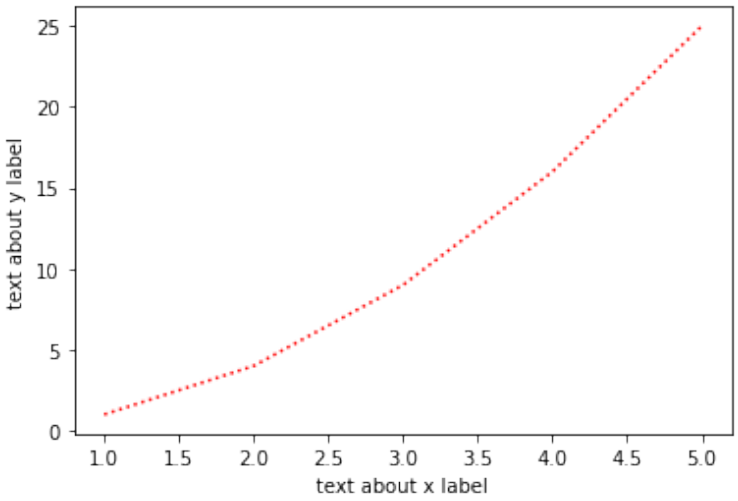
[21]:
```
plt.plot([1,2,3,4,5],[1,4,9,16,25],linestyle =':')  #加一个‘:’看看  变成点线
plt.xlabel('text about x label')
plt.ylabel('text about y label')
```

[21]: Text(0, 0.5, 'text about y label')

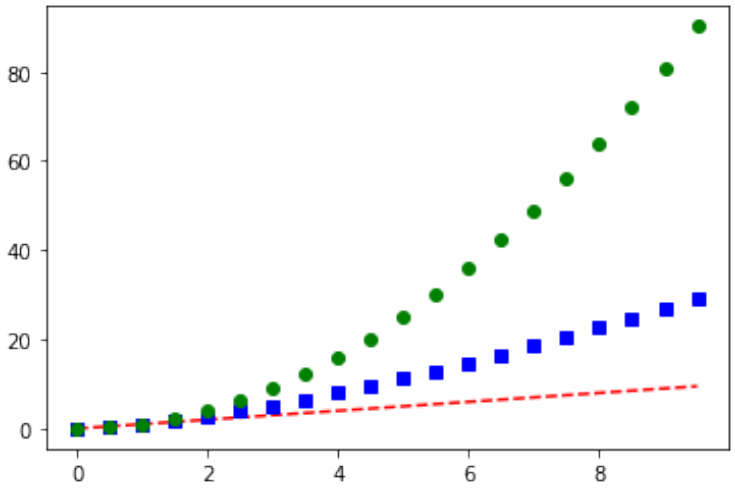


[22]:
```
plt.plot([1,2,3,4,5],[1,4,9,16,25],linestyle =':',color = 'r')  #变个红色玩玩
plt.xlabel('text about x label')
plt.ylabel('text about y label')
```

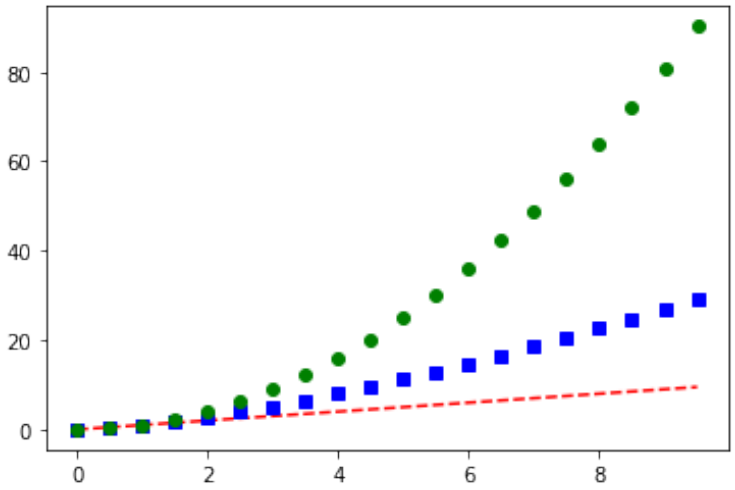
[22]: Text(0, 0.5, 'text about y label')



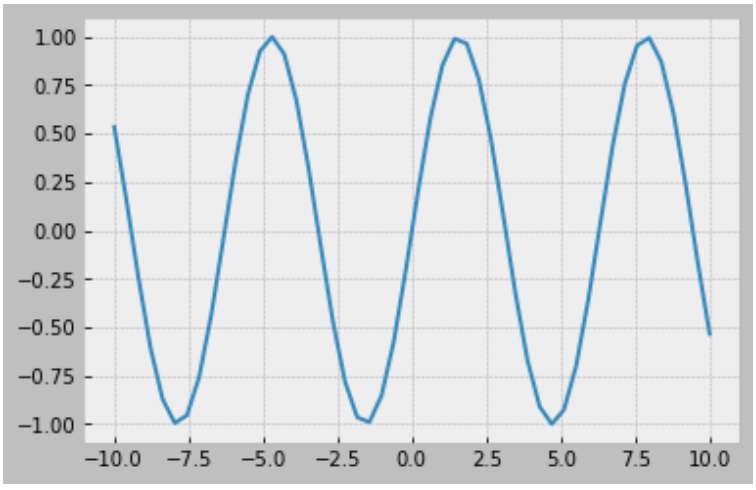
[26]:
```
array = np.arange(0,10,0.5)  #0-10之间每0.5生成一个数字
plt.plot(array,array,linestyle = '--',color = 'r')
plt.plot(array,array**1.5,linestyle ='s',color = 'b') #y轴设置成array的1.5次方
plt.plot(array,array**2,linestyle ='o',color = 'g') #y轴设置成array的平方
#把三个图画在一起
```



[28]:
```
plt.plot(array,array,'r--',
         array,array**1.5,'bs',
         array,array**2,'go')
#也可以把它们写在一个括号里
#但是这样的话就必须把线条和颜色指定写在同一条命令了  '颜色 线条': 这么个写法，但是中间没有空格，空格是为了好分辨
```

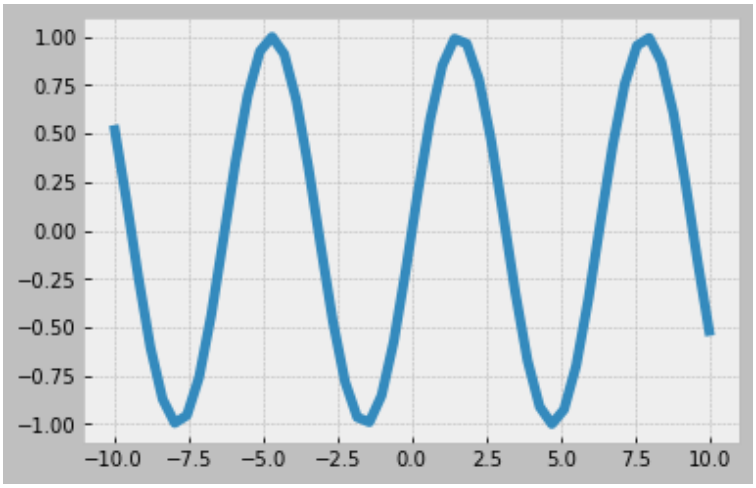


[47]:
```
x = np.linspace(-10,10)
y = np.sin(x)
plt.plot(x,y) #画出这个图觉得线条有点细
```



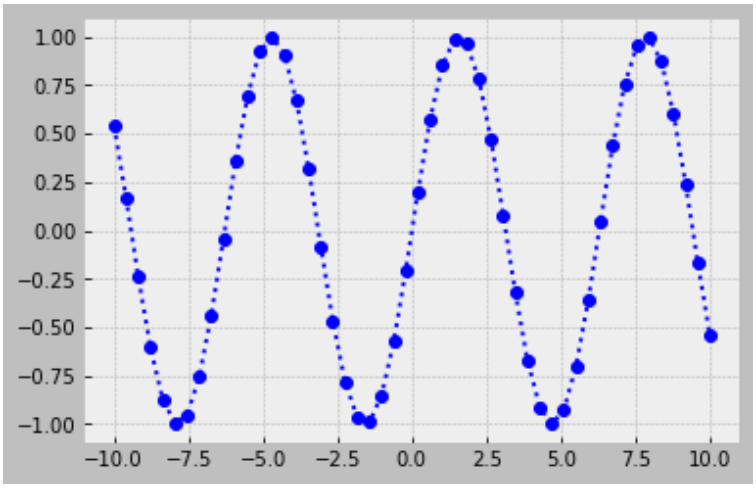
[48]:
```
plt.plot(x,y,linewidth = 5)
#加粗
```

[48]: [<matplotlib.lines.Line2D at 0x7fe20ddb5f70>]



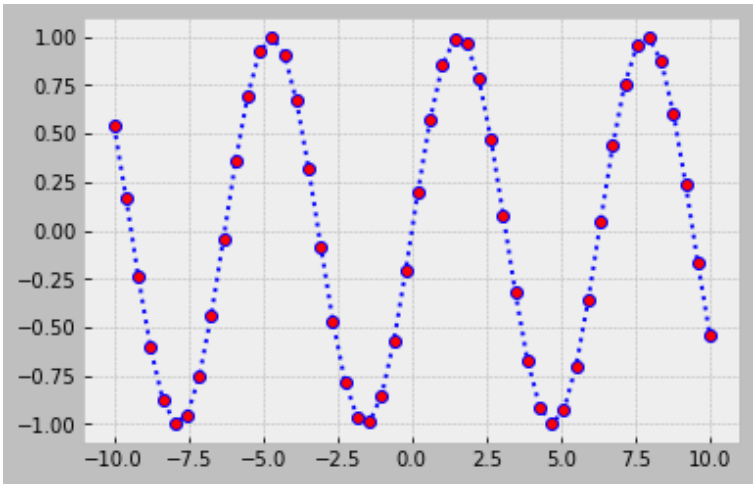
[49]:
```
plt.plot(x,y,color = 'b',linestyle = ':',marker = 'o') #用 marker 指令来点出关键节点
```

[49]: [<matplotlib.lines.Line2D at 0x7fe207e554c0>]



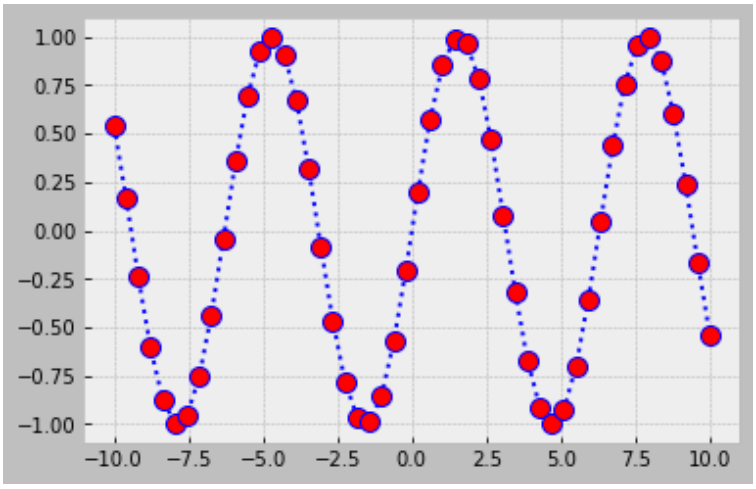
[50]:
```
plt.plot(x,y,color = 'b',linestyle = ':',marker = 'o',markerfacecolor = 'r')
# 线和 marker 点都是蓝色觉得有点不好分辨，于是把点设成红色
```

[50]: [<matplotlib.lines.Line2D at 0x7fe20e2a92b0>]



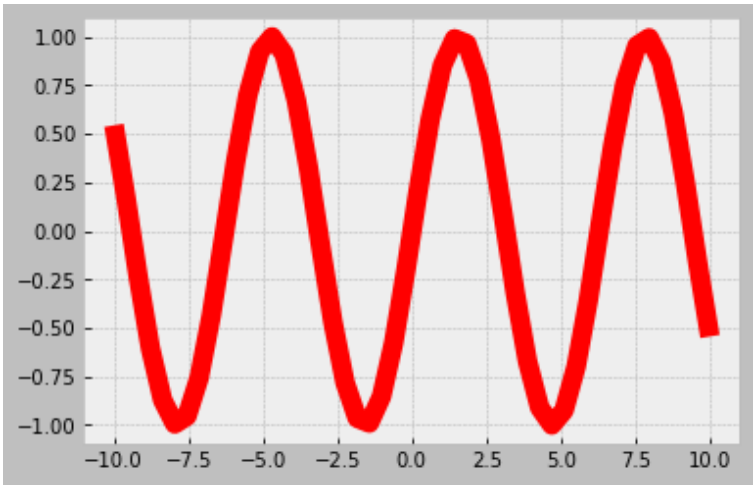
[51]:
```
plt.plot(x,y,color = 'b',linestyle = ':',marker = 'o',markerfacecolor =␣
↪'r',markersize = 10)
#marker 点有点小，给它整大点
```

[51]: [<matplotlib.lines.Line2D at 0x7fe20e3f6700>]



[52]:
```
line = plt.plot(x,y)
plt.setp(line,color = 'r',linewidth = 10)
#还可以在已经画好的图像，通过 plt.setp() 来更改
```

[52]: [None, None]



2.3 风格设置（皮肤）

1. **plt.style.available** 查看都有什么风格可以用

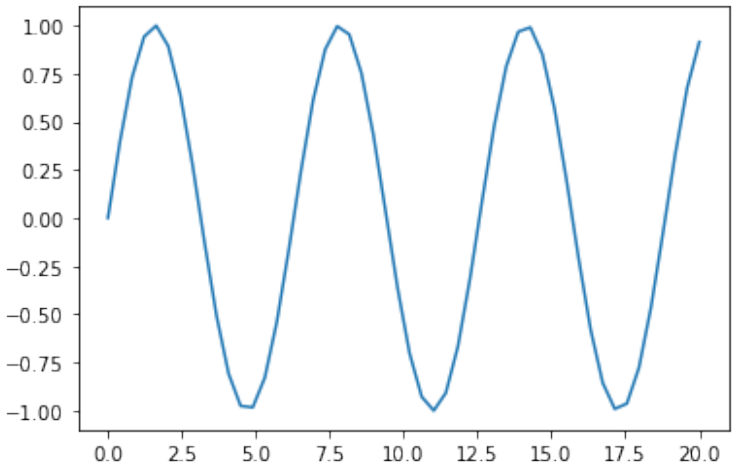
2. **plt.style.use(风格)** 打印图片前加上这个命令应用风格，风格两个字的内容是从 **plt.style.available** 里面挑

