Exercise-1

1.1

试下载IBM公司股票每日的开盘价、收盘价、最高价、最低价；
略

1.2

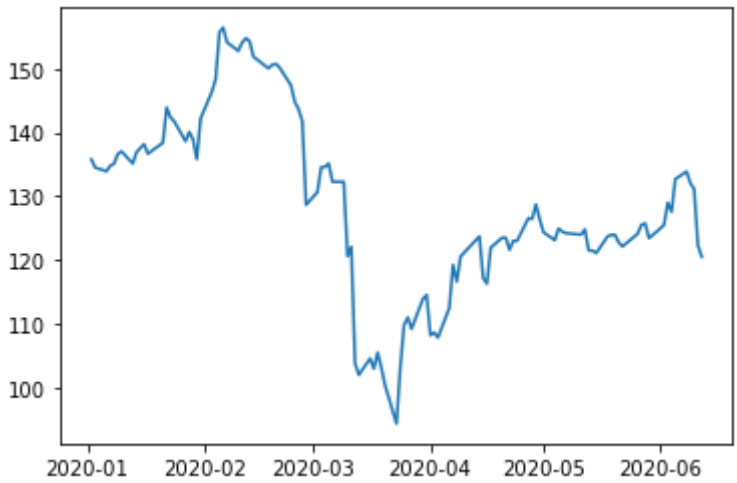
提取其中的收盘价，并画出其时间序列图；

代码：

```
# 导入 pandas，用以读取excel
import pandas as pd
# 导入 matplotlib.pyplot，用以画图
from matplotlib import pyplot as plt
df = pd.read_excel('IBM.xlsx')
x = df[['日期']].values
y = df[['收盘价']].values

plt.plot(x,y)
plt.show()
```

运行：



1.3

计算收盘价的每日简单收益率、对数收益率；

代码：

```
import math

df['简单收益率'] = df.apply(lambda x: (x['收盘价'] / x['开盘价']) - 1, axis=1)
df['对数收益率'] = df.apply(lambda x: math.log((x['收盘价'] / x['开盘价']),math.e),
axis=1)
print(df)
```

运行：

```
            日期       开盘价       收盘价       最高价       最低价      简单收益率
对数收益率
0   2020-01-02  135.200  135.775  135.8200  135.0900  0.004253  0.004244
1   2020-01-03  134.000  134.500  134.6730  134.0000  0.003731  0.003724
2   2020-01-06  133.290  133.925  134.2770  133.2100  0.004764  0.004753
3   2020-01-07  133.420  134.775  134.8150  133.4200  0.010156  0.010105
4   2020-01-08  134.240  135.075  135.2580  134.2400  0.006220  0.006201
..         ...      ...      ...       ...       ...       ...       ...
106 2020-06-08  132.270  133.875  134.9950  132.2700  0.012134  0.012061
107 2020-06-09  133.310  132.025  133.5200  131.2078 -0.009639 -0.009686
108 2020-06-10  131.605  131.150  131.6050  130.3000 -0.003457 -0.003463
109 2020-06-11  126.000  122.300  126.0000  118.6600 -0.029365 -0.029805
110 2020-06-12  121.400  120.500  122.8177  119.9100 -0.007414 -0.007441

[111 rows x 7 columns]
```

