棕榈学院

10 天 Python 训练营讲义

第八讲

2018 年 10 月 22 日-2018 年 10 月 31 日

目录

首先载入这节课需要的包，以及导入数据。

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

df1 = pd.read_csv('AAPL_daily_data.csv')
```

df1.head()

	Unnamed: 0	Open	High	Low	Close	Volume
0	2018-05-31	187.2200	188.23	186.14	186.87	27482793
1	2018-06-01	187.9912	190.26	187.75	190.24	23442510
2	2018-06-04	191.6350	193.42	191.35	191.83	26266174
3	2018-06-05	193.0650	193.94	192.36	193.31	21565963
4	2018-06-06	193.6300	194.08	191.92	193.98	20933619

注：.head 用于查看这个数据的前几行，一般为前五行。可以在括号内填入数字，比如说 10，就会显示前 10 行。

df1.tail()

	Unnamed: 0	Open	High	Low	Close	Volume
95	2018-10-15	221.16	221.83	217.2700	217.36	30791007
96	2018-10-16	218.93	222.99	216.7627	222.15	29183963
97	2018-10-17	222.30	222.64	219.3400	221.19	22885397
98	2018-10-18	217.86	219.74	213.0000	216.02	32581315
99	2018-10-19	218.06	221.26	217.4300	219.31	33078726

注：.tail 用于查看这个数据的末几行，一般默认数据的末五行。用法与.head 相似。

```
plt.figure(figsize = (10,5))
plt.plot(df1.index,df1.Close)
plt.title('AAPL daily close price')
plt.show()
```

注：

这是一个收盘价格随时间变化的走势图。

df1.Close 等同于 df1['Close'] ,但它不是一个通用的写法 ,如果标题含有空格的话，就不方便使用了。

Tip：如何更新列的名字

首先可以使用 df1.columns 来查看当前每一列的名字。

第二步可以直接复制每一列的名字，加在 df1.columns=后，并根据实际情况进行修改，这里把“Unnamed: 0”改为：Date。

最后，我们可以使用 df1.head()来查看，这个时候就会发现列的名字已经改好了。

更新完列的名字后，我们用修改好的列名来画图

但当我们运行行代码时，出现了报错：

```
ValueError: could not convert string to float: '2018-10-19'
```

这个报错的含义是不能将一个字符串转化成为一个小数，因为在画图的时候，默认横纵坐标都是数字，但是当我们查看”Date”的类型时，发现它是一个字符串。所以我们只能先把它变回 index，同时我们可以利用 plt.xticks 对横坐标的值进行改写。

```
plt.figure(figsize = (10,5))
plt.plot(df1.index,df1['Close'])
plt.xticks(df1.index,df1.Date)
plt.title('AAPL daily close price')
plt.show()
```

这里它自动显示了所有的日期，导致我们看不清楚，其实我们只需要几个关键的时间点就可以了。

```
k = list(range(0,100,15))
k
```

```
[0, 15, 30, 45, 60, 75, 90]
```

注：

① 使用 range(0,100,15)来返回一串从 0 到 100 间隔为 15 的整数序列

② 这里我们要把它设置成一个列表。

```
plt.figure(figsize = (10,5))
plt.plot(df1.index,df1['Close'])
plt.xticks(df1.index[k],df1.Date[k])
plt.title('AAPL daily close price')
plt.show()
```

注 :把 k 填入 plt.xticks(df1.index[k],df1.Date[k])中 ,对这 10 个值同时进行索引。

一、定义时间变量 – Datetime

在 Python 里有一个专门为表示时间变量的命名，即 Datetime。

例 1：

df1['Date'] = pd.to_datetime(df1['Date'], format ='%Y-%m-%d')

注：使用 pd.to_datetime()这个函数，括号内填入的是原来的 Date，format 这个参数可以设置在我们给的这个字符串内，包含时间变量的字符的形式。

type(df1['Date'][0])