[25]:
```
plt.style.available
#查看都有什么风格可以用
```

[25]:
```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

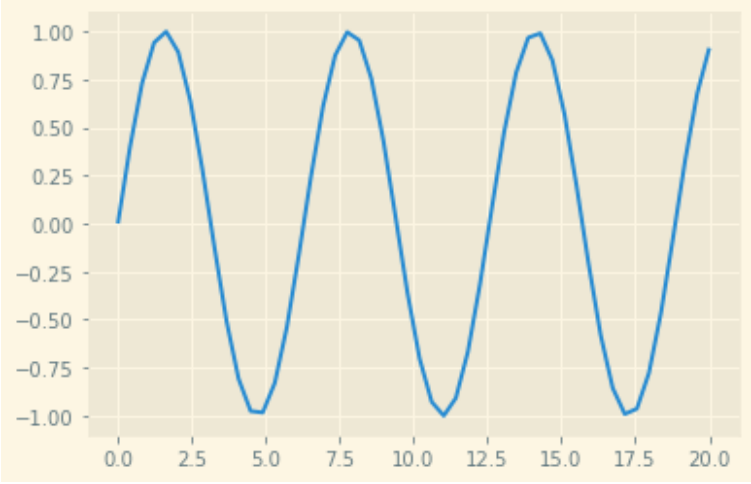
[27]:
```
x = np.linspace(0,20)
y = np.sin(x)
plt.plot(x,y)
#看一下原图
```

[27]: [<matplotlib.lines.Line2D at 0x7fe20d56e160>]



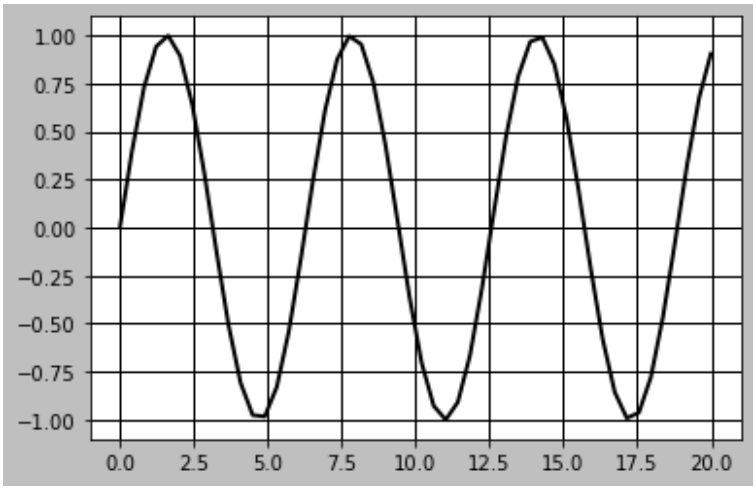
[28]:
```
plt.style.use('Solarize_Light2')
plt.plot(x,y)
```

[28]: [<matplotlib.lines.Line2D at 0x7fe20d68bbe0>]



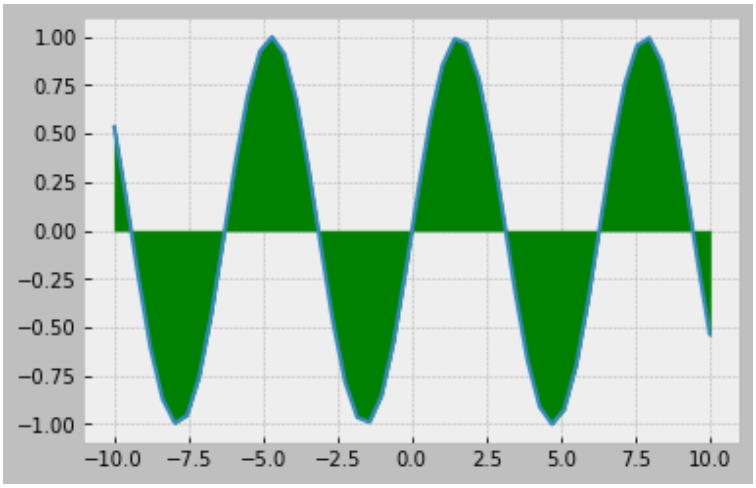
[30]:
```
plt.style.use('grayscale')
plt.plot(x,y)
```

[30]: [<matplotlib.lines.Line2D at 0x7fe20cc78940>]

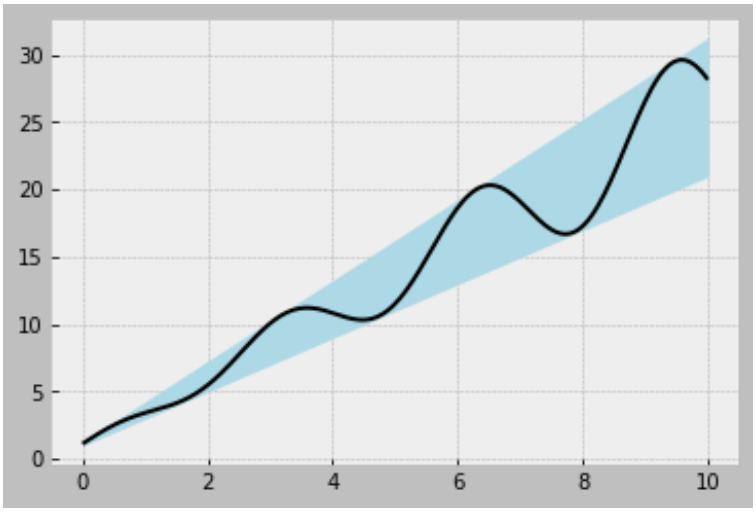


2.4 填充图片

[53]:
```
x = np.linspace(-10,10)
y = np.sin(x)
plt.plot(x,y)
plt.fill_between(x,y,color = 'green') #原本是个单调的折线图，然后现在可以填充轴和线之间的空白
```



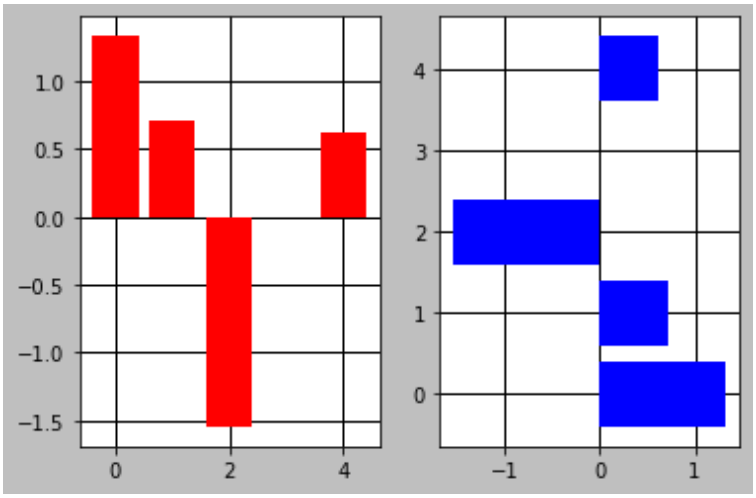
[55]:
```
x = np.linspace(0,10,200) #0～10中间等差取200个数
y1 = 2*x + 1
y2 = 3*x + 1.2
y_mean = 0.5*x*np.cos(2*x) + 2.5*x + 1.1
plt.fill_between(x,y1,y2,color = 'lightblue')
plt.plot(x,y_mean,color = 'black')
```



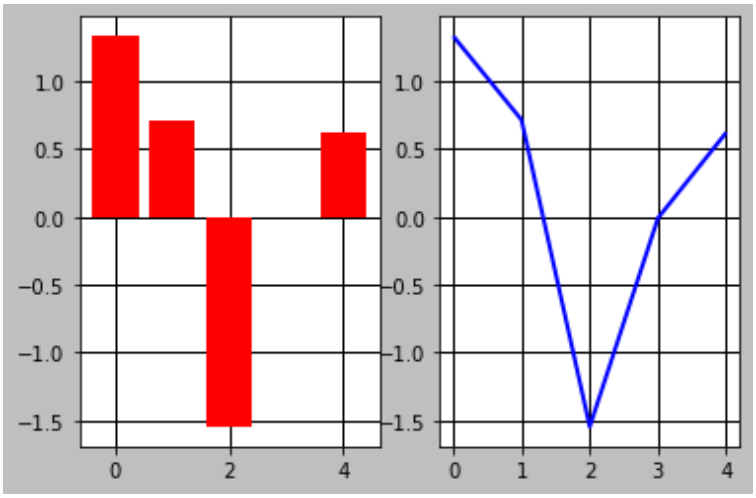
3 条形图

- 普通绘制条形图：
  - 横图：**plt.bar(横轴刻度/显示的文字这种感觉, 数值的 array,color = '')**
  - 纵图：**plt.barh(纵轴刻度/显示的文字这种感觉, 数值的 array,color = '')**
- 子图：
  - 同时设置 fig 和 axe：**fig,axes = plt.subplots(ncols = 2)**，里面写 2 表示是话两个图
  - 画第一个图：**axes[0]. 图命令**，其中图命令可写：plot 呀 bar，barh 等
  - 画第二个图：**axes[1]. 图命令**，同上
  - 下面的就都是以此类推了
- 指定一条‘境界线’：
  - 横着的线：**plt.axhline(纵轴坐标,color = '',linewidth =)**，只用指定【纵】坐标就行
  - 竖着的线：**plt.axvline(横轴坐标,color = '',linewidth =)**，只用指定【横】坐标就行
  - 于子图的应用：**axes[n].axhline(参数同上)** / `axes[n].axvline(参数同上)`，n 里面写诗第几幅图，从 0 开始计数
- 改柱子颜色，宽度，包边颜色, 花纹等的**万能设置函数**: 详见：click this
  bar.set(color = 'red',edgecolor = 'black',linewidth = 3,hatch = ...)，基本上什么都能设置
  只要’ 图的名字’ 或者是’ 子图的名字’.set 就行了，想改什么参数就往里面传就行。
  python 内查看帮助文档**help(matplotlib.patches.Rectangle.set)**

[35]:
```
np.random.seed(10)
x = np.arange(5)
y = np.random.randn(5)

fig,axes = plt.subplots(ncols = 2)
v_bars = axes[0].bar(x,y,color = 'r')
h_bars = axes[1].barh(x,y,color = 'b')
```

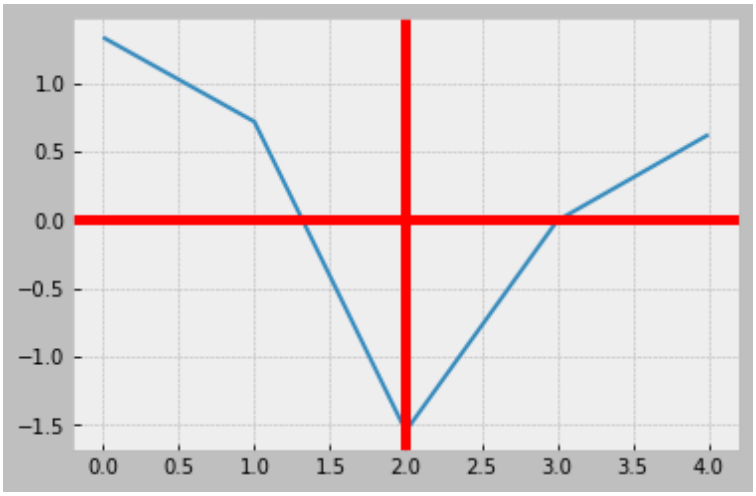


[39]:
```
fig,axes = plt.subplots(ncols = 2)
v_bars = axes[0].bar(x,y,color = 'r')
plot = axes[1].plot(x,y,color = 'b')
#可以这样给不同的图拼起来
```



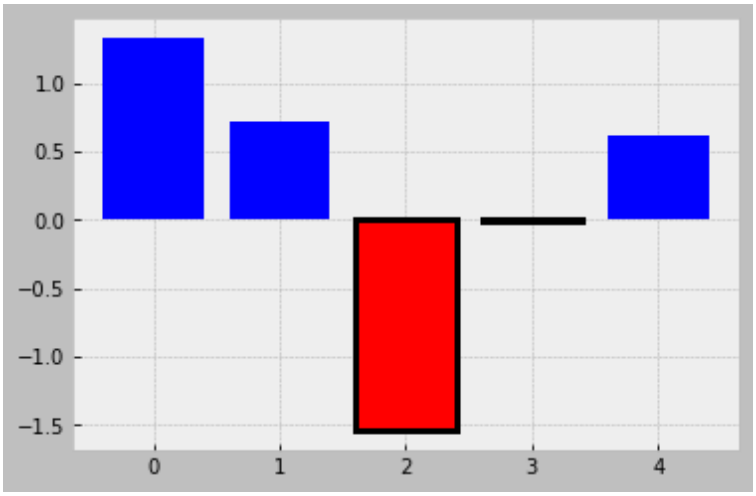
[42]:
```
plt.style.use('bmh')
plt.plot(x,y)
plt.axhline(0,color = 'r',linewidth = 5)
plt.axvline(2,color = 'r',linewidth = 5)
#换个画图风格的同时，指定两条线
```

[42]: <matplotlib.lines.Line2D at 0x7fe20e118a00>



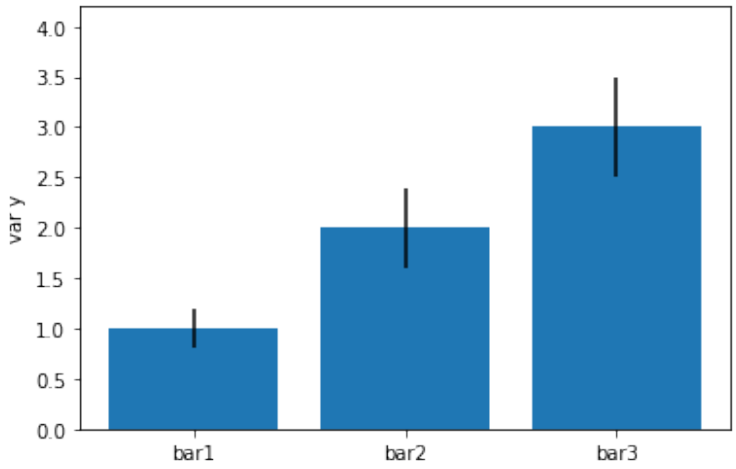
3.1 设置不同的条件下表示不同颜色（比如大于 0 就是 xx 色，小于 0 就是 yy 色）

[46]:
```
fig,axes = plt.subplots()
v_bar = axes.bar(x,y,color='blue')
for bar,height in zip(v_bar,y):
    if height<0:
        bar.set(color = 'red',edgecolor = 'black',linewidth = 3)
#大于 0 的就是蓝色，小于 0 的就是红色
```



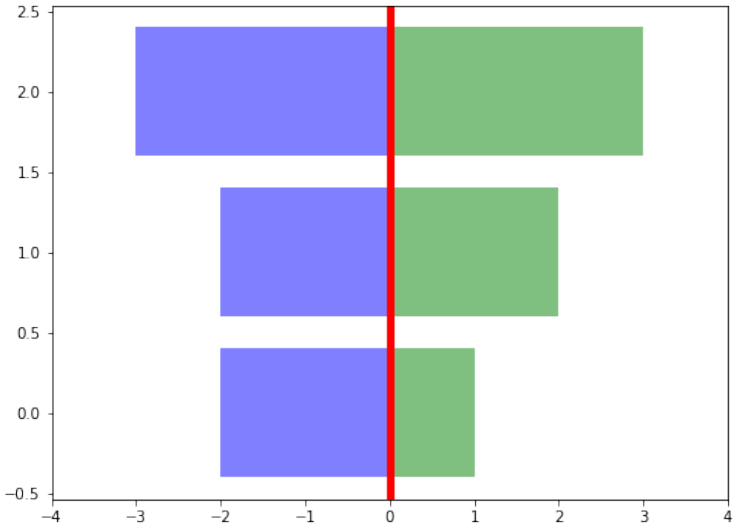
[5]:
```
mean_values = [1,2,3]
var = [0.2,0.4,0.5]
bar_label = ['bar1','bar2','bar3']