1.4

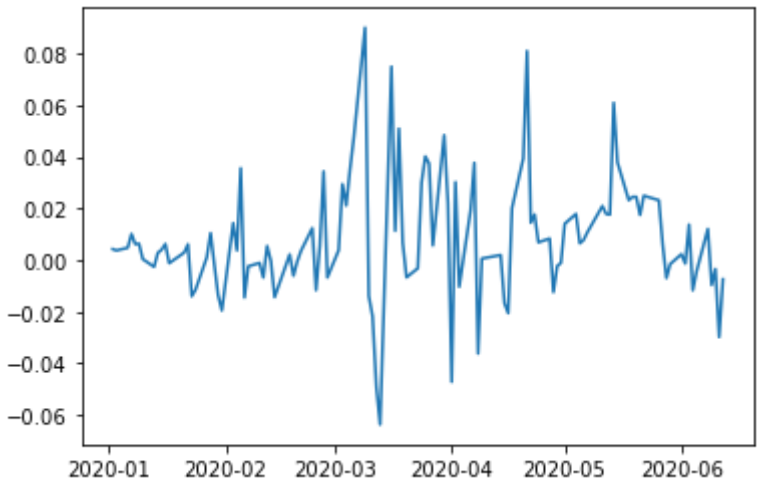
画出收盘价对数收益率的时间序列图。

代码：

```
x = df[['日期']].values
y = df[['对数收益率']].values

plt.plot(x,y)
plt.show()
```

运行：



Exercise-2

考虑从2001年9月1日到2011年9月30日美国运通公司（AXP）、CRSP价值权重指数（VW）、CRSO的等权重指数（EW）以及S&P综合指数的日简单收益率。收益率中包含有支付的股息。数据来自d-axp3dx-0111.txt.

2.1

计算每个收益率序列的样本均值、标准差、偏度、超额峰度、最大值和最小值；
代码：

```
import pandas as pd

df = pd.read_csv('d-axp3dx-0111.txt',delim_whitespace = True)
for k in ['axp','vw','ew','sp']:
    print(k)
    print('均值' + ' ' *3 + str(df[k].mean()))
    print('标准差' + ' ' * 3 + str(df[k].std()))
    # 偏度
    print('偏度' + ' ' * 3 + str(df[k].skew()))
    # 峰度
    print('峰度' + ' ' * 3 + str(df[k].kurt()))
    print('最大值' + ' ' * 3 + str(df[k].max()))
    print('最小值' + ' ' * 3 + str(df[k].min()))
    print('\n')
```

运行：

```
axp
均值   0.000533948717948718
标准差   0.026368406811986583
偏度   0.4603179756054301
峰度   9.623329664672113
最大值   0.206485
最小值   -0.175949


vw
均值   0.00022406153846153844
标准差   0.013651835330332094
偏度   -0.09843483120957383
峰度   8.008957349657706
最大值   0.114889
最小值   -0.089762


ew
均值   0.0006258232741617356
标准差   0.012080369186962037
偏度   -0.2477032329471355
峰度   8.135600589820209
最大值   0.107422
最小值   -0.07824


sp
均值   9.422840236686389e-05
```

```
标准差    0.013779116971407941
偏度    0.008161598940268256
峰度    8.56101370353777
最大值    0.1158
最小值    -0.09035
```

2.2

把简单收益率转换成对数收益率，计算每个对数收益率的样本均值、标准差、偏度、超额峰度、最大值和最小值；
代码：

```
import math

for k in ['axp','vw','ew','sp']:
    df[k + '_log'] = df.apply(lambda x: math.log(x[k] + 1,math.e), axis=1)

for k in ['axp','vw','ew','sp']:
    print(k)
    k = k + '_log'
    print('均值' + ' ' *3 + str(df[k].mean()))
    print('标准差' + ' ' * 3 + str(df[k].std()))
    # 偏度
    print('偏度' + ' ' * 3 + str(df[k].skew()))
    # 峰度
    print('峰度' + ' ' * 3 + str(df[k].kurt()))
    print('最大值' + ' ' * 3 + str(df[k].max()))
    print('最小值' + ' ' * 3 + str(df[k].min()))
    print('\n')
```

