在…里 [1]:

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
```

在…里 [2]:

```
df=pd.read_excel(r"D:\zhijiao\code-math.xlsx")
df
```

Out[2]:

	不	标题	密码
0	1	1.1.10立体几何万能解题（文科）	1086
1	2	1.3 一节课搞定百分之九十圆锥曲线计算量	1639
2	3	1.3.48 圆锥曲线 向量问题3	1175
3	4	1.3.87 数学秒杀技巧-轮换对称式	1335
4	5	1.3.10 函数专题精讲	1414
...	...	...	...
307	308	数学 全刷模拟题	1163
308	309	数学 圆锥曲线 角度问题	1173
309	310	数学-浙江卷导数专项习题———试听课	1356
310	311	直线、平面垂直的判定及性质定理——试听课	1431
311	312	直线、平面平行的判定及其性质——试听	1411

312行× 3列

在…里 [9]:

```
df=pd.read_excel(r"D:\zhijiao\code-math.xlsx")
df['title']=df['title'].str.replace('数学[ |-|—]+','')
df['title']=df['title'].str.replace('^[0123456789.]+','')
df['title']=df['title'].str.replace(' ','')
df['title']=df['title'].str.replace('[试听课|试听]','')
df['title']=df['title'].str.replace('[-] ','')
df['title']=df['title'].str.replace('[——]+ ','')
df['title']=df['title'].str.replace('[(|)] ','')
df['title']=df['title'].str.replace('[.] ','')
df['title']=df['title'].str.replace('[mp4]+','')
df
```

Out[9]:

	不	标题	密码
0	1	立体几何万能解题（文科）	1086
1	2	一节搞定百分之九十圆锥曲线计算量	1639
2	3	8圆锥曲线向量问题3	1175
3	4	87数学秒杀技巧-轮换对称式	1335
4	5	函数专题精讲	1414
...	...	...	...
307	308	全刷模拟题	1163
308	309	圆锥曲线角度问题	1173
309	310	数学-浙江卷导数专项习题———	1356
310	311	直线、平面垂直的判定及性质定理——	1431
311	312	直线、平面平行的判定及其性质——	1411

312行× 3列

在…里 [7]:

```
df.drop_duplicates('title',keep='last',inplace=True)
df.index=range(1,df.shape[0]+1)
```

在…里 [8]:

```
df[['title','code']].to_excel("result_2.xlsx")
```

在…里 [ ]:

```

```

在…里 [ ]:

```

```

在…里 [ ]:

```
```

在…里 [ ]:

```
```

在…里 [4]:

```
dti=pd.date_range(start='2018-01-01',end='2018-12-31',freq='D')
df=pd.DataFrame(np.random.randint(0,100,size=dti.shape),index=dti)
```

在…里 [9]:

```
df[df.index.weekday==6].sum()
```

Out[9]:

```
0    2797
dtype: int64
```

在…里 [10]:

```
df = pd.DataFrame({'A': ['one', 'one', 'two', 'three'] * 3,
                   'B': ['A', 'B', 'C'] * 4,
                   'C': ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'] * 2,
                   'D': np.random.randn(12),
                   'E': np.random.randn(12)})

print(df)
```

```
        A  B    C         D         E
0     one  A  foo  1.111406  1.726900
1     one  B  foo -0.687180 -0.805918
2     two  C  foo -0.779389 -0.292853
3   three  A  bar  1.035632  1.334932
4     one  B  bar -0.837283  0.629399
5     one  C  bar -0.378164  0.097210
6     two  A  foo -0.323725  0.264675
7   three  B  foo  1.142563 -2.026599
8     one  C  foo -0.968754  0.076967
9     one  A  bar -0.294419  1.723638
10    two  B  bar  0.905325 -0.718391
11  three  C  bar  0.412672  0.972323
```

在…里 [1]:

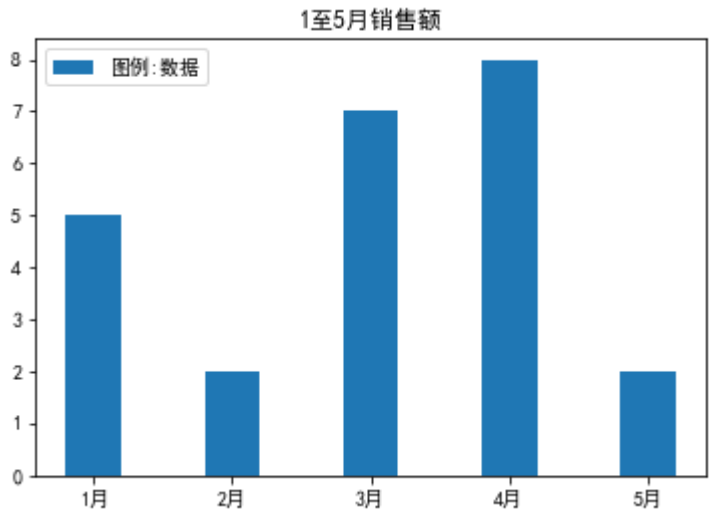
```
from sklearn import tree#导入树
from sklearn.datasets import load_iris#导入数据集
load_iris()
```

Out[1]:

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
```

在…里 [4]:

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

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
fig=plt.figure()
axes=fig.add_subplot(111)
axes.bar([1, 2,3,4,5], [5, 2, 7, 8, 2],width=0.4,tick_label=['1月','2月','3月','4月','5月'], label='

axes.set_title("1至5月销售额")
axes.legend()
plt.show()
```



在…里 [5]:

```
import pandas as pd
from pandas import DataFrame

df: DataFrame = pd.read_excel("D:/zhijiao/课程标准上传情况.xlsx", sheet_name="sheet1")
df=df.sort_values(by='课程名称')
df.index = range(1,df.shape[0]+1)
df.to_excel("处理后.xlsx",sheet_name="sheet2")
```

在…里 [ ]:

```

```

在…里 [ ]:

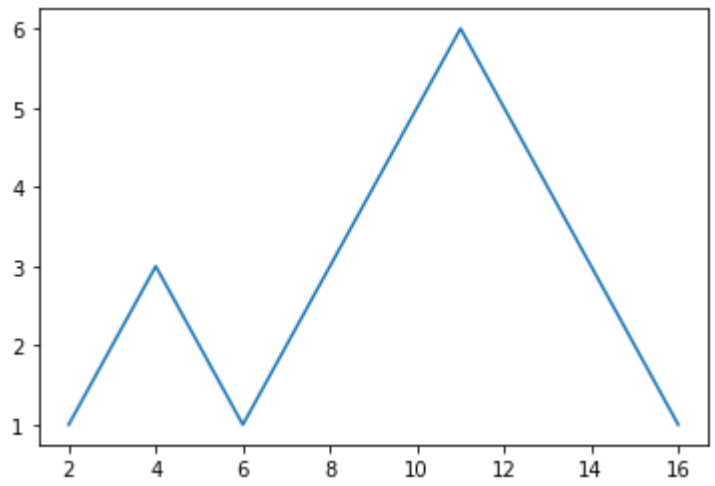
```

```

在…里 [15]:

```
%matplotlib inline
from matplotlib import pyplot as plt

plt.plot([2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16],#自定义X值
         [1, 2, 3, 2, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1])#自定义Y值
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x1c174ff5700>]
```



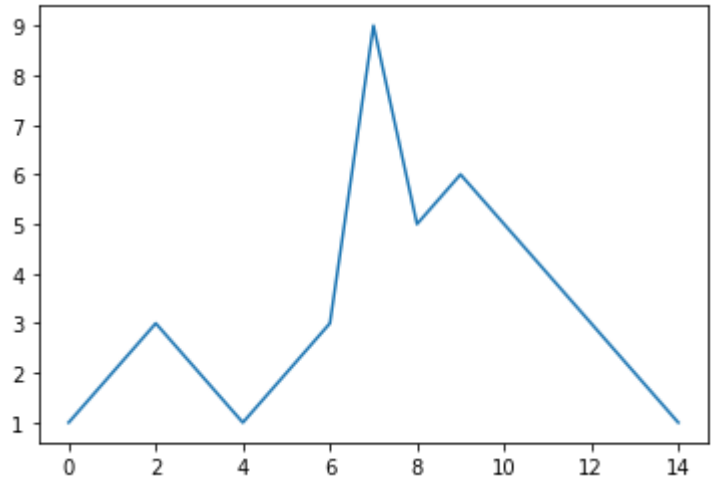
#折线图括号里Y值↓

在…里 [14]:

```
plt.plot([1, 2, 3, 2, 1, 2, 3, 9, 5, 6, 5, 4, 3, 2, 1])#自定义Y值。
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x1c174f880d0>]
```

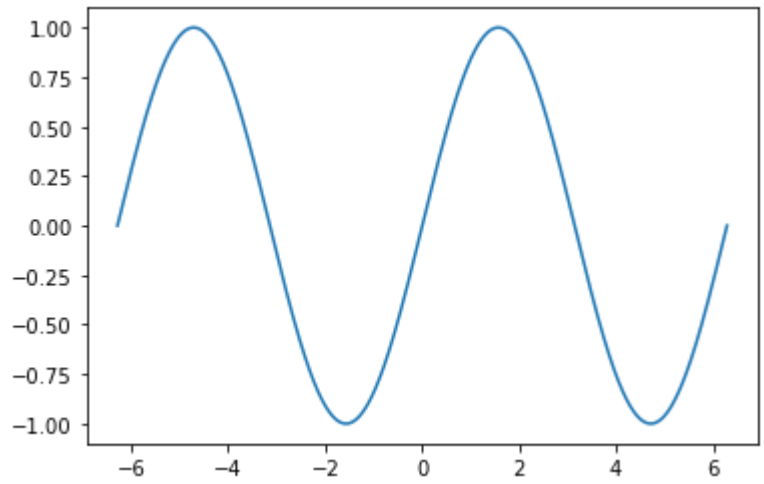


在…里 [19]:

```
import numpy as np  # 载入数值计算模块

# 在 -2PI 和 2PI 之间等间距生成 1000 个值，也就是 X 坐标
X = np.linspace(-2*np.pi, 2*np.pi, 1000)
# 计算 y 坐标
y = np.sin(X)

# 向方法中 `*args` 输入 X, y 坐标
plt.plot(X, y)
#plt.bar([1, 2, 3], [1, 2, 3])
```

Out[19]:

```
[<matplotlib.lines.Line2D at 0x1c17616feb0>]
```

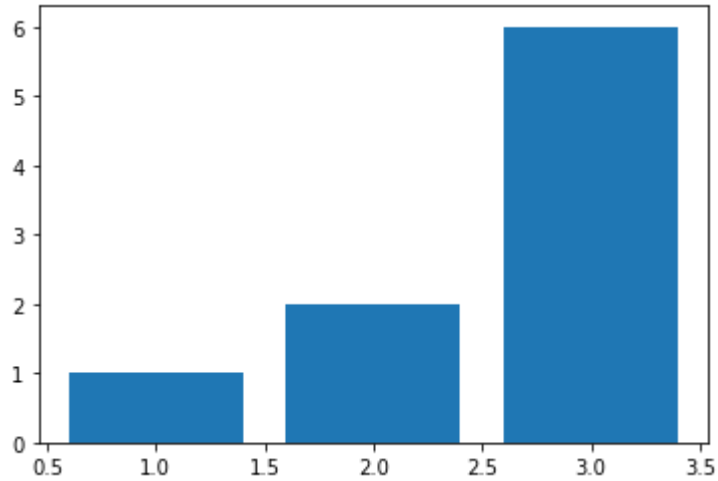


在…里 [21]:

```
plt.bar([1, 2, 3], [1, 2, 6])
```

Out[21]:

```
<BarContainer object of 3 artists>
```

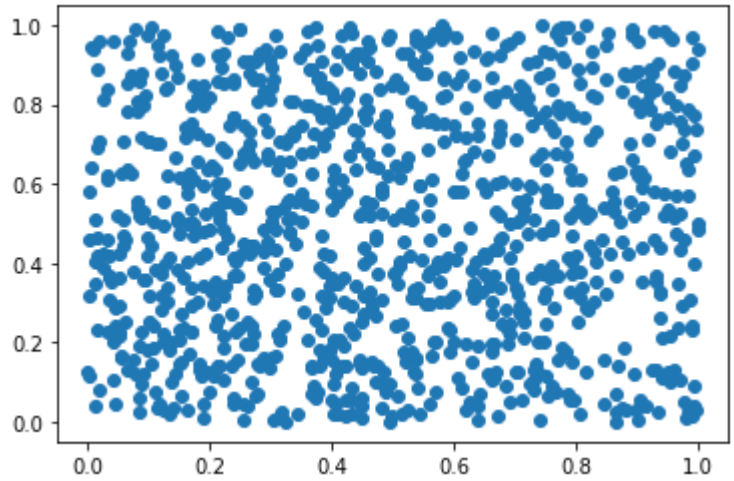


在…里 [22]:

```
# X,y 的坐标均有 numpy 在 0 到 1 中随机生成 1000 个值
X = np.random.ranf(1000)
y = np.random.ranf(1000)
# 向方法中 `*args` 输入 X, y 坐标
plt.scatter(X, y)
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x1c1762c4400>
```



在…里 [27]:

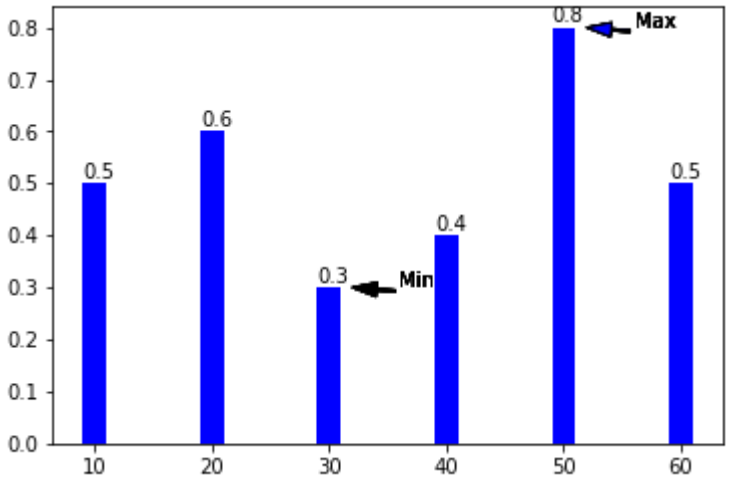
```
plt.pie([1, 1, 1, 1, 1, 1, 1])
```

Out[27]:

```
([<matplotlib.patches.Wedge at 0x1c17648a310>,
  <matplotlib.patches.Wedge at 0x1c17648a850>,
  <matplotlib.patches.Wedge at 0x1c17648ad30>,
  <matplotlib.patches.Wedge at 0x1c17649a250>,
  <matplotlib.patches.Wedge at 0x1c17649a760>,
  <matplotlib.patches.Wedge at 0x1c17649ac40>,
  <matplotlib.patches.Wedge at 0x1c1764a5160>],
 [Text(0.9910657451172095, 0.47727213291294374, ''),
  Text(0.24477296280441296, 1.0724207181325571, ''),
  Text(-0.68583886831644, 0.8600145619153347, ''),
  Text(-1.0999999999999892, -1.5448414893833034e-07, ''),
  Text(-0.6858386267552901, -0.8600147545539077, ''),
  Text(0.24477326402622548, -1.0724206493806185, ''),
  Text(0.9910658791734962, -0.47727185454211024, '')])
```

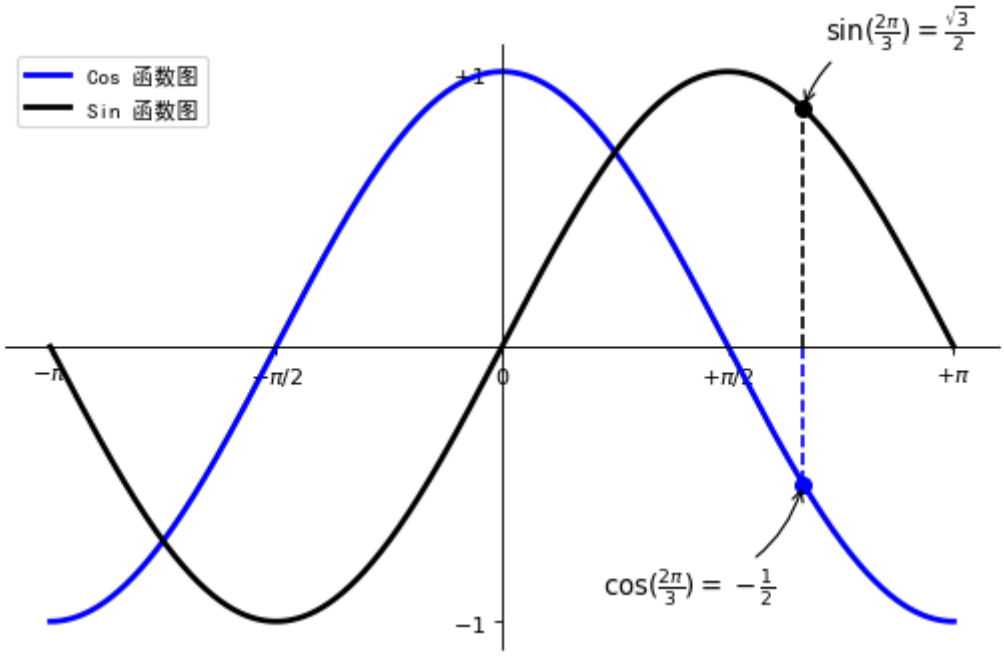
在…里 [39]:

```
fig, axes = plt.subplots()


x_bar = [10, 20, 30, 40, 50 ,60]  # 柱形图横坐标
y_bar = [0.5, 0.6, 0.3, 0.4, 0.8 ,0.5]  # 柱形图纵坐

bars = axes.bar(x_bar, y_bar, color='blue', label=x_bar, width=2)  # 绘制柱形图
for i, rect in enumerate(bars):
    x_text = rect.get_x()  # 获取柱形图横坐标
    y_text = rect.get_height() + 0.01  # 获取柱子的高度并增加 0.01
    plt.text(x_text, y_text, '%.1f' % y_bar[i])  # 标注文字

    # 增加箭头标注
    plt.annotate('Min', xy=(32, 0.3), xytext=(36, 0.3),
                 arrowprops=dict(facecolor='black', width=1, headwidth=7))

    plt.annotate('Max', xy=(52, 0.8), xytext=(56, 0.8),
                 arrowprops=dict(facecolor='blue', width=1, headwidth=7))
```



在…里 [70]:

```
from pylab import *
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False

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, 1000, endpoint=True)
C, S = np.cos(X), np.sin(X)

plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="Cos 函数图")
plt.plot(X, S, color="black", linewidth=2.5, linestyle="-", label="Sin 函数图")

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'$\sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
             xy=(t, np.sin(t)), xycoords='data',
             xytext=(+10, +30), textcoords='offset points', fontsize=12,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.3"))

plt.plot([t, t], [0, np.sin(t)],
         color='black', linewidth=1.5, linestyle="--")
plt.scatter([t, ], [np.sin(t), ], 50, color='black')
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=12,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.3"))

plt.legend(loc='upper left', frameon=True)

plt.show()
```



在…里 [2]:

```
import numpy as np
import pandas as pd
import datetime
```

在…里 [4]:

```
df_ferrara = pd.read_csv(r'D:\zhijiao\zhuang\ferrara_270615.csv')
df_milano = pd.read_csv(r'D:\zhijiao\zhuang\milano_270615.csv')
df_mantova = pd.read_csv(r'D:\zhijiao\zhuang\mantova_270615.csv')
df_ravenna = pd.read_csv(r'D:\zhijiao\zhuang\ravenna_270615.csv')
df_torino = pd.read_csv(r'D:\zhijiao\zhuang\torino_270615.csv')
df_asti = pd.read_csv(r'D:\zhijiao\zhuang\asti_270615.csv')
df_bologna = pd.read_csv(r'D:\zhijiao\zhuang\bologna_270615.csv')
df_piacenza = pd.read_csv(r'D:\zhijiao\zhuang\piacenza_270615.csv')
df_cesena = pd.read_csv(r'D:\zhijiao\zhuang\cesena_270615.csv')
df_faenza = pd.read_csv(r'D:\zhijiao\zhuang\faenza_270615.csv')
```

在…里 [5]:

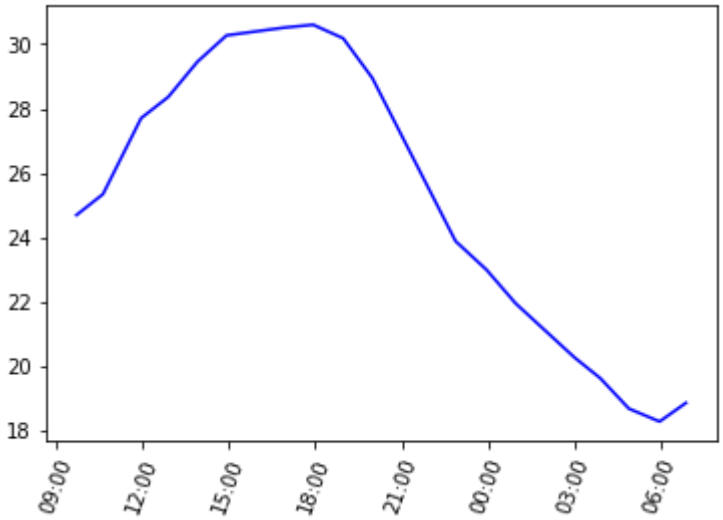
```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from dateutil import parser
```

在…里 [6]:

```
y1 = df_milano['temp']
x1 = df_milano['day']
day_milano = [parser.parse(x) for x in x1]
fig, ax = plt.subplots()
plt.xticks(rotation=70)
hours = mdates.DateFormatter('%H:%M')
ax.xaxis.set_major_formatter(hours)
ax.plot(day_milano ,y1, 'b')
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x277387b8730>]
```

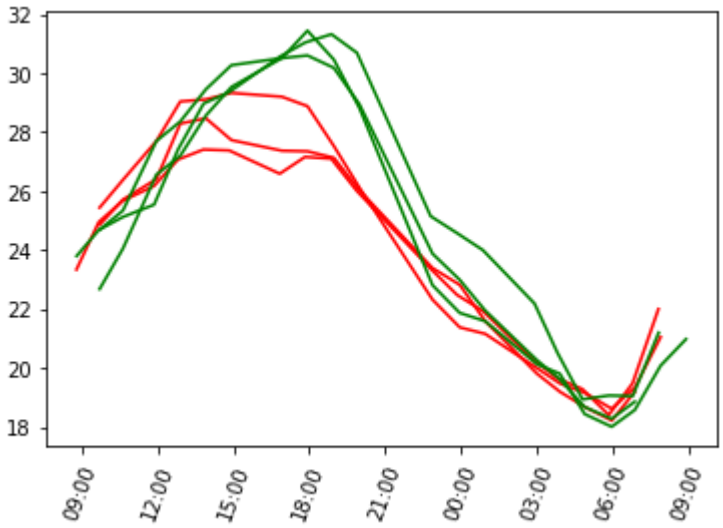


在…里 [7]:

```
y1 = df_ravenna['temp']
x1 = df_ravenna['day']
y2 = df_faenza['temp']
x2 = df_faenza['day']
y3 = df_cesena['temp']
x3 = df_cesena['day']
y4 = df_milano['temp']
x4 = df_milano['day']
y5 = df_asti['temp']
x5 = df_asti['day']
y6 = df_torino['temp']
x6 = df_torino['day']
day_ravenna = [parser.parse(x) for x in x1]
day_faenza = [parser.parse(x) for x in x2]
day_cesena = [parser.parse(x) for x in x3]
day_milano = [parser.parse(x) for x in x4]
day_asti = [parser.parse(x) for x in x5]
day_torino = [parser.parse(x) for x in x6]
fig, ax = plt.subplots()
plt.xticks(rotation=70)
hours = mdates.DateFormatter('%H:%M')
ax.xaxis.set_major_formatter(hours)
ax.plot(day_ravenna,y1,'r',day_faenza,y2,'r',day_cesena,y3,'r')
ax.plot(day_milano,y4,'g',day_asti,y5,'g',day_torino,y6,'g')
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x277388d59d0>,
 <matplotlib.lines.Line2D at 0x277388d5820>,
 <matplotlib.lines.Line2D at 0x277388d5a00>]
```

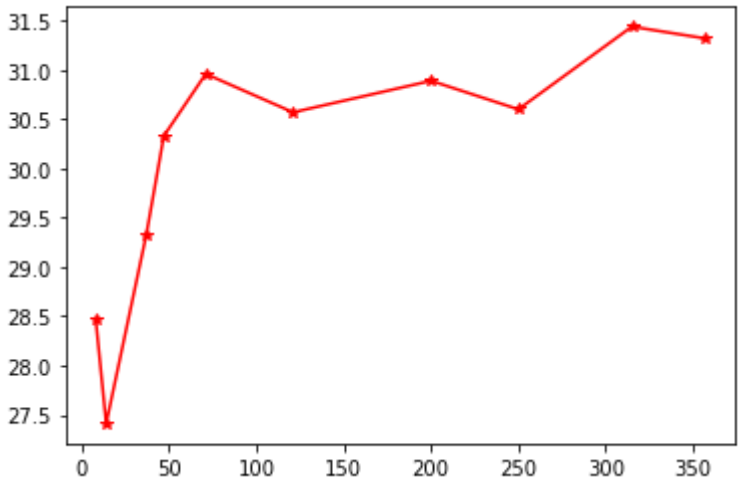


在…里 [8]:

```
dist = [df_ravenna['dist'][0],
    df_cesena['dist'][0],
    df_faenza['dist'][0],
    df_ferrara['dist'][0],
    df_bologna['dist'][0],
    df_mantova['dist'][0],
    df_piacenza['dist'][0],
    df_milano['dist'][0],
    df_asti['dist'][0],
    df_torino['dist'][0]
]

temp_max = [df_ravenna['temp'].max(),
    df_cesena['temp'].max(),
    df_faenza['temp'].max(),
    df_ferrara['temp'].max(),
    df_bologna['temp'].max(),
    df_mantova['temp'].max(),
    df_piacenza['temp'].max(),
    df_milano['temp'].max(),
    df_asti['temp'].max(),
    df_torino['temp'].max()
]
temp_min = [df_ravenna['temp'].min(),
    df_cesena['temp'].min(),
    df_faenza['temp'].min(),
    df_ferrara['temp'].min(),
    df_bologna['temp'].min(),
    df_mantova['temp'].min(),
    df_piacenza['temp'].min(),
    df_milano['temp'].min(),
    df_asti['temp'].min(),
    df_torino['temp'].min()
]
fig, ax = plt.subplots()
ax.plot(dist,temp_max,'r*-')
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x27738955790>]
```

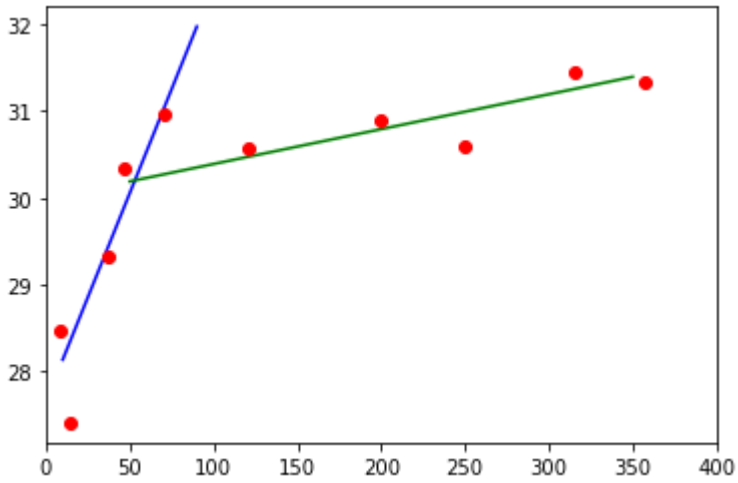


在…里 [9]:

```
from sklearn.svm import SVR
import numpy as np
# dist1是靠近海的城市集合，dist2是远离海洋的城市集合
dist1 = dist[0:5]
dist2 = dist[5:10]

# 改变列表的结构，dist1现在是5个列表的集合
# 之后我们会看到 nbumpy 中 reshape() 函数也有同样的作用
dist1 = [[x] for x in dist1]
dist2 = [[x] for x in dist2]

# temp_max1 是 dist1 中城市的对应最高温度
temp_max1 = temp_max[0:5]
# temp_max2 是 dist2 中城市的对应最高温度
temp_max2 = temp_max[5:10]

# 我们调用SVR函数，在参数中规定了使用线性的拟合函数
# 并且把 C 设为1000来尽量拟合数据（因为不需要精确预测不用担心过拟合）
svr_lin1 = SVR(kernel='linear', C=1e3)
svr_lin2 = SVR(kernel='linear', C=1e3)

# 加入数据，进行拟合（这一步可能会跑很久，大概10多分钟，休息一下:）)
svr_lin1.fit(dist1, temp_max1)
svr_lin2.fit(dist2, temp_max2)

#xp1 = np.arange(10,100,10).reshape((9,1))
#xp2 = np.arange(50,400,50).reshape((7,1))
xp1 = np.array[1,2,3,4,5]
xp2 = np.array[]

yp1 = svr_lin1.predict(xp1)
yp2 = svr_lin2.predict(xp2)



fig, ax = plt.subplots()
ax.set_xlim(0,400)


ax.plot(xp1, yp1, c='b', label='Strong sea effect')
ax.plot(xp2, yp2, c='g', label='Light sea effect')
ax.plot(dist,temp_max,'ro')
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x2773ad24610>]
```



在…里 [10]:

```
print(svr_lin1.coef_)  #斜率
print(svr_lin1.intercept_)  # 截距
print(svr_lin2.coef_)
print(svr_lin2.intercept_)
```

```
[[0.04794118]]
[27.65617647]
[[0.00401274]]
[29.98745222]
```

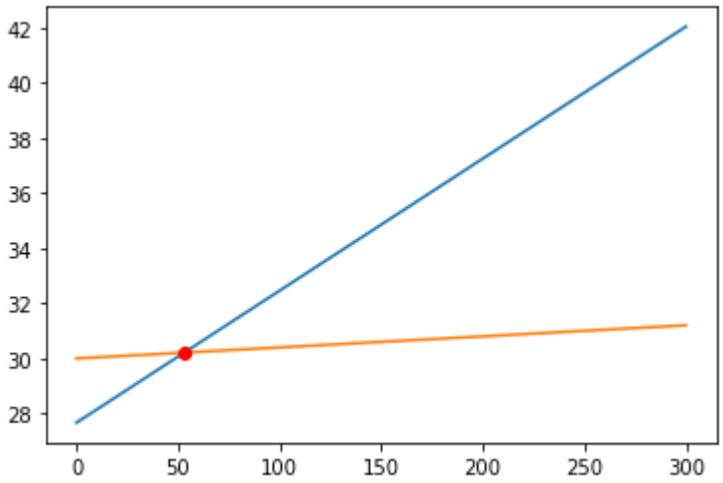
在…里 [11]:

```
from scipy.optimize import fsolve

# 定义了第一条拟合直线
def line1(x):
    a1 = svr_lin1.coef_[0][0]
    b1 = svr_lin1.intercept_[0]
    return a1*x + b1

# 定义了第二条拟合直线
def line2(x):
    a2 = svr_lin2.coef_[0][0]
    b2 = svr_lin2.intercept_[0]
    return a2*x + b2

# 定义了找到两条直线的交点的 x 坐标的函数
def findIntersection(fun1,fun2,x0):
    return fsolve(lambda x : fun1(x) - fun2(x),x0)

result = findIntersection(line1,line2,0.0)
print("[x,y] = [ %d , %d ]" % (result,line1(result)))