x_pos = list(range(len(bar_label)))
plt.bar(x_pos,mean_values,yerr = var) #重点是这个yerr，误差棒，表示误差是怎么样的范围内
max_y = max(zip(mean_values,var))
plt.ylim(0,(max_y[0]+max_y[1])*1.2)
plt.ylabel('var y')
plt.xticks(x_pos,bar_label)
```

[5]:
```
([<matplotlib.axis.XTick at 0x7fce19a1a250>,
  <matplotlib.axis.XTick at 0x7fce19a1a220>,
  <matplotlib.axis.XTick at 0x7fce19a092b0>],
 [Text(0, 0, 'bar1'), Text(1, 0, 'bar2'), Text(2, 0, 'bar3')])
```



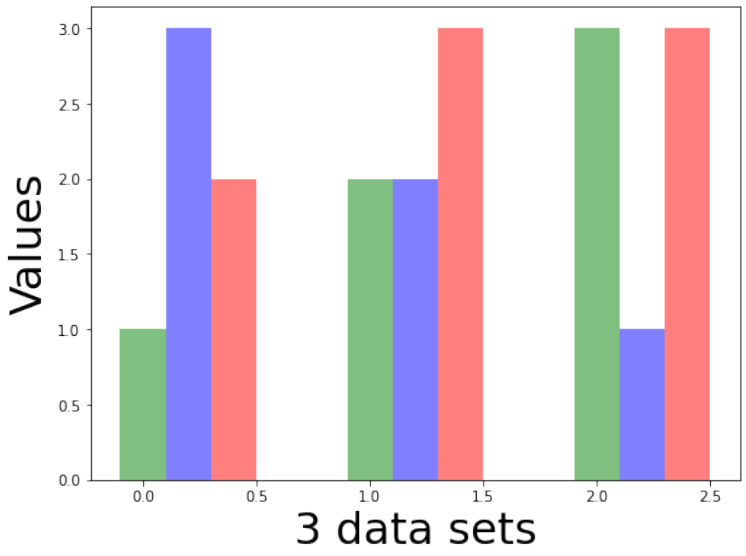
[9]:
```
x1 = np.array([1,2,3])
x2 = np.array([2,2,3])

bar_labels = ['bar1','bar2','bar3']
fig = plt.figure(figsize = (8,6))
y_pos = np.arange(len(x1))
y_pos = [x for x in y_pos]
plt.axvline(0,color = 'r',linewidth = 5)
plt.barh(y_pos,x1,color = 'g',alpha = 0.5)
plt.barh(y_pos,-x2,color = 'b',alpha = 0.5) #让这两个图背靠背
plt.xlim(-max(x2)-1,max(x1)+1)              #设置最大值

#右 x2，左 x1，画一个背靠背出来的图
```

[9]: (-4.0, 4.0)

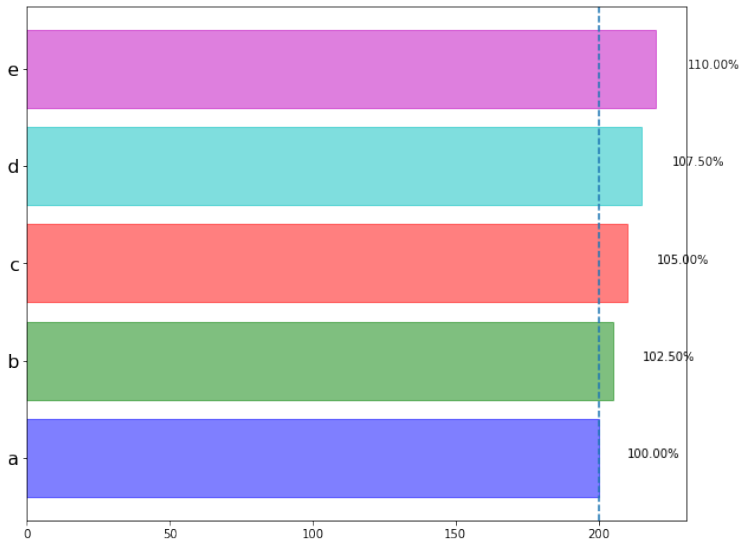


[14]:
```
#把三个柱状图叠在一起
green_data = [1,2,3]
blue_data = [3,2,1]
red_data = [2,3,3]
labels = ['group 1','group 2','group 3']

pos = list(range(len(green_data)))
width = 0.2
fig,ax = plt.subplots(figsize=(8,6))

plt.bar(pos,green_data,width,alpha = 0.5,color = 'g',label = labels[0])
plt.bar([p+width for p in pos],blue_data,width,alpha = 0.5,color = 'b',label =␣
↪labels[1])
plt.bar([p+width*2 for p in pos],red_data,width,alpha = 0.5,color = 'r',label =␣
↪labels[2])
plt.xlabel('3 data sets',fontsize = 30)
plt.ylabel('Values',fontsize=30)
```

[14]:
```
Text(0, 0.5, 'Values')
```

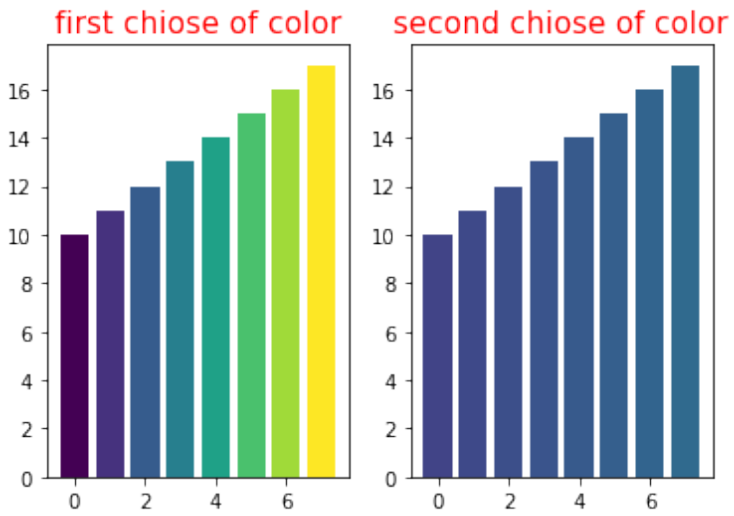


[25]:
```
data = range(200,225,5)
bar_labels = ['a','b','c','d','e']
fig = plt.figure(figsize = (10,8))
y_pos = np.arange(len(data))
plt.yticks(y_pos,bar_labels,fontsize = 16)  #指定 y 轴上的刻度写几个，都写什么
color = ['b','g','r','c','m']
bars = plt.barh(y_pos,data,alpha = 0.5,color = 'g')
plt.axvline(min(data),linestyle = '--')   #画一条竖线看看各个数据高出最低的多少。

for b,d,e in zip(bars,data,color):   # b in bars 表示中，bars 的范围是里面有几根柱子
    plt.text(b.get_width()+b.get_width()*0.05,b.get_y()+b.get_height()/2,'{0:.
↪2%}'.format(d/min(data)))
    #在图中给每根柱子都添加文本，内容为现在这根柱子的值是最小的那个的百分之多少。
    #b.get_width()+b.get_width()*0.05 【横坐标位置】意为：在横坐标上位置比柱子的末尾多出来柱子长度的 0.05
    #b.get_y()+b.get_height()/2 【纵坐标位置】意为：在纵坐标上的位置正好放在柱子的横着的中心线上
    #'{0:.2%}'.format(d/min(data))。【指定文本内容】，为最小值的百分之几，并且只取到小数点后两位
    b.set(color = e)
    #给每根柱子不同的颜色
```

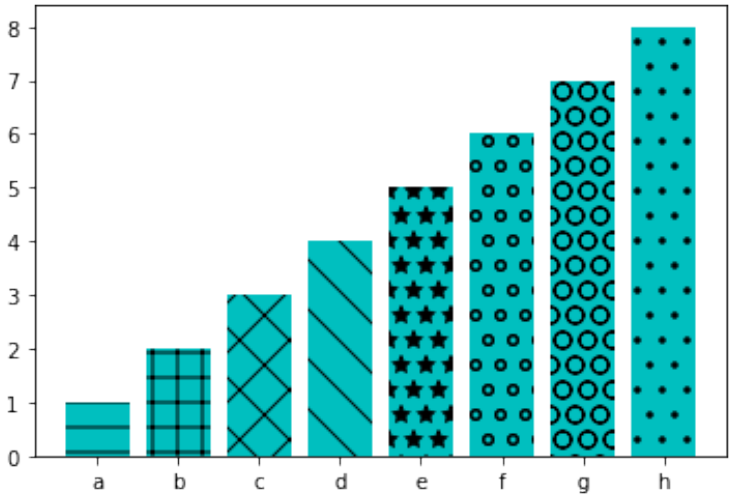


[35]:
```
#不去挨个指定每个柱子的颜色，让他采用一种规律去指定，这样就容易做出那种渐变阶梯的效果
#先加载两个颜色工具
import matplotlib.colors as col
import matplotlib.cm as cm
mean_values = range(10,18)  #range 型，一种 list，表达从 xx 到 yy 的 n 个元素，但是 print 出来只记录头和尾
x_pos = range(len(mean_values))  #从 0 开始间隔为 1 生成 len(mean_values) 个元素的 list
cmap1 = cm.ScalarMappable(col.Normalize(min(mean_values),max(mean_values),cm.hot))   #设置颜色梯度
cmap2 = cm.ScalarMappable(col.Normalize(0,50,cm.hot))
plt.subplot(121)
plt.title('first chiose of color',fontsize = 15,color = 'r')
plt.bar(x_pos,mean_values,color = cmap1.to_rgba(mean_values))
plt.subplot(122)
plt.title('second chiose of color',fontsize = 15,color = 'r')
plt.bar(x_pos,mean_values,color = cmap2.to_rgba(mean_values))
```

[35]:
```
<BarContainer object of 8 artists>
```



[91]:
```
import string  #为了使用下面的字母表而加载

patterns = ('-','+','x','\\','*','o','O','.')

values = range(1,len(patterns)+1)
x_pos = list(range(len(patterns))) #设置一个 list 用于下面的循环
for i in range(len(patterns)):
  x_pos[i] = list(string.ascii_lowercase)[i]