运行：

```
axp
均值    0.0001880014136029377
标准差    0.026294387576735707
偏度    0.02101665279290005
峰度    9.050195096827077
最大值    0.18771117334921902
最小值    -0.1935228577840872


vw
均值    0.0001307503587262582
标准差    0.013670358972095054
偏度    -0.3007080567155763
峰度    7.906623231123868
最大值    0.10875484838698898
最小值    -0.09404917520492419


ew
均值    0.0005525758364483264
标准差    0.01209955048903987
偏度    -0.42782121785230354
```

```
峰度    8.044633729746181
最大值    0.10203479156301737
最小值    -0.08147039299806404


sp
均值    -7.486380122537771e-07
标准差    0.013790413376831727
偏度    -0.20660133413839948
峰度    8.350592104368243
最大值    0.10957163642929083
最小值    -0.09469536883932453
```

Exercise-3

几何X 79 75 77 73 78 81 76 72 70
代数Y 80 82 76 77 84 81 72 70 75

3.1

请建立由X估计Y的回归方程；

代码：

```
from sklearn.linear_model import LinearRegression
import numpy as np

x = [79,75,77,73,78,81,76,72,70]
y = [80,82,76,77,84,81,72,70,75]

lr = LinearRegression()
lr.fit(np.array(x).reshape(-1,1),y)
print(lr.coef_)
print(lr.intercept_)
print('一元线性回归方程为: '+'y' + '=' + str(lr.intercept_)+' + ' +str(lr.coef_) +
'*x')
```

运行：

```
[0.78333333]
18.172222222222224
一元线性回归方程为: y=18.172222222222224 + [0.78333333]*x
```

3.2

检验回归方程的显著性；
代码：

```
import statsmodels.api as sm
print(sm.OLS(y,x).fit().pvalues)
```

运行：

```
[6.73496314e-12]
```

3.3

几何得分为71分的学生，其代数分数为几何？
代码：

```
x_predict = [71]
print(lr.predict(np.array(x_predict).reshape(-1,1)))
```