# x = [0,10,20, ..., 300]
x = np.linspace(0,300,31)
plt.plot(x,line1(x),x,line2(x),result,line1(result),'ro')
```

```
[x,y] = [ 53 , 30 ]
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x2773ad92a30>,
 <matplotlib.lines.Line2D at 0x2773ad92a00>,
 <matplotlib.lines.Line2D at 0x2773ad92a60>]
```

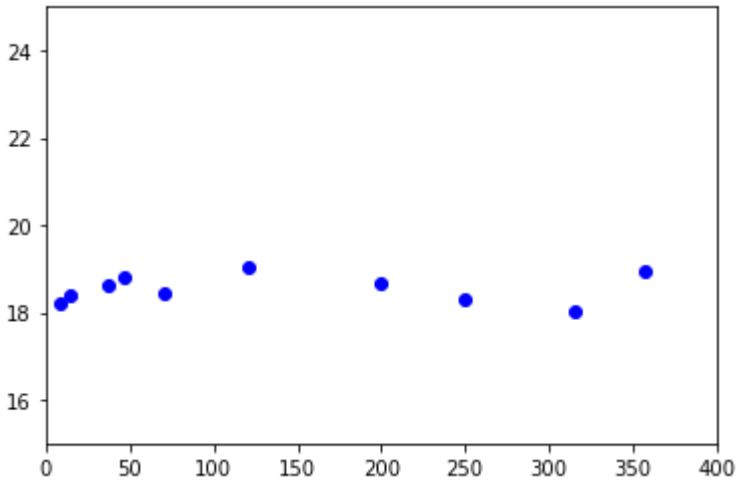


在…里 [12]:

```
# axis 函数规定了 x 轴和 y 轴的取值范围
plt.axis((0,400,15,25))
plt.plot(dist,temp_min,'bo')
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x2773ae015e0>]
```

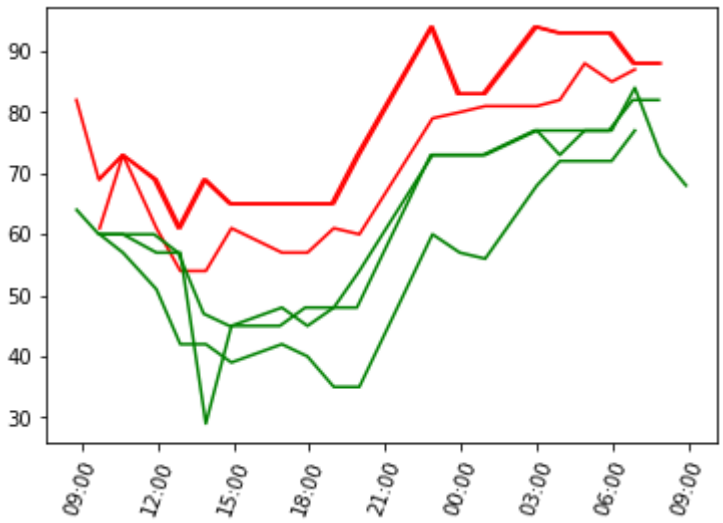


在…里 [13]:

```
# 读取湿度数据
y1 = df_ravenna['humidity']
x1 = df_ravenna['day']
y2 = df_faenza['humidity']
x2 = df_faenza['day']
y3 = df_cesena['humidity']
x3 = df_cesena['day']
y4 = df_milano['humidity']
x4 = df_milano['day']
y5 = df_asti['humidity']
x5 = df_asti['day']
y6 = df_torino['humidity']
x6 = df_torino['day']

# 重新定义 fig 和 ax 变量
fig, ax = plt.subplots()
plt.xticks(rotation=70)

# 把时间从 string 类型转化为标准的 datetime 类型
day_ravenna = [parser.parse(x) for x in x1]
day_faenza = [parser.parse(x) for x in x2]
day_cesena = [parser.parse(x) for x in x3]
day_milano = [parser.parse(x) for x in x4]
day_asti = [parser.parse(x) for x in x5]
day_torino = [parser.parse(x) for x in x6]

# 规定时间的表示方式
hours = mdates.DateFormatter('%H:%M')
ax.xaxis.set_major_formatter(hours)

#表示在图上
ax.plot(day_ravenna,y1,'r',day_faenza,y2,'r',day_cesena,y3,'r')
ax.plot(day_milano,y4,'g',day_asti,y5,'g',day_torino,y6,'g')
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x2773ae86250>,
 <matplotlib.lines.Line2D at 0x2773ae81d30>,
 <matplotlib.lines.Line2D at 0x2773ae81d90>]
```

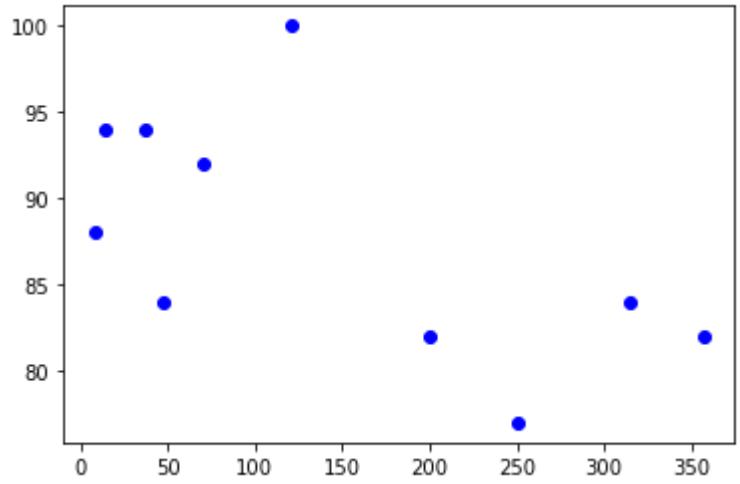


在…里 [14]:

```
# 获取最大湿度数据
hum_max = [df_ravenna['humidity'].max(),
df_cesena['humidity'].max(),
df_faenza['humidity'].max(),
df_ferrara['humidity'].max(),
df_bologna['humidity'].max(),
df_mantova['humidity'].max(),
df_piacenza['humidity'].max(),
df_milano['humidity'].max(),
df_asti['humidity'].max(),
df_torino['humidity'].max()
]

plt.plot(dist,hum_max,'bo')
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x2773aef80a0>]
```

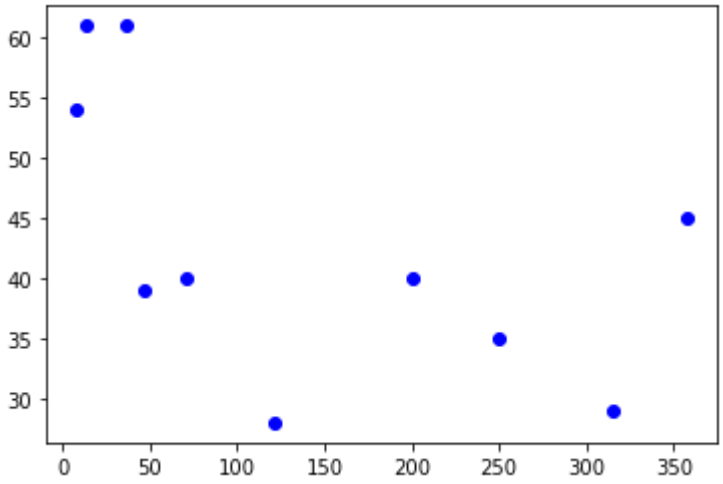


在…里 [15]:

```
# 获取最小湿度
hum_min = [
df_ravenna['humidity'].min(),
df_cesena['humidity'].min(),
df_faenza['humidity'].min(),
df_ferrara['humidity'].min(),
df_bologna['humidity'].min(),
df_mantova['humidity'].min(),
df_piacenza['humidity'].min(),
df_milano['humidity'].min(),
df_asti['humidity'].min(),
df_torino['humidity'].min()
]
plt.plot(dist,hum_min,'bo')
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x2773af55c40>]
```

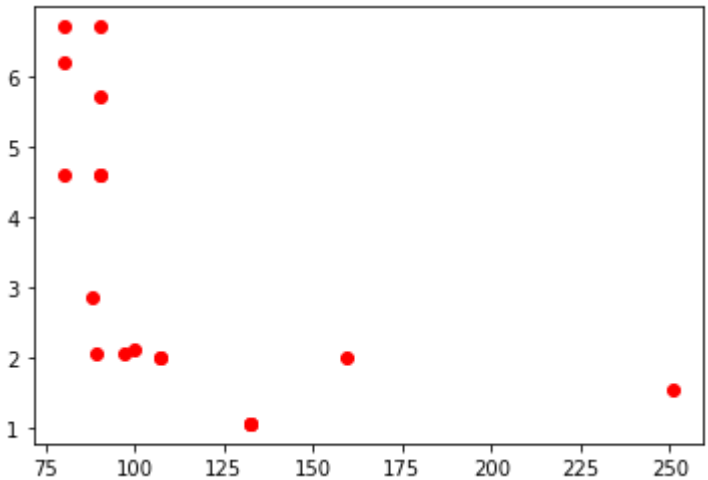


在…里 [16]:

```
plt.plot(df_ravenna['wind_deg'],df_ravenna['wind_speed'],'ro')
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x2773afc0ee0>]
```



在…里 [16]:

```
hist, bins = np.histogram(df_ravenna['wind_deg'],8,[0,360])
print(hist)
print(bins)
```

```
[ 0  5 11  1  0  1  0  0]
[  0.  45.  90. 135. 180. 225. 270. 315. 360.]
```

在…里 [18]:

```
def showRoseWind(values,city_name,max_value):
    N = 8

    # theta = [pi*1/4, pi*2/4, pi*3/4, ..., pi*2]
    theta = np.arange(2 * np.pi / 16, 2 * np.pi, 2 * np.pi / 8)
    radii = np.array(values)
    # 绘制极区图的坐标系
    plt.axes([0.025, 0.025, 0.95, 0.95], polar=True)

    # 列表中包含的是每一个扇区的 rgb 值，x越大，对应的color越接近蓝色
    colors = [(1-x/max_value, 1-x/max_value, 0.75) for x in radii]

    # 画出每个扇区
    plt.bar(theta, radii, width=(2*np.pi/N), bottom=0.0, color=colors)

    # 设置极区图的标题
    plt.title(city_name, x=0.2, fontsize=20)
    showRoseWind(hist,'Ravenna',max(hist))
```

在…里 [3]:

```
hist, bin = np.histogram(df_ferrara['wind_deg'],8,[0,360])
print(hist)
showRoseWind(hist,'Ferrara', max(hist))
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call last)
C:\Windows\TEMP/ipykernel_15016/1473403885.py in <module>
----> 1 hist, bin = np.histogram(df_ferrara['wind_deg'],8,[0,360])
      2 print(hist)
      3 showRoseWind(hist,'Ferrara', max(hist))

NameError: name 'df_ferrara' is not defined
```

在…里 [ ]:

```
def RoseWind_Speed(df_city):
    # degs = [45, 90, ..., 360]
    degs = np.arange(45,361,45)
    tmp = []
    for deg in degs:
        # 获取 wind_deg 在指定范围的风速平均值数据
        tmp.append(df_city[(df_city['wind_deg']>(deg-46)) & (df_city['wind_deg']<deg)]
        ['wind_speed'].mean())
    return np.array(tmp)
```

在…里 [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['ases.unicode_minus'] = False
plt.title('披萨价格与直径数据')
plt.xlabel('直径（英寸）')
plt.ylabel('价格（美元）')
plt.axis([0,25,0,25])
x=np.array([[6],[8],[10],[14],[18]])
y=np.array([[7],[9],[13],[17.5],[18]])
X_ba=X.mean()
y_ba=y.mean()
b_hat=np.sum(((X-X_ba)*(y-y_ba)))/np.sum((X-X_ba)**2)
a_hat=y_ba-b_hat*X_ba
plt.p
```

在…里 [1]:

```
import numpy as np

def createDataSet():
    group = np.array([[1.0, 1.1], [1.0, 1.0], [0, 0], [0, 0.1]])
    labels = ['A', 'A', 'B', 'B']
    return group, labels
group, labels = createDataSet()

print('group:', group)
print('labels:', labels)  # 输出数值
```

```
group: [[1.  1.1]
 [1.  1. ]
 [0.  0. ]
 [0.  0.1]]
labels: ['A', 'A', 'B', 'B']
```

在…里 [2]:

```
!wget -nc "http://labfile.oss.aliyuncs.com/courses/777/digits.zip"

# 在 Jupyter Notebook 单元格中执行，下载并解压数据。
!wget -nc "http://labfile.oss.aliyuncs.com/courses/777/digits.zip"
# 解压缩
!unzip -o digits.zip
```

```
File 'digits.zip' already there; not retrieving.

File 'digits.zip' already there; not retrieving.
```