bars = plt.bar(x_pos,values,color = 'c')
for b,p in zip(bars,patterns):
    b.set(hatch = p) #设置花纹，还可以用更标准的写法：set_hatch()
```



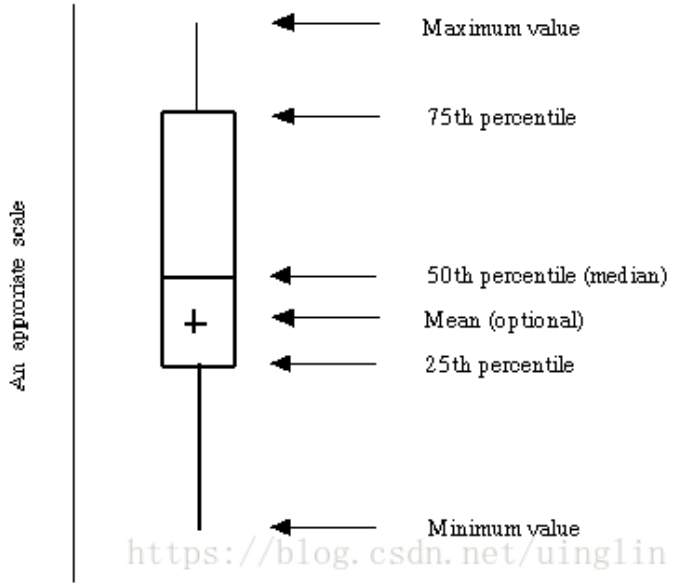
4 盒图

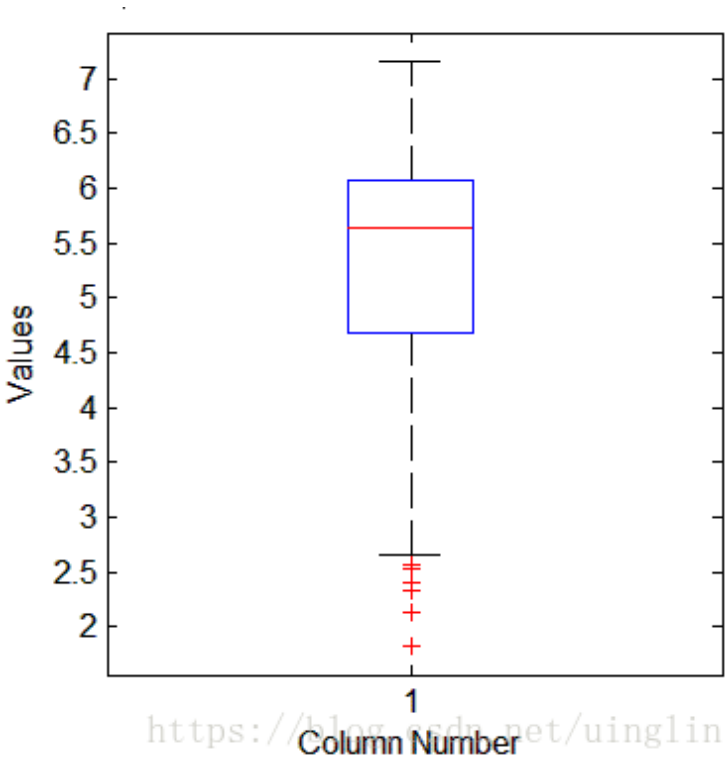
1. **介绍**

   盒图是在 1977 年由美国的统计学家约翰 · 图基 (John Tukey) 发明的。**它只专注于五个特殊的统计量，并由五个数值点组成：最小值 (min)，下四分位数 (Q1)，中位数 (median)，上四分位数 (Q3)，最大值 (max)。也可以往盒图里面加入平均值 (mean)**。如上图。下四分位数、中位数、上四分位数组成一个 “带有隔间的盒子”。上四分位数到最大值之间建立一条延伸线，这个延伸线成为 “胡须 (whisker)”。

由于现实数据中总是存在各式各样地 “脏数据”，也成为 “离群点”，于是为了不因这些少数的离群数据导致整体特征的偏移，将这些离群点单独汇出，而盒图中的胡须的两级修改成最小观测值与最大观测值。这里有个经验，就是最大 (最小) 观测值设置为与四分位数值间距离为 1.5 个 IQR(中间四分位数极差)。即 IQR = Q3-Q1，即上四分位数与下四分位数之间的差，也就是盒子的长度。

最小观测值为 min = Q1 - 1.5IQR，如果存在离群点小于最小观测值，则胡须下限为最小观测值，离群点单独以点汇出。如果没有比最小观测值小的数，则胡须下限为最小值。
最大观测值为 $max = Q3 + 1.5$IQR，如果存在离群点大于最大观测值，则胡须上限为最大观测值，离群点单独以点汇出。如果没有比最大观测值大的数，则胡须上限为最大值。





通过盒图，在分析数据的时候，盒图能够有效地帮助我们识别数据的特征：直观地识别数据集中的异常值 (查看离群点)。判断数据集的数据离散程度和偏向 (观察盒子的长度，上下隔间的形状，以及胡须的长度)。

1. 箱体的左侧 (下) 边界代表第一四分位 (Q1), 而右侧 (上) 边界代表第三四分位 (Q3)。至于箱体部分代表四分位距 (IQR), 也就是观测值的中间 50% 值。2. 在箱体中间的线代表的是数据的中位数值。3. 从箱体边缘延伸出去的直线称为触须 (whisker). 触须 (whisker) 的向外延伸表示了数据集中的最大和最小 (异常点除外)。

4. 极端值或异常点 (outlier), 用星号 (*) 来标识. 如果一个值位于箱体外面 (大于 Q3 或小于 Q1), 并且距离相应边界大于 1.5 倍的 IQR, 那么这个点就被认为是一个异常点 (outlier)。

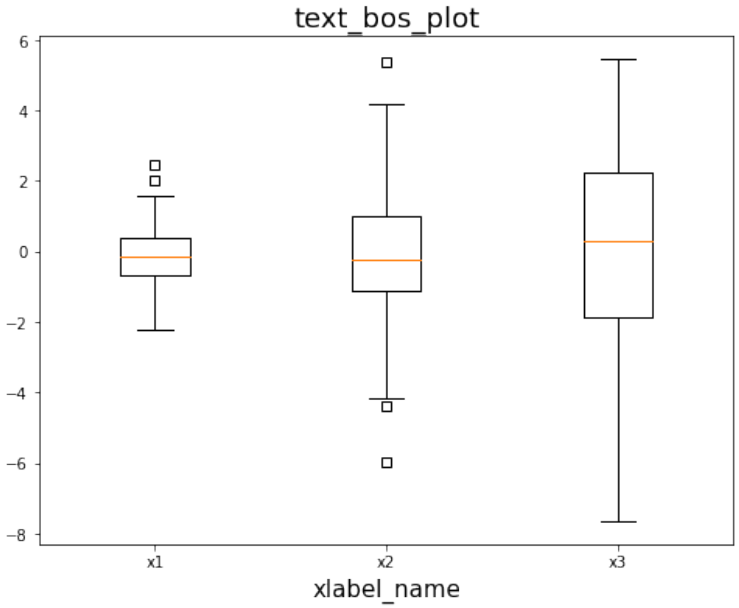
2. **画图：**

   plt.boxplot(data,notch = False,sym = 's',vert = True...)：

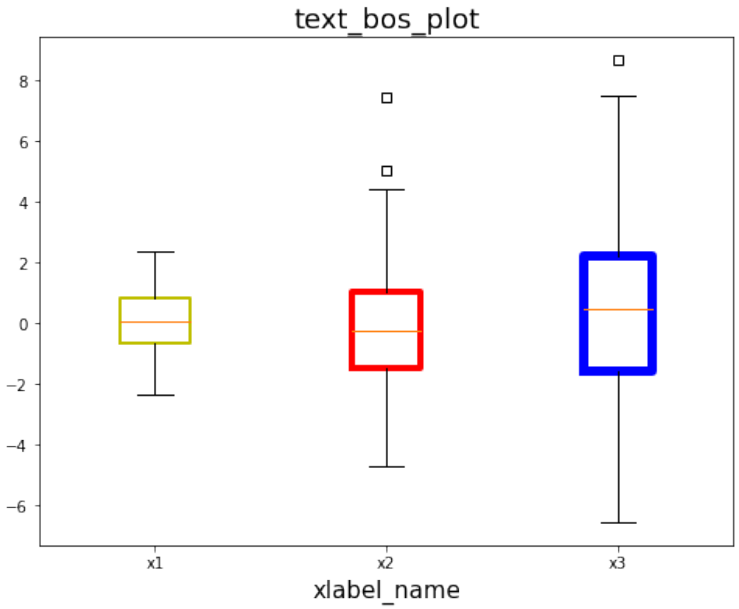
   - **data**:data_array
   - **notch = False**: 默认 false，是否画出凹点，来表示置信区间
   - **sym = 's'**: 离群点用什么来标记，还有其他的选项，参考 1.1.2 的【折线图-设置不同线条】
   - **vert = True**: 是否要竖着画图
   - **其他还有很多参数，参考 For more information：click this**

[10]:
```
data = [np.random.normal(0,std,100) for std in range(1,4)]  #正态分布，标准差在1～4循环
fig = plt.figure(figsize = (8,6))
plt.boxplot(data,notch = False,sym = 's',vert = True)
#notch = False不用特别形状，sym = 's'离群点为方块，vert = True竖着画图
plt.xticks([y+1 for y in range(len(data))],['x1','x2','x3'])   #指定刻度
plt.xlabel('xlabel_name',fontsize = 15)
plt.title('text_bos_plot',fontsize = 18)
```

[10]:
```
Text(0.5, 1.0, 'text_bos_plot')
```

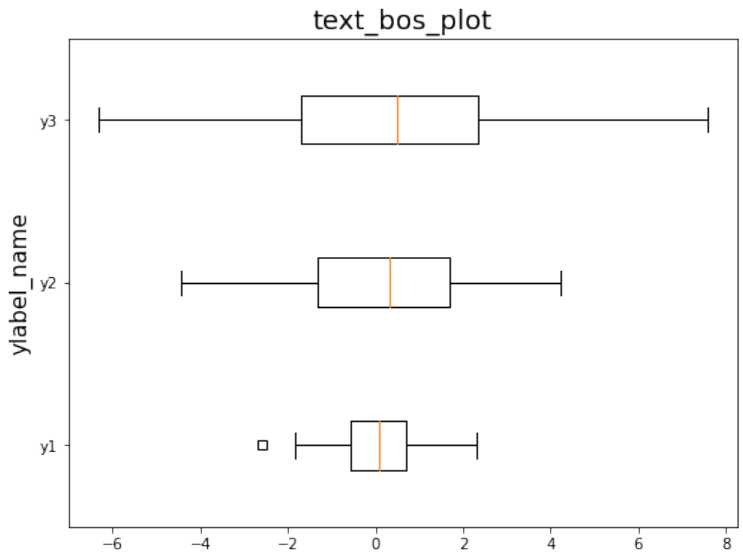


[25]:
```
#给上面的图加加工

data = [np.random.normal(0,std,100) for std in range(1,4)]  #正态分布，标准差在1~4循环
fig = plt.figure(figsize = (8,6))
box_plot = plt.boxplot(data,notch = False,sym = 's',vert = True)
#notch = False不用特别形状，sym = 's'离群点为方块，vert = True竖着画图

plt.xticks([y+1 for y in range(len(data))],['x1','x2','x3'])   #指定刻度
plt.xlabel('xlabel_name',fontsize = 15)
plt.title('text_bos_plot',fontsize = 18)

for component in box_plot.keys():
    for line in box_plot[component]:
        for c,box,lw in zip(['y','r','b'],box_plot['boxes'],[2,4,6]):
            box.set(color = c,linewidth = lw)
```

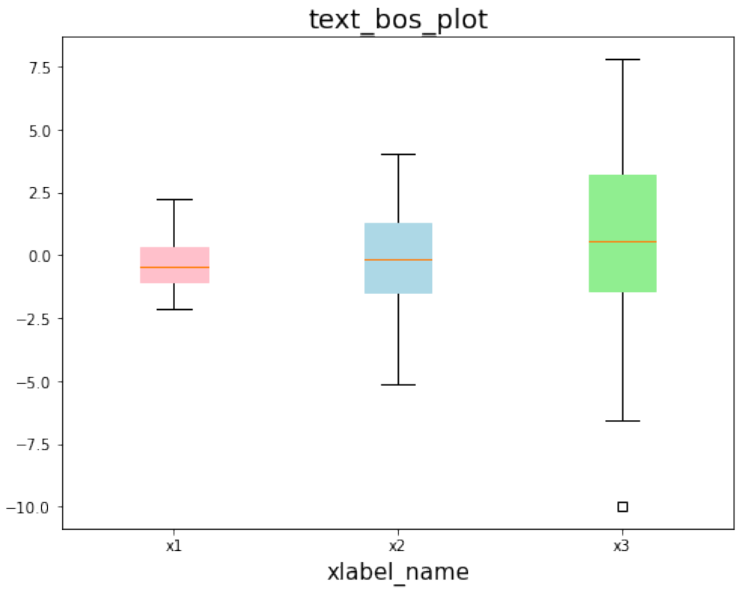


[15]:
```
#改成横着画
data = [np.random.normal(0,std,100) for std in range(1,4)]  #正态分布，标准差在1~4循环
fig = plt.figure(figsize = (8,6))
box_plot = plt.boxplot(data,notch = False,sym = 's',vert = False)
#notch = False不用特别形状，sym = 's'离群点为方块，vert = True竖着画图


plt.yticks([y+1 for y in range(len(data))],['y1','y2','y3'])   #指定刻度，之前是x，这里因为是横着画，旋转了90度，所以改成y
plt.ylabel('ylabel_name',fontsize = 15)
plt.title('text_bos_plot',fontsize = 18)
```

[15]:
```
Text(0.5, 1.0, 'text_bos_plot')
```



[19]:
```
#填充颜色
data = [np.random.normal(0,std,100) for std in range(1,4)]  #正态分布，标准差在1~4循环
fig = plt.figure(figsize = (8,6))
box_plot = plt.boxplot(data,notch = False,sym = 's',vert = True,patch_artist =␣
 ↪True)
#notch = False不用特别形状，sym = 's'离群点为方块，vert = True竖着画图

plt.xticks([y+1 for y in range(len(data))],['x1','x2','x3'])   #指定刻度
plt.xlabel('xlabel_name',fontsize = 15)
plt.title('text_bos_plot',fontsize = 18)

colors = ['pink','lightblue','lightgreen']
for patch,c in zip(box_plot['boxes'],colors):
    patch.set(color = c)
```



5 子图

- **有关 fig 和 axes 参数的意义，参考**：click this
  简单来说,fig 指定的是画布（整体）的各种参数画多大 | 而 axes 是指定画几个子图在这个画布里，怎么排列，画在哪里
  其中，fig 参数的具体设置参考：

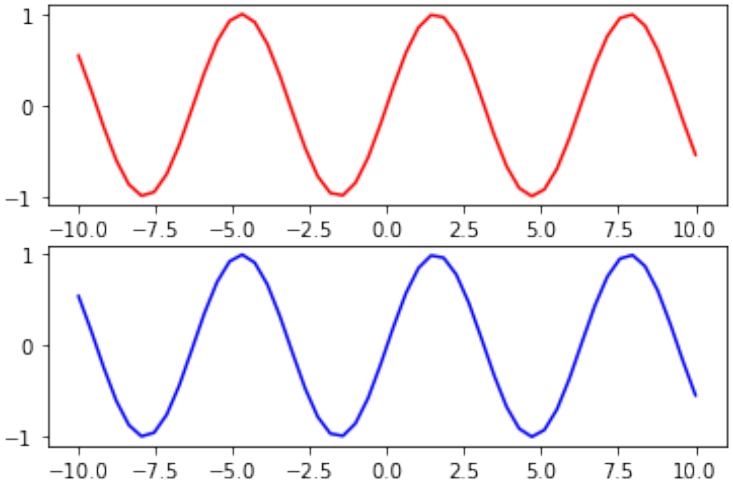
覚えておくと便利な `Figure` の引数を紹介します。

設定内容	引数名	引数の指定方法
描画領域のサイズ変更	`figsize`	`(width, height)` をインチで指定 デフォルト：`(6.4, 4.8)`
サブプロットのレイアウト自動調整	`tight_layout`	`True` or `False` デフォルト：`False` `True` にすると自動調整
描画領域の背景色変更	`facecolor`	色名などで指定 デフォルト：`'white'`
描画領域の枠線表示	`linewidth` `edgecolor`	`linewidth = 数値pt` `edgecolor = '色名など'`

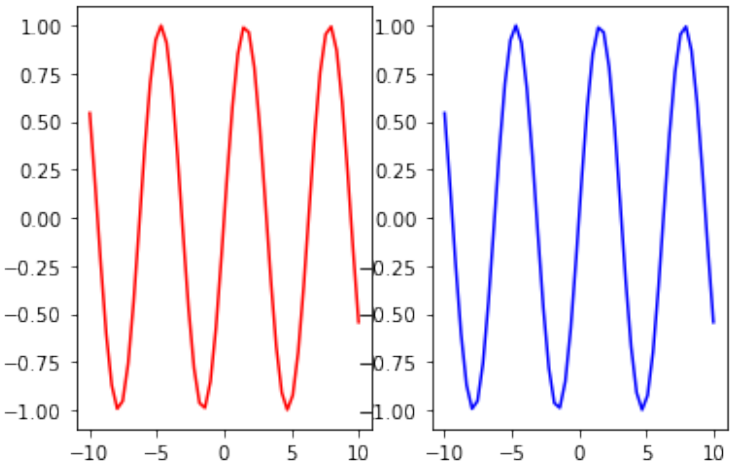
- **有关 `plt.sublots()`**：参考 **：click this
  - **`ig,axes = plt.sublots(nrows=2, ncols=3)`** 或者是 (2,3) 意为：
    * 図を縦に nrows 個、横に ncols 個に分割
    * axes に各 Axes オブジェクトを配列形式で格納
  - **`plt.sublots(nrows=2)`**：单独写也行，画两个图，竖着排
  - **`plt.sublots(figsize = (5,5))`**:figsize 是用来指定 fig 画布的大小的，5inch * 5inch 的感觉

- **在做子图的时候经常会用到 for 去循环，这时候想一次性指定多个 object 在某几个范围内去循环的话，就可以用 `zip()`**：
  - 比如 **`for name, age in zip(names, ages):`**

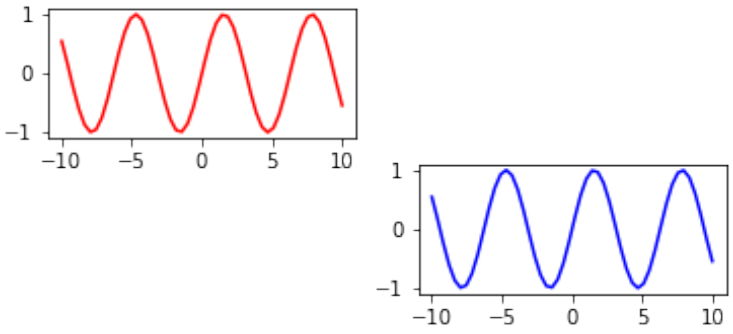
[3]:
```
plt.subplot(211) #211表示一会要画的图是2行1列，最后的1表达的是两个子图的第1个图，顺着就是211
plt.plot(x,y,color = 'r')
plt.subplot(212)#表示的是第二个
plt.plot(x,y,color = 'b')
#下面的图的排列就是正好是两行一列
```



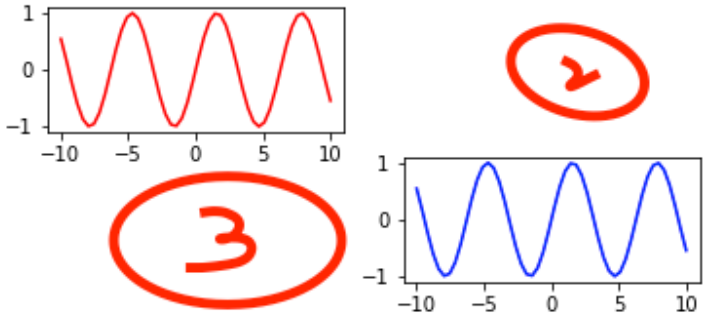
[4]:
```
plt.subplot(121) #改成一行两列看看
plt.plot(x,y,color = 'r')