运行：

```
[73.78888889]
```

Exercise-4

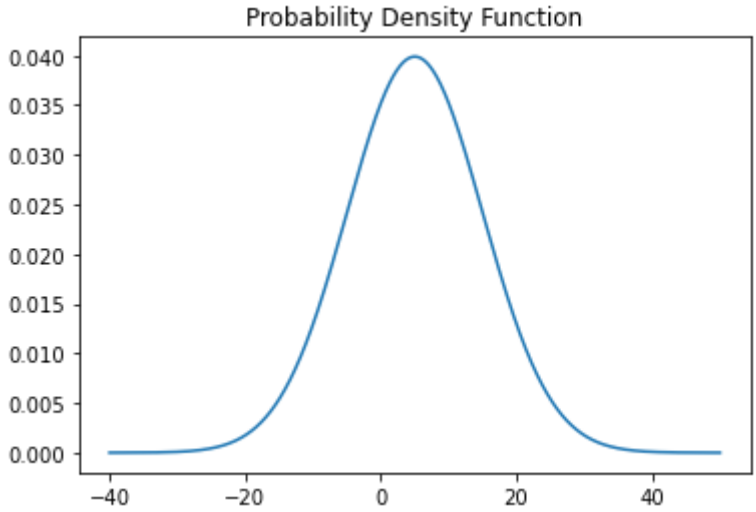
4.1

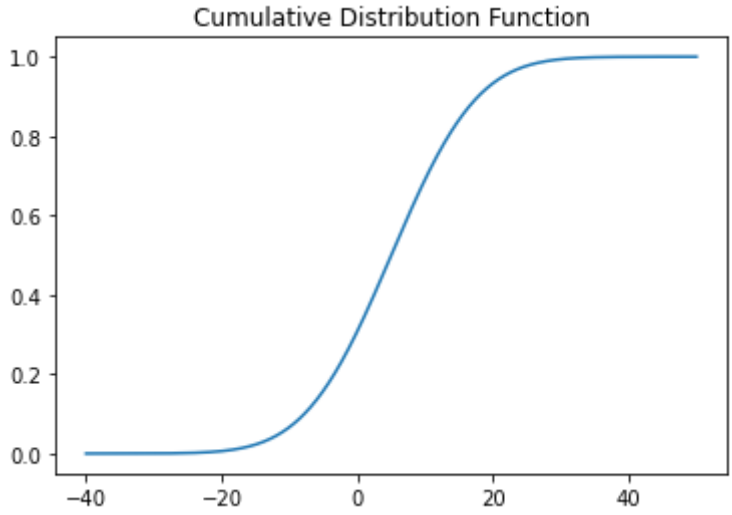
试画出正态分布N(5,100)的密度函数图和分布函数图；
代码：

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

x = np.linspace(-40,50,100000)
# 密度函数
y = stats.norm.pdf(x,5,10)
# 分布函数
z = stats.norm.cdf(x,5,10)
plt.plot(x,y)
plt.title('Probability Density Function')
plt.show()
plt.plot(x,z)
plt.title('Cumulative Distribution Function')
plt.show()
```

运行：



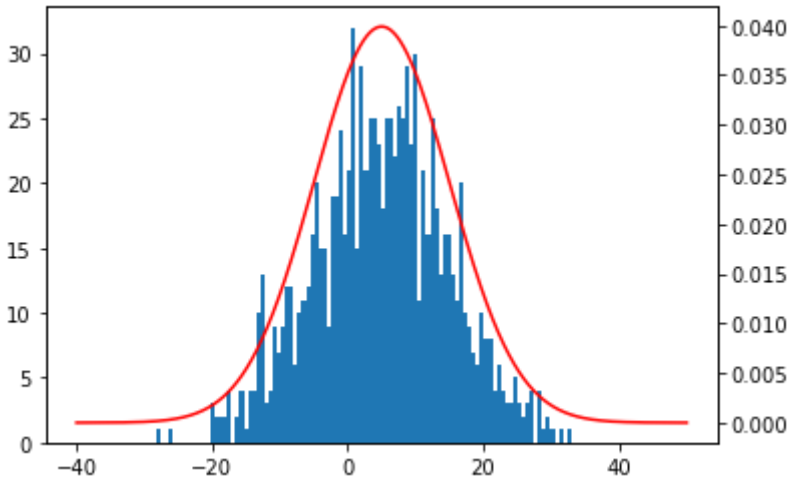


4.2

试求出服从正态分布N(5,100)的随机数1000个，并画出直方图，上面套上正态分布拟合图和非参数核函数拟合图（用不同颜色）；
代码：

```
mu ,sigma = 5, 10
sampleNo = 1000
# np.random.seed(10)
s = np.random.normal(mu, sigma, sampleNo)
fig = plt.figure()
ax1 = fig.add_subplot()
ax1.hist(s,bins=100)
ax2 = ax1.twinx() # this is the important function
ax2.plot(x,y,c='red')
plt.show()
```

运行：



4.3

对随机数进行正态性检验，并给出结论。
代码：

```
from scipy import stats

# 输出结果中第一个为统计量，第二个为P值
# 统计量越接近1越表明数据和正态分布拟合的好
# 如果P值大于显著性水平，通常是0.05，接受原假设，则判断样本的总体服从正态分布
print(stats.shapiro(s))


# 输出结果中第一个为统计量，第二个为P值
# 注：统计量越接近0就越表明数据和标准正态分布拟合的越好，
# 如果P值大于显著性水平，通常是0.05，接受原假设，则判断样本的总体服从正态分布
print(stats.kstest(s, 'norm'))

# 输出结果中第一个为统计量，第二个为P值
# 如果P值大于显著性水平，通常是0.05，接受原假设，则判断样本的总体服从正态分布
print(stats.normaltest(s))
```

运行：

```
(0.9987096190452576, 0.6950380206108093)
KstestResult(statistic=0.6053328770600853, pvalue=0.0)
NormaltestResult(statistic=0.47794529833069066, pvalue=0.7874364191214629)
```

解释：
综上所述，该随机数符合正态分布。