在…里 [5]:

```
!type C:\Users\吴秋雨\digits\testDigits\0_1.txt #windows用type不用cat
```

```
00000000000000011000000000000000
00000000000111111110000000000000
00000000001111111111100000000000
00000000001111111111110000000000
00000000011111111111111000000000
00000000011111100011111000000000
00000000111111000000111100000000
00000000111111000000111110000000
00000000111111000000111110000000
00000000111111000000011111000000
00000001111110000000011111000000
00000001111110000000001111000000
00000001111110000000001111100000
00000001111100000000001111000000
00000001111000000000001111000000
00000001111000000000001111000000
00000001111000000000001111000000
00000001111000000000001111000000
00000001111000000000001111000000
00000001111000000000001111000000
00000001111000000000001111000000
00000001111000000000011110000000
```

C:\Users\吴秋雨\digits\testDigits\0_1.txt

系统找不到指定的文件。
处理: #windows用type不用cat 时出错。

```
00000001111000000000011110000000
00000001110000000000011110000000
00000001111110000011111110000000
00000001111110001111111100000000
00000001111111111111111000000000
00000000111111111111111000000000
00000001111111111111100000000000
00000000111111111111000000000000
00000000011111111000000000000000
00000000011111100000000000000000
00000000001000000000000000000000
```

在…里 [6]:

```
def img2vector(filename):
    # 创建向量
    returnVect = np.zeros((1, 1024))
    # 打开数据文件，读取每行内容
    fr = open(filename)
    for i in range(32):
        # 读取每一行
        lineStr = fr.readline()
        # 将每行前 32 字符转成 int 存入向量
        for j in range(32):
            returnVect[0, 32*i+j] = int(lineStr[j])

    return returnVect
```

在…里 [8]:

```
import numpy as np
img2vector('digits/testDigits/0_1.txt')
```

Out[8]:

```
array([[0., 0., 0., ..., 0., 0., 0.]])
```

在…里 [11]:

```
import operator


def classify0(inX, dataSet, labels, k):

    """
    参数:
    - inX: 用于分类的输入向量
    - dataSet: 输入的训练样本集
    - labels: 样本数据的类标签向量
    - k: 用于选择最近邻居的数目
    """

    # 获取样本数据数量
    dataSetSize = dataSet.shape[0]

    # 矩阵运算，计算测试数据与每个样本数据对应数据项的差值
    diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet

    # sqDistances 上一步骤结果平方和
    sqDiffMat = diffMat**2
    sqDistances = sqDiffMat.sum(axis=1)

    # 取平方根，得到距离向量
    distances = sqDistances**0.5

    # 按照距离从低到高排序
    sortedDistIndicies = distances.argsort()
    classCount = {}

    # 依次取出最近的样本数据
    for i in range(k):
        # 记录该样本数据所属的类别
        voteIlabel = labels[sortedDistIndicies[i]]
        classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1

    # 对类别出现的频次进行排序，从高到低
    sortedClassCount = sorted(
        classCount.items(), key=operator.itemgetter(1), reverse=True)

    # 返回出现频次最高的类别
    return sortedClassCount[0][0]
```

在…里 [12]:

```
group, labels = createDataSet()
classify0([0, 0], group, labels, 3)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call last)
C:\Windows\TEMP/ipykernel_908/1552587387.py in <module>
----> 1 group, labels = createDataSet()
      2 classify0([0, 0], group, labels, 3)

NameError: name 'createDataSet' is not defined
```

在…里 [13]:

```
from os import listdir


def handwritingClassTest():
    # 样本数据的类标签列表
    hwLabels = []

    # 样本数据文件列表
    trainingFileList = listdir('digits/trainingDigits')
    m = len(trainingFileList)

    # 初始化样本数据矩阵（M*1024）
    trainingMat = np.zeros((m, 1024))

    # 依次读取所有样本数据到数据矩阵
    for i in range(m):
        # 提取文件名中的数字
        fileNameStr = trainingFileList[i]
        fileStr = fileNameStr.split('.')[0]
        classNumStr = int(fileStr.split('_')[0])
        hwLabels.append(classNumStr)

        # 将样本数据存入矩阵
        trainingMat[i, :] = img2vector(
            'digits/trainingDigits/%s' % fileNameStr)

    # 循环读取测试数据
    testFileList = listdir('digits/testDigits')

    # 初始化错误率
    errorCount = 0.0
    mTest = len(testFileList)

    # 循环测试每个测试数据文件
    for i in range(mTest):

        fileNameStr = testFileList[i]
        fileStr = fileNameStr.split('.')[0]
        classNumStr = int(fileStr.split('_')[0])

        # 提取数据向量
        vectorUnderTest = img2vector('digits/testDigits/%s' % fileNameStr)

        # 对数据文件进行分类
        classifierResult = classify0(vectorUnderTest, trainingMat, hwLabels, 3)

        # 打印 K 近邻算法分类结果和真实的分类
        print("测试样本 %d, 分类器预测: %d, 真实类别: %d" %
              (i+1, classifierResult, classNumStr))

        # 判断K 近邻算法结果是否准确
        if (classifierResult != classNumStr):
            errorCount += 1.0

    # 打印错误率
    print("\n错误分类计数: %d" % errorCount)
    print("\n错误分类比例: %f" % (errorCount/float(mTest)))
```

在…里 [14]:

```
handwritingClassTest()
```

```
测试样本 1, 分类器预测: 0, 真实类别: 0
测试样本 2, 分类器预测: 0, 真实类别: 0
测试样本 3, 分类器预测: 0, 真实类别: 0
测试样本 4, 分类器预测: 0, 真实类别: 0
测试样本 5, 分类器预测: 0, 真实类别: 0
测试样本 6, 分类器预测: 0, 真实类别: 0
测试样本 7, 分类器预测: 0, 真实类别: 0
测试样本 8, 分类器预测: 0, 真实类别: 0
测试样本 9, 分类器预测: 0, 真实类别: 0
测试样本 10, 分类器预测: 0, 真实类别: 0
测试样本 11, 分类器预测: 0, 真实类别: 0
测试样本 12, 分类器预测: 0, 真实类别: 0
测试样本 13, 分类器预测: 0, 真实类别: 0
测试样本 14, 分类器预测: 0, 真实类别: 0
测试样本 15, 分类器预测: 0, 真实类别: 0
测试样本 16, 分类器预测: 0, 真实类别: 0
测试样本 17, 分类器预测: 0, 真实类别: 0
测试样本 18, 分类器预测: 0, 真实类别: 0
测试样本 19, 分类器预测: 0, 真实类别: 0
```

在…里 [ ]:

```

```