plt.subplot(122)#表示的是第二个
plt.plot(x,y,color = 'b')
```



[5]:
```
plt.subplot(321) #改成3行2列看看
plt.plot(x,y,color = 'r')
plt.subplot(324)#表示的是 【第4个】
plt.plot(x,y,color = 'b')
#但是注意，是这样的，要画出来的应该是6个图（3行*2列），但是我们只指定了第1个子图和第4个子图
#所以剩下的没指定的就会空着
```



就是这个意思↓

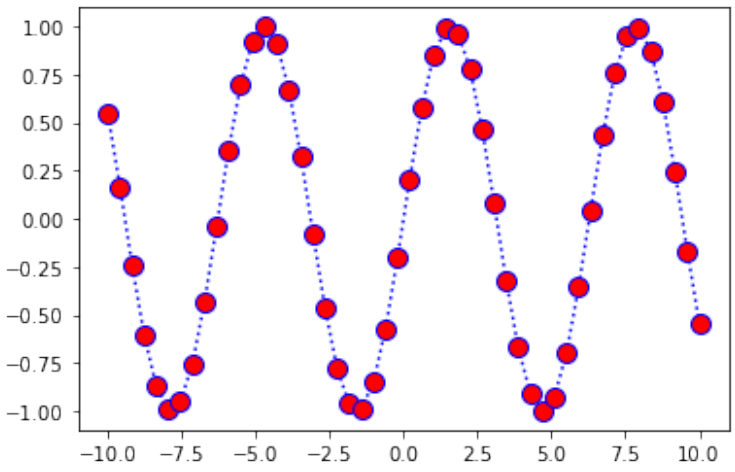


6 给图片增加注释

- **plt.text(10,1,'look here!',fontsize = 10,color = 'r')** ：在图中添加文本, 那个【10，1】是【横坐标 10，纵坐标 1】
- **plt.grid(True)**：添加网格
- **plt.annotate('hey!thisone!',xy=(2.5,-1),xytext=(2,-0.5), arrowprops = dict(facecolor = 'black',shrink = 1))**
  在图中添加箭头，**'hey!thisone!'**是用于描述箭头的文本，**xy=(2.5,-1)**是箭头指向哪里，**xytext=(2,-0.5)**是文本位置，**arrowprops=** 是箭头的参数，长度，颜色等

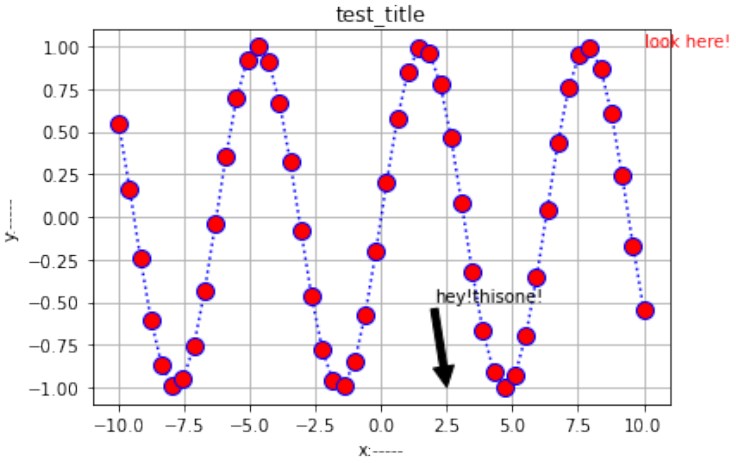
[7]:
```
plt.plot(x,y,color = 'b',linestyle = ':',marker = 'o',markerfacecolor =␣
↪'r',markersize = 10)
#放一下原图
```

[7]:
```
[<matplotlib.lines.Line2D at 0x7fe20c65d7c0>]
```



[24]:
```
plt.plot(x,y,color = 'b',linestyle = ':',marker = 'o',markerfacecolor =␣
 ↪'r',markersize = 10)
plt.xlabel('x:-----')
plt.ylabel('y:-----')
plt.title('test_title')
plt.text(10,1,'look here!',fontsize = 10,color = 'r')    #在图中添加文本
plt.grid(True)
plt.annotate('hey!thisone!',xy=(2.5,-1),xytext=(2,-0.5),arrowprops =␣
 ↪dict(facecolor = 'black',shrink = 1))
#在图中添加箭头
```

[24]:
```
Text(2, -0.5, 'hey!thisone!')
```

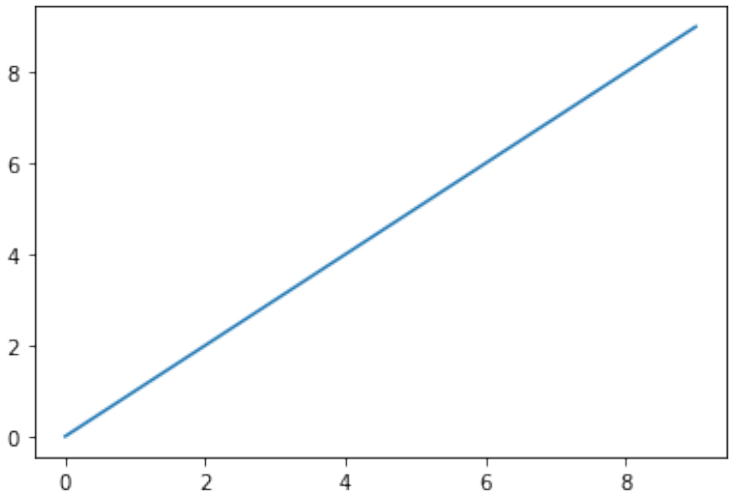


7 绘图细节

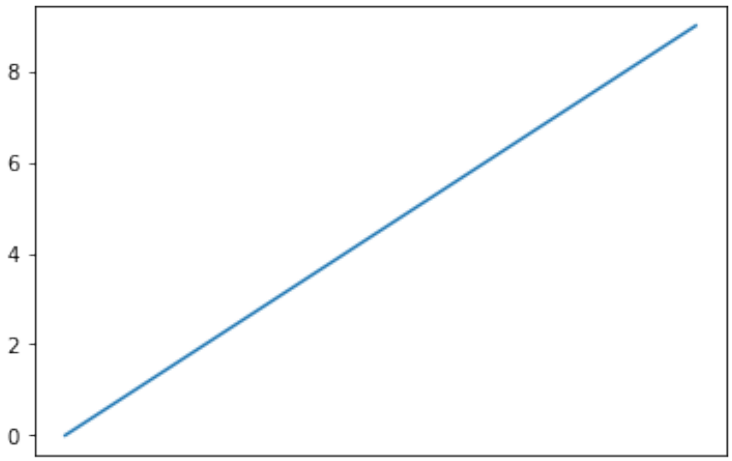
- **fig.axes.get_xaxis().set_visible(False)** : 去掉 x 轴刻度
- **fig.axes.get_yaxis().set_visible(False)** : 去掉 y 轴刻度
- **plt.set_xticklabels(labels,rotation = 45,horizontalalignment='right')** 设置 x 轴刻度的属性，比如 labels 是刻度上的文字内容。rotation 是轴刻度上文字内容旋转多少度，比如转个 45 度就能用于显示全一下显示不下的字。**horizontalalignment=** 是向什么方向对齐，向左对齐，居中对齐等等。
- **plt.legend(loc = 'best')** : 在图中加上对于线或者是柱子的说明，谁是什么颜色表示的是什么，**loc=** 是这个说明放在哪里，写 **'best'** 就是放在一个不会挡住图的不碍事的地方。还有很多参数，具体参考

[3]:
```
x = range(10)
y = range(10)

fig = plt.gca()
plt.plot(x,y)

#先看一下原图
```

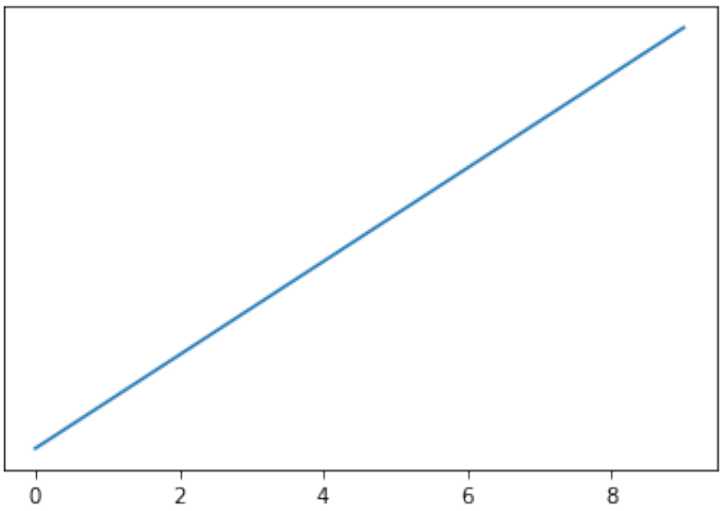
[3]:
```
[<matplotlib.lines.Line2D at 0x7f9ce21affd0>]
```



[6]:
```
fig = plt.gca()
plt.plot(x,y)
fig.axes.get_xaxis().set_visible(False)

#把 x 轴的刻度去掉
```

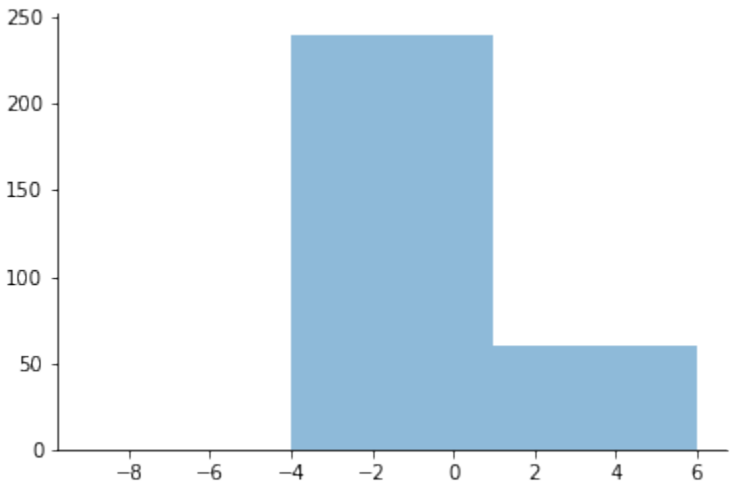


[7]:
```
fig = plt.gca()
plt.plot(x,y)
fig.axes.get_yaxis().set_visible(False)

#把 y 轴的刻度去掉
```



[17]:
```
import math
x = np.random.normal(loc = 0.0 , scale = 1.0 , size = 300)
#loc设置平均值，scale设置标准差，size设置形状（单数字x就是x个，写（x，y）这样的就是矩阵）
#所以意为：从标准正态分布取300个数
width = 5
bins = np.arange(math.floor(x.min())-width,math.ceil(x.max())+width,width)

ax = plt.subplot(111)

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.hist(x,alpha = 0.5,bins = bins)

#设置图的四个边框（轴），让右边的和顶上的不可见
```

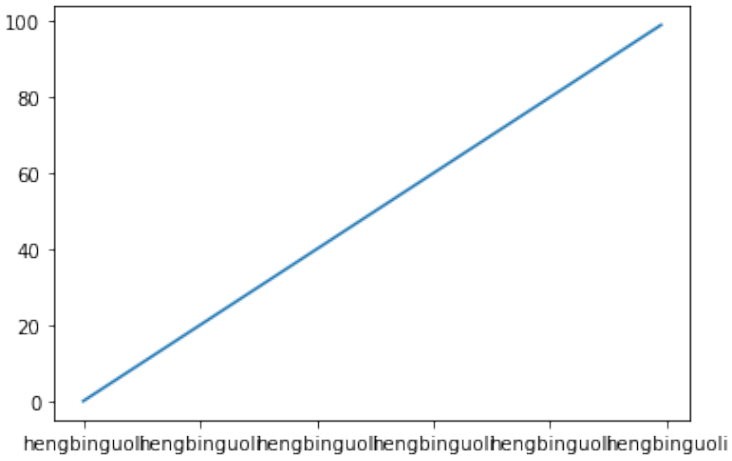
[17]:
```
(array([  0., 240.,  60.]),
 array([-9, -4,  1,  6]),
 <BarContainer object of 3 artists>)
```



[24]:
```
x = range(100)
y = range(100)

labels = ['hengbinguoli' for i in range(10)]
fig,ax = plt.subplots()
plt.plot(x,y)

ax.set_xticklabels(labels)

#可以看到下面图中的x轴很奇怪，刻度很拥挤显示的不好看
```

```
/var/folders/yk/g6zs19fd5jz847sc8_ndgs240000gn/T/ipykernel_23311/3923351342.py:8
: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(labels)
```

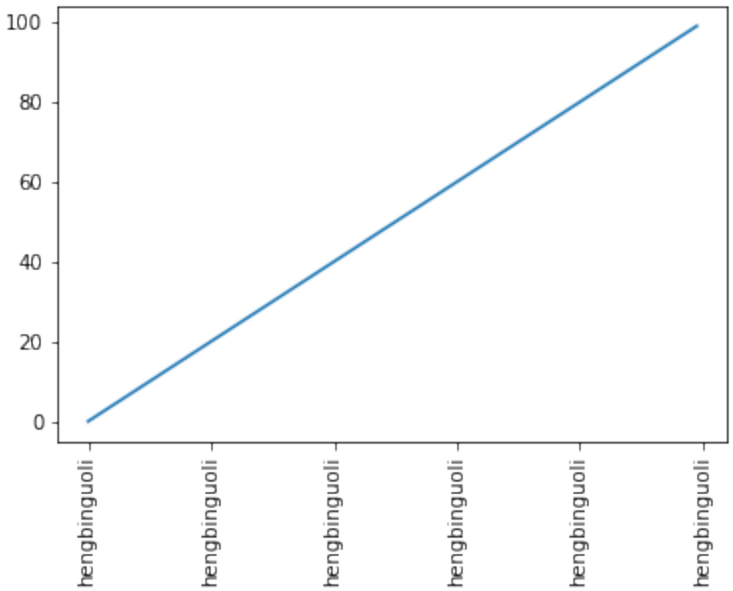
[24]:
```
[Text(-20.0, 0, 'hengbinguoli'),
 Text(0.0, 0, 'hengbinguoli'),
 Text(20.0, 0, 'hengbinguoli'),
 Text(40.0, 0, 'hengbinguoli'),
 Text(60.0, 0, 'hengbinguoli'),
 Text(80.0, 0, 'hengbinguoli'),
 Text(100.0, 0, 'hengbinguoli'),
 Text(120.0, 0, 'hengbinguoli')]
```



[25]:
```
fig,ax = plt.subplots()
plt.plot(x,y)

ax.set_xticklabels(labels,rotation = 90)
# 竖着标 x 轴的刻度
```

```
/var/folders/yk/g6zs19fd5jz847sc8_ndgs240000gn/T/ipykernel_23311/3703574557.py:4
: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(labels,rotation = 90)
```

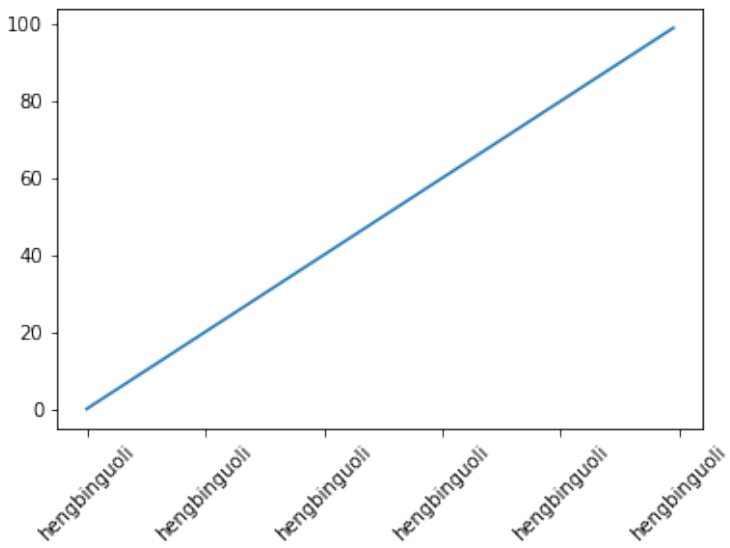
[25]:
```
[Text(-20.0, 0, 'hengbinguoli'),
 Text(0.0, 0, 'hengbinguoli'),
 Text(20.0, 0, 'hengbinguoli'),
 Text(40.0, 0, 'hengbinguoli'),
 Text(60.0, 0, 'hengbinguoli'),
 Text(80.0, 0, 'hengbinguoli'),
 Text(100.0, 0, 'hengbinguoli'),
 Text(120.0, 0, 'hengbinguoli')]
```



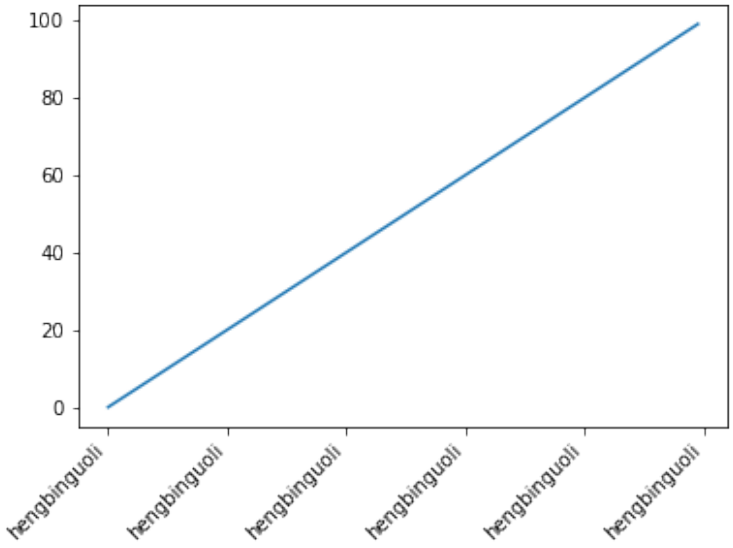
[26]:
```
fig,ax = plt.subplots()
plt.plot(x,y)

ax.set_xticklabels(labels,rotation = 45)
#斜着标x轴的刻度，但是这个刻度没对齐，就有歧义
```

```
/var/folders/yk/g6zs19fd5jz847sc8_ndgs240000gn/T/ipykernel_23311/1947170141.py:4
: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(labels,rotation = 45)
```

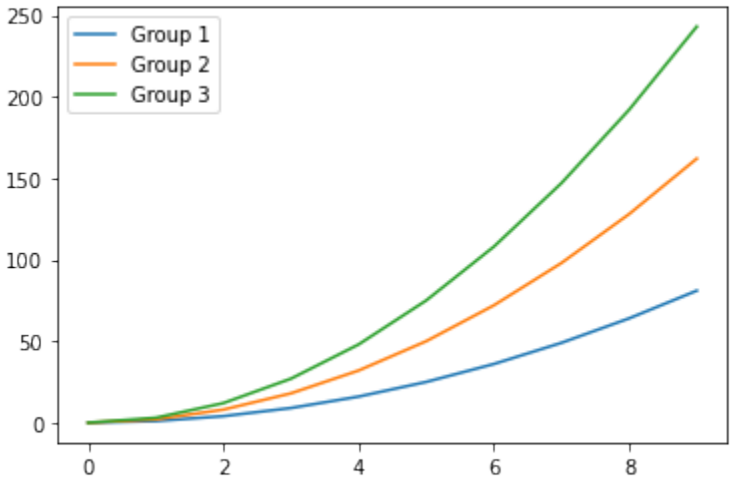
[26]:
```
[Text(-20.0, 0, 'hengbinguoli'),
 Text(0.0, 0, 'hengbinguoli'),
 Text(20.0, 0, 'hengbinguoli'),
 Text(40.0, 0, 'hengbinguoli'),
 Text(60.0, 0, 'hengbinguoli'),
 Text(80.0, 0, 'hengbinguoli'),
 Text(100.0, 0, 'hengbinguoli'),
 Text(120.0, 0, 'hengbinguoli')]
```



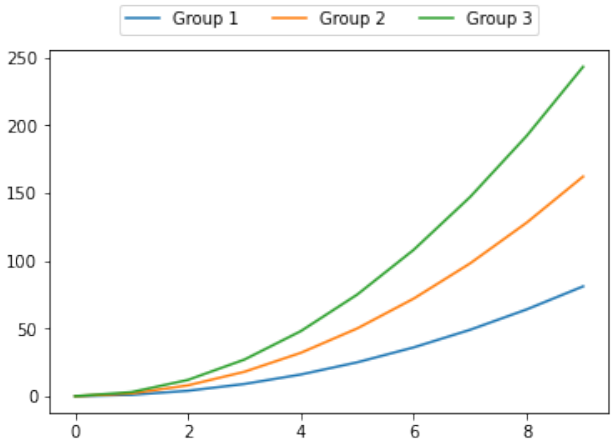
[27]:
```
fig,ax = plt.subplots()
plt.plot(x,y)

ax.set_xticklabels(labels,rotation = 45,horizontalalignment='right')
#加上一个对齐参数这样就好多了
```



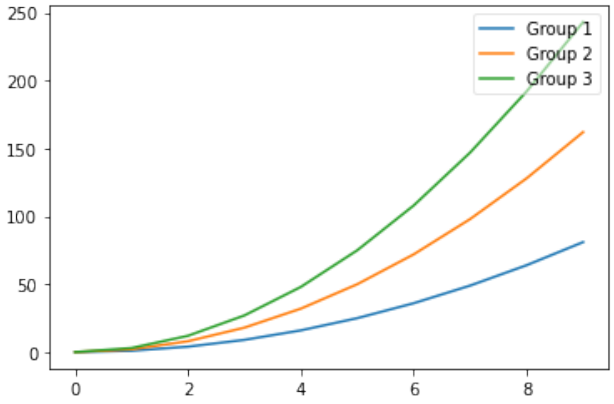
[3]:
```
x = np.arange(10)
for i in range(1,4):
    plt.plot(x,i*x**2,label='Group %d'%i)
plt.legend()  #加一个表示说明线表达的是什么
```



[6]:
```
fig,ax = plt.subplots()

for i in range(1,4):
    plt.plot(x,i*x**2,label='Group %d'%i)
ax.legend(loc = 'upper center',bbox_to_anchor = (0.5,1.15),ncol = 3)
#bbox_to_anchor是配合 loc 写放哪里，一个指定坐标的感觉，如果没写 loc，就是按照图中数据的原点作为原点，如果写了 loc，就按 loc 的参数作为原点
#ncol 指定图例放几列
```



[10]:
```
fig,ax = plt.subplots()

for i in range(1,4):
    plt.plot(x,i*x**2,label='Group %d'%i)
ax.legend(loc = 'upper right',framealpha = 0.5)   #framealpha,指定图例的框框的透明度，1是完全不透明，0是完全透明
```

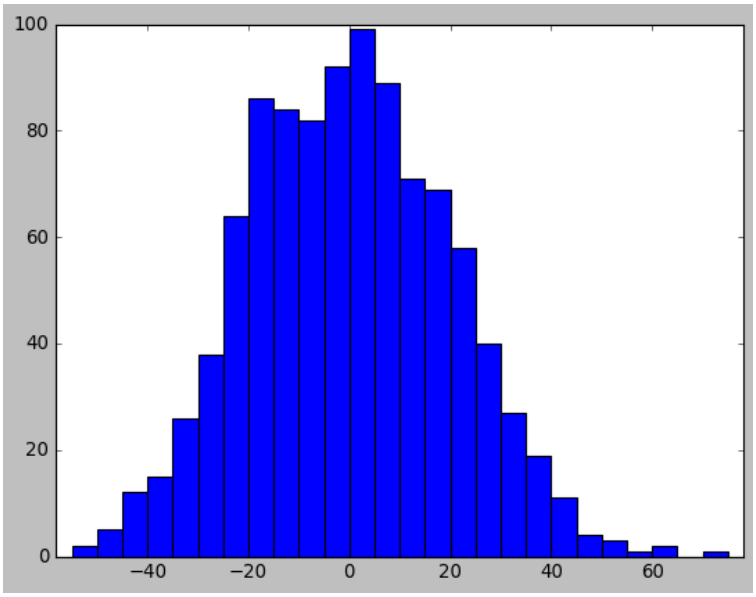


8 直方图

- 普通绘制直方图:
  plt.hist(数据的 array,(间隔参数)bins=,alpha= ...) 详见: click this, **alpha=** 参数设置透明度, 文档里没有
- 设置一下画图的区间范围:
  plt.xlim(区间下界, 区间上界,...) 详见: click this
- 改柱子颜色, 宽度, 包边颜色, 花纹等的**万能设置函数**: 详见: click this
  bar.set(color = 'red',edgecolor = 'black',linewidth = 3,hatch = ...), 基本上什么都能设置
  只要' 图的名字' 或者是' 子图的名字'.set 就行了, 想改什么参数就往里面传就行。
  python 内查看帮助文档 **help(matplotlib.patches.Rectangle.set)**

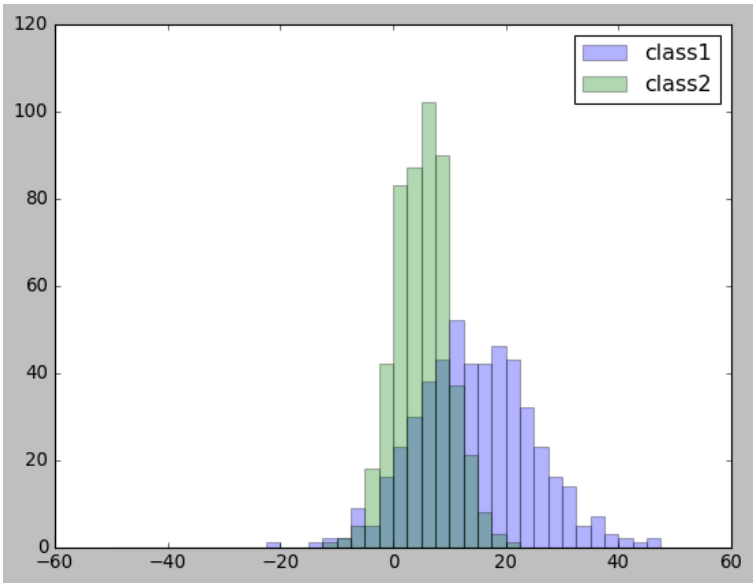
[5]:
```
plt.style.use('classic')  #设置下风格
data = np.random.normal(0,20,1000)
bins = np.arange(-100,100,5)

plt.hist(data,bins = bins)
plt.xlim([min(data)-5,max(data)+5])  #设置一下画图的区间范围
```

[5]: (-58.20859885178959, 78.05794725093352)



[8]:
```
#把两个直方图叠一起

import random  #用于生成随机数的模块
data1 = [random.gauss(15,10) for i in range(500)] #guass，中文：高斯，这个命令的意思是平均值 15，标准差 10 的正态分布取 500 个数
data2 = [random.gauss(5,5) for i in range(500)]

bins = np.arange(-50,50,2.5)

plt.hist(data1,bins = bins,label = 'class1',alpha = 0.3)  #alpha 参数设置透明度
plt.hist(data2,bins = bins,label = 'class2',alpha = 0.3)
plt.legend(loc='best')
```

[8]:
```
<matplotlib.legend.Legend at 0x7fafaf45fc10>
```



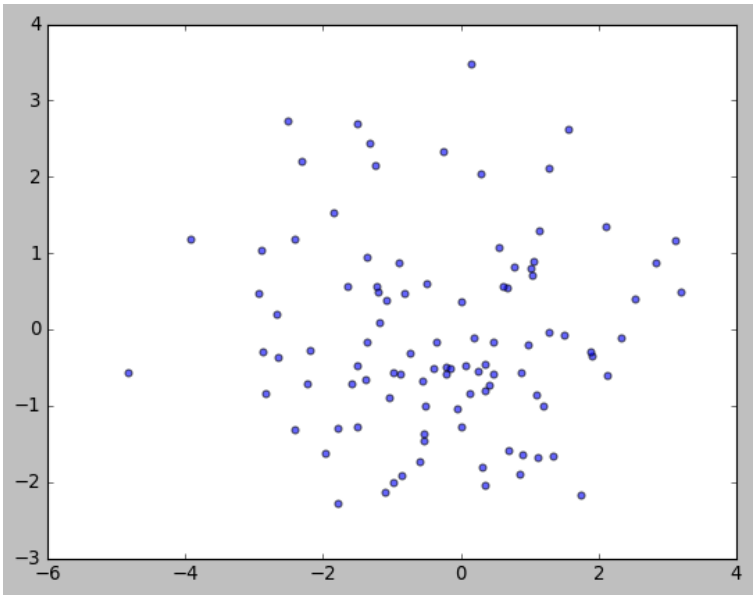
9 散点图

- 普通绘制直方图：
  plt.scatter(数据点的 x 坐标, 数据点的 y 坐标,alpha= ...) 详见：click this，**alpha=** 参数设置透明度，文档里没有

- 设置一下画图的区间范围：
  plt.xlim(区间下界, 区间上界,...) 详见：click this

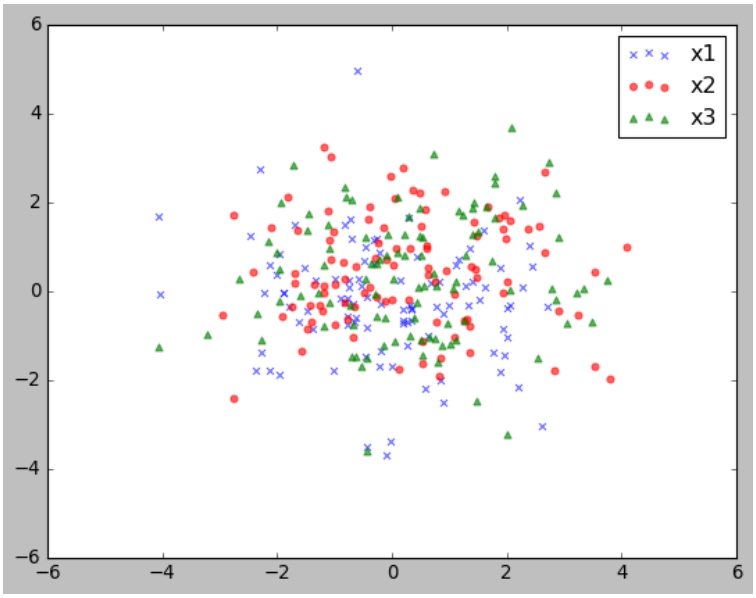
- 改柱子颜色，宽度，包边颜色, 花纹等的**万能设置函数**：详见：click this
  bar.set(color = 'red',edgecolor = 'black',linewidth = 3,hatch = ...)，基本上什么都能设置
  只要 **' 图的名字'** 或者是 **' 子图的名字'.set** 就行了，想改什么参数就往里面传就行。
  python 内查看帮助文档 **help(matplotlib.patches.Rectangle.set)**

[17]:
```
#散点图是要数据点是二维的，因此可以用二维正态分布取点
mu_vec1 = np.array([0,0])  #设置平均值
cov_mat1 = np.array([[2,0],[0,2]])  #设置协方差矩阵
x1 = np.random.multivariate_normal(mu_vec1,cov_mat1,100)
#注意，多重正态要求传的就不是标准差，而是协方差了，同时是 n 重就传 n 维的正方形的协方差矩阵
plt.figure(figsize = (8,6))
plt.scatter(x1[:,0],x1[:,1],alpha = 0.6,label='x1')  #alpha= 参数设置透明度
#可以看出他们没什么相关性
```

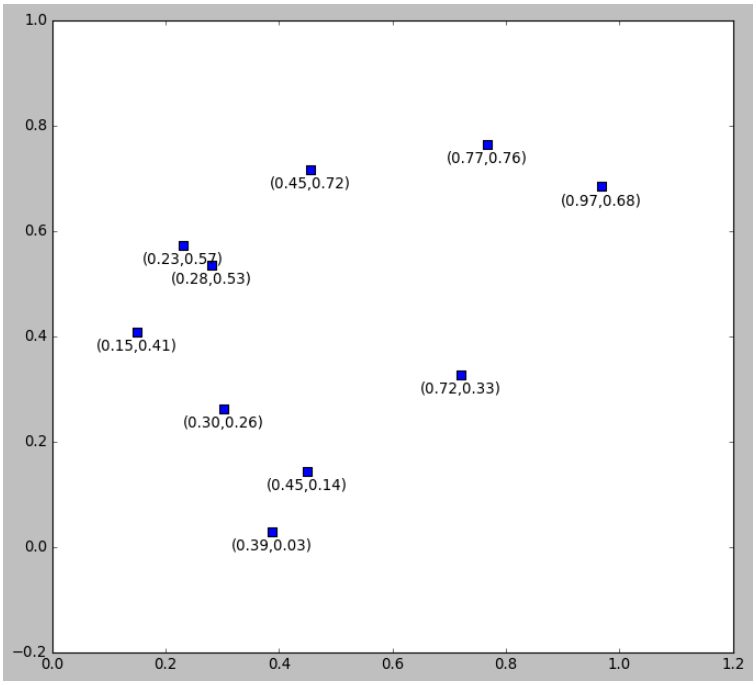


[18]:
```
mu_vec1 = np.array([0,0])
cov_mat1 = np.array([[2,0],[0,2]])

x1 = np.random.multivariate_normal(mu_vec1,cov_mat1,100)
x2 = np.random.multivariate_normal(mu_vec1+0.2,cov_mat1+0.2,100)
x3 = np.random.multivariate_normal(mu_vec1+0.4,cov_mat1+0.4,100)

plt.figure(figsize = (8,6))
plt.scatter(x1[:,0],x1[:,1],marker = 'x',color = 'blue',alpha = 0.
↪6,label='x1')#alpha=参数设置透明度
plt.scatter(x2[:,0],x2[:,1],marker = 'o',color = 'red',alpha = 0.6,label='x2')
plt.scatter(x3[:,0],x3[:,1],marker = '^',color = 'green',alpha = 0.6,label='x3')

plt.legend(loc='best')
```

[18]:
```
<matplotlib.legend.Legend at 0x7fafb0a232b0>
```

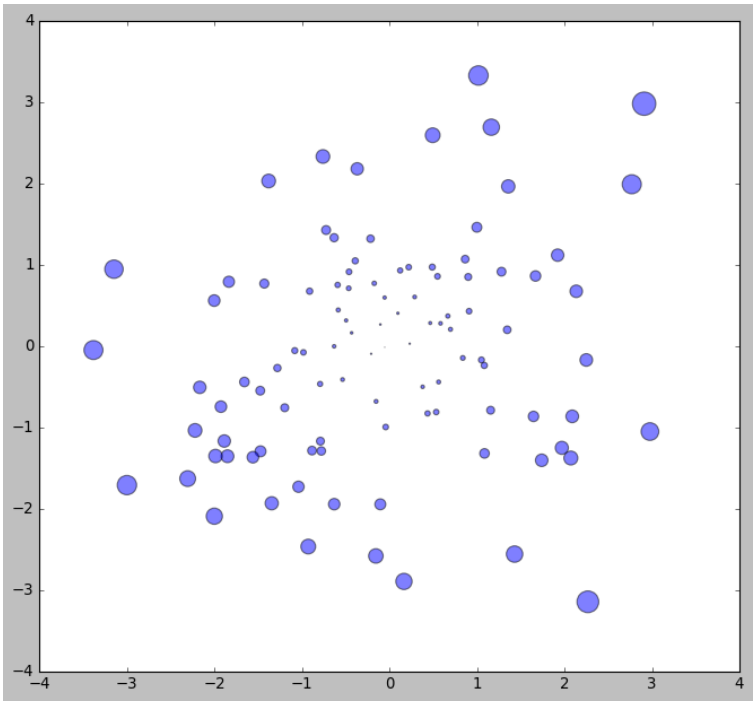


[37]:
```
x_records = np.random.random(10)
y_records = np.random.random(10)

plt.figure(figsize = (10,9))
plt.scatter(x_records,y_records,marker = 's',s=50)  #s=,指定图中点的大小

for x,y in zip(x_records,y_records):
    plt.annotate('({:.2f},{:.2f})'.format(x,y),xy = (x,y),xytext =␣
 ↪(0,-15),textcoords = 'offset points',ha = 'center')
    #在点的下面标注坐标，并且设置取小数点后两位
```



[41]:
```
mu_vec1 = np.array([0,0])
cov_mat1 = np.array([[2,0],[0,2]])

X = np.random.multivariate_normal(mu_vec1,cov_mat1,100)
fig = plt.figure(figsize=(10,9))

R = X**2 #取平方
R_sum = R.sum(axis = 1)

plt.scatter(X[:,0],X[:,1],s = 20*R_sum,alpha = 0.5)
#从（0，0）圆心开始向外扩展，越往外点越大，点的大小是按指数函数取的
```

[41]:
```
<matplotlib.collections.PathCollection at 0x7fafb33064c0>
```

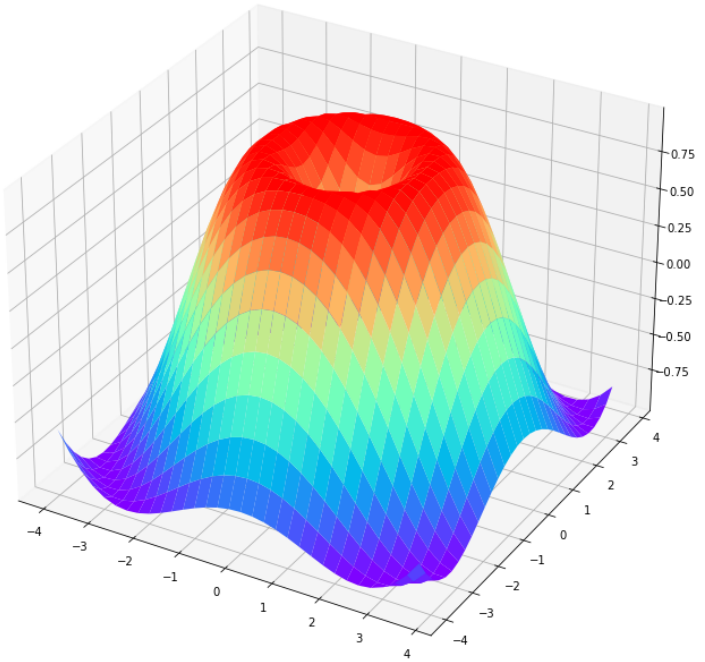


10 三维做图

0. 先加载库: **from mpl_toolkits.mplot3d import Axes3D**

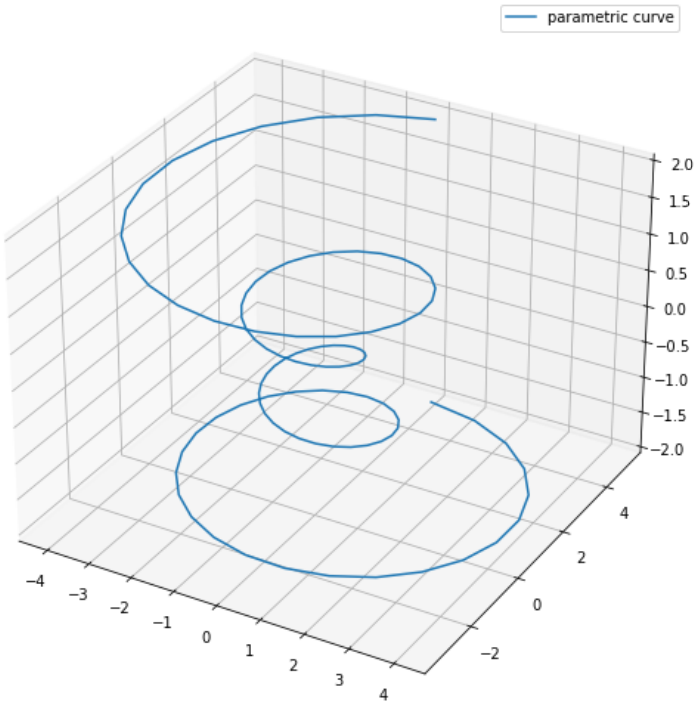
- 在用 **plt.figure** 构造面板时，直接传入 3d 参数 (**projection = '3d'**)，使其变成三维面板: **fig = plt.figure(figsize=(10,9)) ax = plt.subplot(111,projection = '3d')**

[16]:
```
from mpl_toolkits.mplot3d import Axes3D    #加载三维模块
fig = plt.figure(figsize=(10,9))    #构成一个二维的图面板
ax = Axes3D(fig,auto_add_to_figure=False)      #把二维面板传进加载的 Axes3D 模块里，转化成三维的
fig.add_axes(ax)

x = np.arange(-4,4,0.25)
y = np.arange(-4,4,0.25)

X,Y = np.meshgrid(x,y)  #生成网格

Z = np.sin(np.sqrt(X**2 + Y**2))    #np.sqrt 求平方根，生成 Z 轴的值
ax.plot_surface(X,Y,Z,rstride = 1,cstride = 1,cmap = 'rainbow')
```



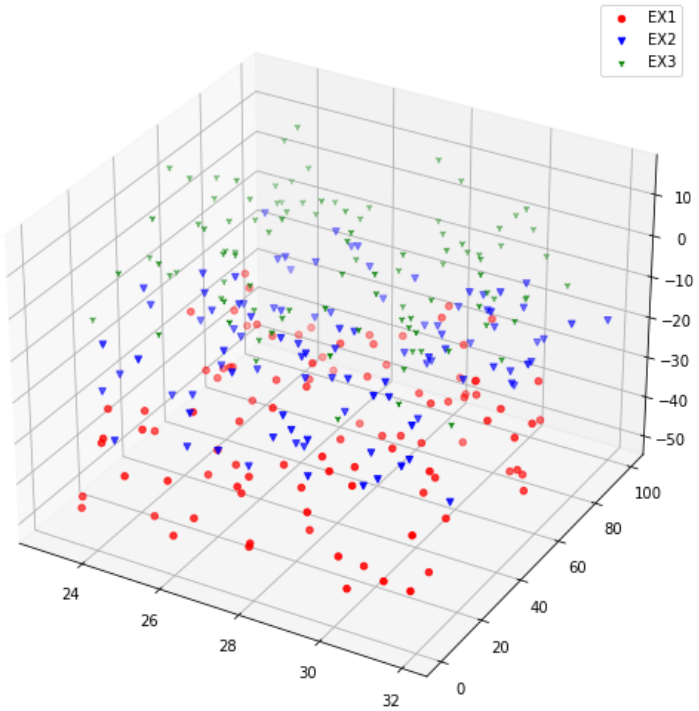
[26]:
```
fig = plt.figure(figsize=(10,9))
ax = plt.subplot(111,projection = '3d')  #也可以用 plt.subplot 来传递 3d 参数

theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
z = np.linspace(-2, 2, 100)
r = z**2 + 1
x = r * np.sin(theta)
y = r * np.cos(theta)
ax.plot(x, y, z, label='parametric curve')
ax.legend(loc = 'best')
```

[26]:
```
<matplotlib.legend.Legend at 0x7feebd6b6070>
```

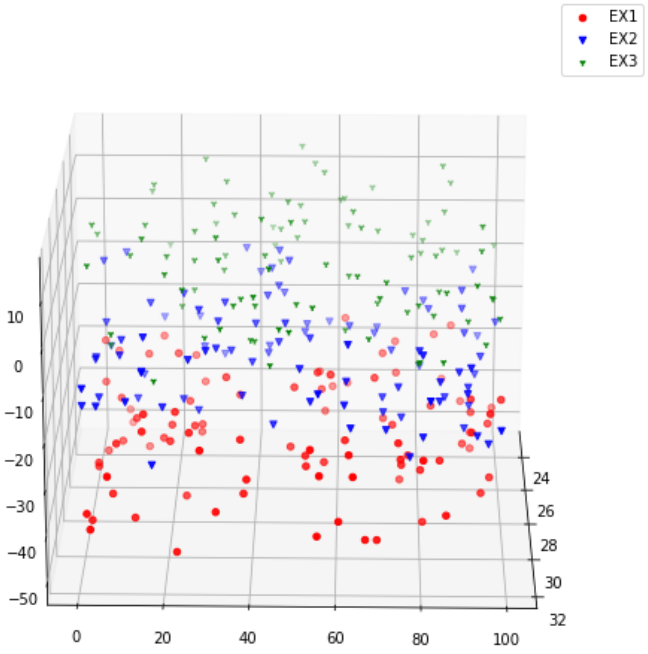


[28]:
```
#画三维的散点图
np.random.seed(1)  #打破次次随机
def randrange(n,vmin,vmax):
    return (vmax-vmin)*np.random.rand(n)+vmin     #np.random.rand:0~1之间的均匀分布


fig = plt.figure(figsize=(10,9))
ax = plt.subplot(111,projection = '3d')
n = 100
for color,marker,zmin,zmax,label in␣
 ↪zip(['r','b','g'],['o','v','1'],[-50,-30,-10],[-25,-5,15],['EX1','EX2','EX3']):
 ↪
    xs = randrange(n,23,32)
    ys = randrange(n,0,100)
    zs = randrange(n,zmin,zmax)
    ax.scatter(xs,ys,zs,color = color,marker = marker,label = label)
ax.legend(loc = 'best')
```

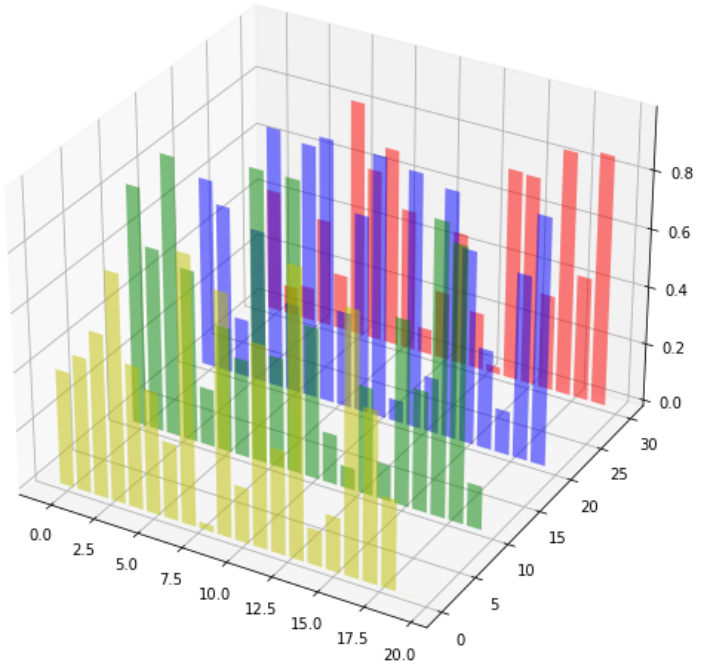
[28]: <matplotlib.legend.Legend at 0x7feebe166100>



[22]:
```
#把上面的图转个角度
np.random.seed(1)  #打破次次随机
def randrange(n,vmin,vmax):
    return (vmax-vmin)*np.random.rand(n)+vmin     #np.random.rand:0~1之间的均匀分布


fig = plt.figure(figsize=(10,9))
ax = fig.subplot(111,projection = '3d')
n = 100
for color,marker,zmin,zmax,label in␣
 ↪zip(['r','b','g'],['o','v','1'],[-50,-30,-10],[-25,-5,15],['EX1','EX2','EX3']):
 ↪
    xs = randrange(n,23,32)
    ys = randrange(n,0,100)
    zs = randrange(n,zmin,zmax)
    ax.scatter(xs,ys,zs,color = color,marker = marker,label = label)
ax.legend(loc = 'best')
ax.view_init(20,1)       #转角度的命令
```



[34]:
```
#画三维条形图
fig = plt.figure(figsize=(10,9))
ax = plt.subplot(111,projection = '3d')

for color,z_value in zip(['r','b','g','y'],[30,20,10,0]):
    xs = np.arange(20)
    ys = np.random.rand(20)
    ax.bar(xs,ys,zs = z_value,zdir = 'y',color = color,alpha = 0.5)
```



11 Pie 図

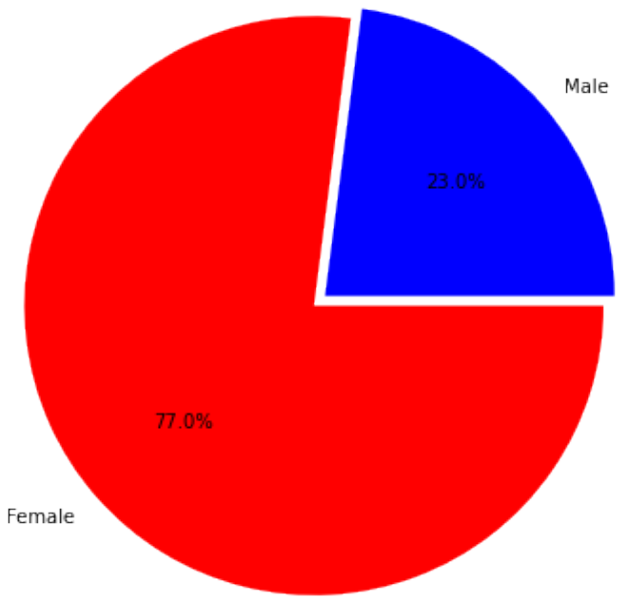
- **matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1,startangle=None, radius=None, counterclock=True, wedgeprops=None, textprops=None, center=(0, 0),frame=False, hold=None, data=None)**

- **matplotlib.pyplot.pie** の主要な引数 **'x'** (必須) 各要素の大きさを配列で指定。
  explode= 各要素を中心から離して目立つように表示。
  labels= 各要素のラベル。
  colors= 各要素の色を指定。
  autopct= 構成割合をパーセンテージで表示。(デフォルト値: **None**)
  pctdistance= 上記のパーセンテージを出力する位置。円の中心 **0.0** から円周 **1.0** を目安に指定。**autopct** を指定した場合のみ有効。(デフォルト値: **0.6**)
  shadow= True に設定すると影を表示。(デフォルト値: **False**)
  labeldistance= ラベルを表示する位置。円の中心 0.0 から円周 1.0 を目安に指定。(デフォルト値: **1.1**)
  startangle= 各要素の出力を開始する角度。(デフォルト値: **None**)
  radius= 円の半径。(デフォルト値: **1**)
  counterclock= True に設定すると時計回りで出力。**False** に設定すると反時計回りで出力。(デフォルト値: **True**)
  wedgeprops= ウエッジ (くさび形の部分) に関する指定。枠線の太さなどを設定可能。(デフォルト値: **None**)
  textprops= テキストに関するプロパティ。(デフォルト値: **None**)

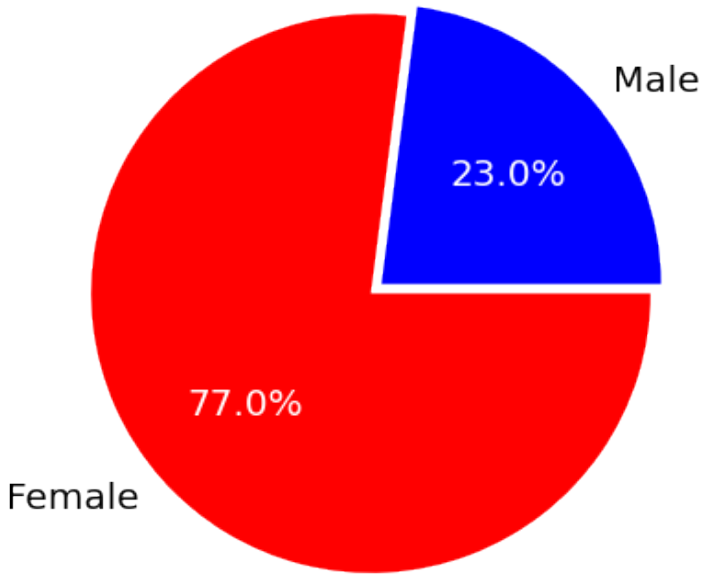
详细参考 click

[12]:
```
m = 1345
f = 4513

m_per = m/(m+f)  #取得百分比
f_per = f/(m+f)

colors = ['b','r']
labels = ['Male','Female']
plt.figure(figsize = (7,7))
plt.pie([m_per,f_per],colors = colors,labels = labels,autopct = '%1.
 ↪1f%%',explode = [0,0.05])

#autopct = '%1.1f%%',explode = [0,0.05]
```

[12]:
```
([<matplotlib.patches.Wedge at 0x7ff0dceafb80>,
  <matplotlib.patches.Wedge at 0x7ff0dcf67340>],
 [Text(0.8260344044875126, 0.7264070226828486, 'Male'),
  Text(-0.8635814228733089, -0.7594255237138869, 'Female')],
 [Text(0.4505642206295523, 0.39622201237246285, '23.0%'),
  Text(-0.4881112390153485, -0.4292405134035013, '77.0%')])
```

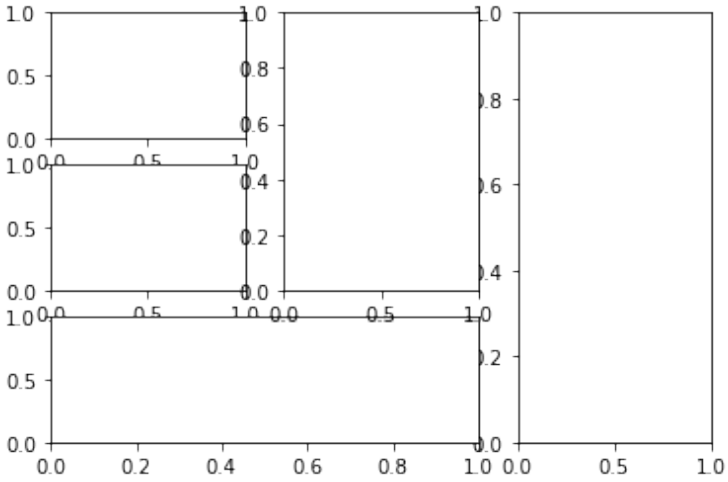


[4]:
```
m = 1345
f = 4513

m_per = m/(m+f)
f_per = f/(m+f)

colors = ['b','r']
labels = ['Male','Female']
plt.figure(figsize = (7,7))
paches,texts,autotexts = plt.pie([m_per,f_per],colors = colors,labels =␣
 ↪labels,autopct = '%1.1f%%',explode = [0,0.05])
#为了下面改颜色和字的大小，取一下参数

for text in texts+autotexts:
    text.set(fontsize = 20)

for text in autotexts:
    text.set(color = 'white')
```



12 设置子图布局

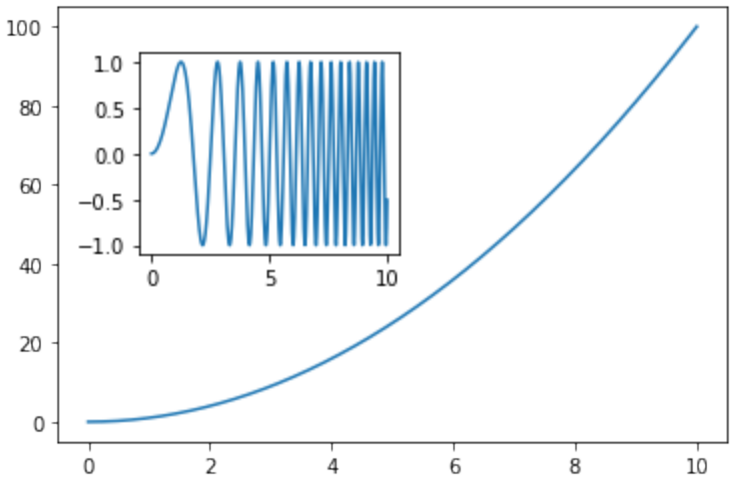
[8]:
```
#第一个（3，3）值的是这个图有 3x3 的布局，然后 (0,0) 指的是这个图在第 0 行的第 0 列
ax1 = plt.subplot2grid((3,3),(0,0))
ax2 = plt.subplot2grid((3,3),(1,0))
ax3 = plt.subplot2grid((3,3),(0,2),rowspan = 3)  #rowspan = 3 往下占 3 个行的位置
ax4 = plt.subplot2grid((3,3),(2,0),colspan = 2)  #colspan = 2 往右占 2 个列的位置
ax5 = plt.subplot2grid((3,3),(0,1),rowspan = 2)
```



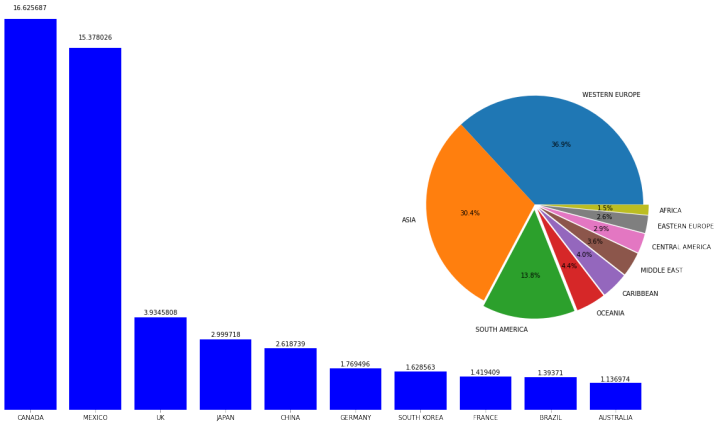
12.1 嵌套图

[10]:
```
x = np.linspace(0,10,1000)
y2 = np.sin(x**2)
y1 = x**2

fig,ax1 = plt.subplots()

left,bottom,width,height = [0.22,0.45,0.3,0.35]
ax2 = fig.add_axes([left,bottom,width,height])

ax1.plot(x,y1)
ax2.plot(x,y2)
```

[10]:
```
[<matplotlib.lines.Line2D at 0x7fac8d203be0>]
```



[44]:
```
from mpl_toolkits.axes_grid1.inset_locator import inset_axes

top10_arrivals_countries = ['CANADA','MEXICO','UK',
                            'JAPAN','CHINA','GERMANY','SOUTH KOREA',
                            'FRANCE','BRAZIL','AUSTRALIA']
top10_arrivals_values = [16.625687,15.378026,3.9345808,2.999718,
                         2.618739,1.769496,1.628563,1.419409,
                         1.393710,1.136974]
arrivals_countries = ['WESTERN EUROPE','ASIA','SOUTH AMERICA',
                      'OCEANIA','CARIBBEAN','MIDDLE EAST',
                      'CENTRAL AMERICA','EASTERN EUROPE','AFRICA']
arrivals_percent = [36.9,30.4,13.8,4.4,4.0,3.6,2.9,2.6,1.5]

fig,ax1 = plt.subplots(figsize = (20,12))
ax1_bar = ax1.bar(range(10),top10_arrivals_values,color = 'b')
plt.yticks([])#去掉 y 轴刻度值
ax1.set_xticks(range(10))          #把刻度改成 10 个值
ax1.set_xticklabels(top10_arrivals_countries)  #把刻度代入
#在柱子上头标注值
for rect in ax1_bar:
    height = rect.get_height()
    ax1.text(rect.get_x() + rect.get_width()/2 , 1.02*height,"{:,}".
 ↪format(float(height))
             ,ha = 'center',va = 'bottom')

ax2 = inset_axes(ax1,width = 6,height = 6, loc = 5)
explode = (0.0,0.0,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05)
ax2.pie(arrivals_percent,labels = arrivals_countries,autopct = '%1.
 ↪1f%%',explode = explode)

#把轴的线去掉
for spines in ax1.spines.values():
    spines.set(visible = False)
```


16.625687
15.378026
3.9345808
2.999718
2.618739
1.769496
1.628563
1.419409
1.39371
1.136974
CANADA
MEXICO
UK
JAPAN
CHINA
GERMANY
SOUTH KOREA
FRANCE
BRAZIL
AUSTRALIA
WESTERN EUROPE
36.9%
ASIA
30.4%
SOUTH AMERICA
13.8%
OCEANIA
4.4%
CARIBBEAN
4.0%
MIDDLE EAST
3.6%
CENTRAL AMERICA
2.9%
EASTERN EUROPE
2.6%
AFRICA
1.5%