```
pandas._libs.tslibs.timestamps.Timestamp
```

注：我们可以看到，它已经变成了一个全新的变量。

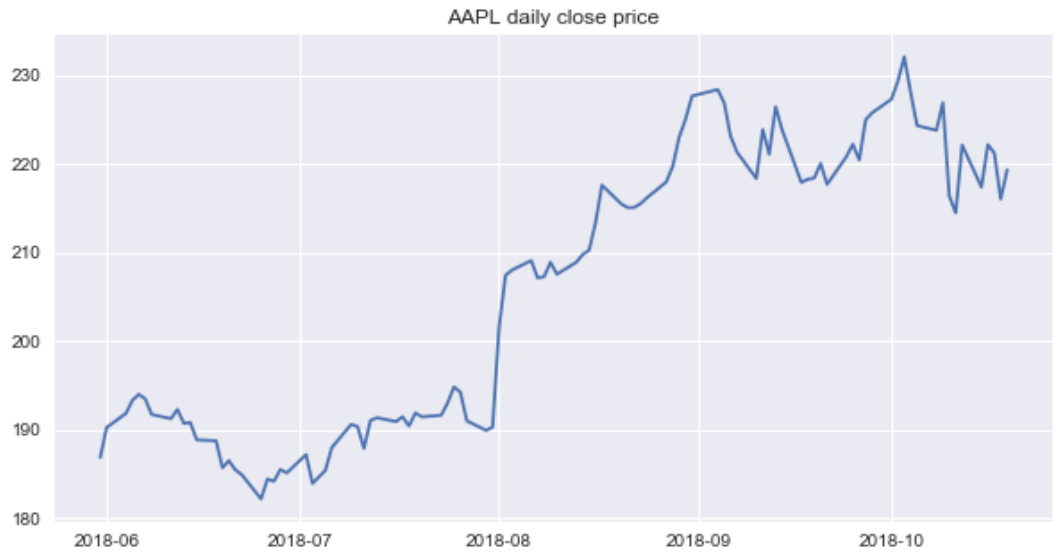
下面我们再来试试之前运行不了的代码：

plt.figure(figsize = (10,5))

plt.plot(df1['Date'],df1['Close'])

plt.title('AAPL daily close price')

plt.show()



有没有什么更快的方法来写代码呢？

pandas 的 series 自带索引，如果我们不需要固定两列设置横纵坐标，那么可以使用一下方法：

df1.index = df1['Date']

df1.head()

Date	Date	Open	High	Low	Close	Volume
2018-05-31	2018-05-31	187.2200	188.23	186.14	186.87	27482793
2018-06-01	2018-06-01	187.9912	190.26	187.75	190.24	23442510
2018-06-04	2018-06-04	191.6350	193.42	191.35	191.83	26266174
2018-06-05	2018-06-05	193.0650	193.94	192.36	193.31	21565963
2018-06-06	2018-06-06	193.6300	194.08	191.92	193.98	20933619

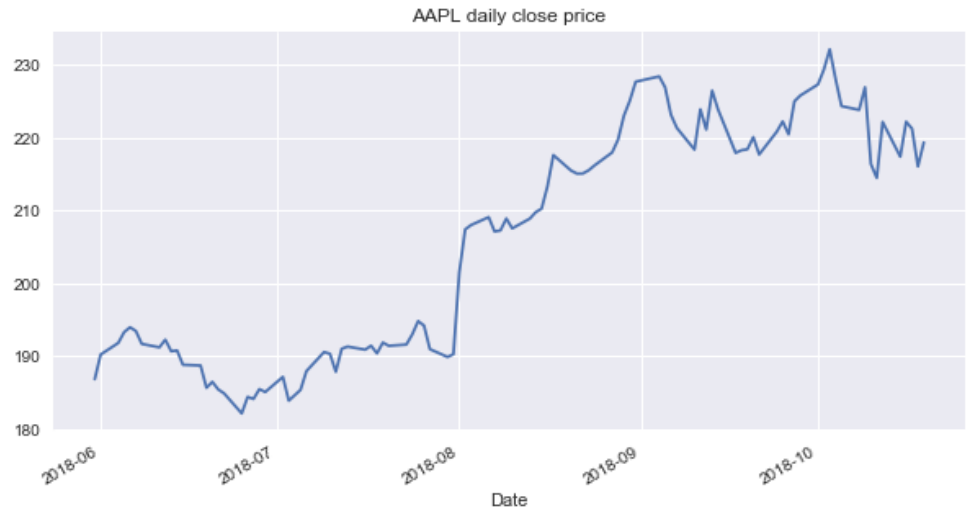
注：把 index 变成日期，直接对 index 进行赋值。

plt.figure(figsize = (10,5))

df1['Close'].plot()

plt.title('AAPL daily close price')

plt.show()



注：.plot 默认 x 轴就是 index。

例 2：

df2 = pd.read_csv('GOOG_daily_data.csv', index_col = 0)

df2.head()

	Open	High	Low	Close	Volume
2018-05-31	1067.56	1097.19	1067.56	1084.99	3088305
2018-06-01	1099.35	1120.00	1098.50	1119.50	2421598
2018-06-04	1122.33	1141.89	1122.01	1139.29	1889579
2018-06-05	1140.99	1145.74	1133.19	1139.66	1677973
2018-06-06	1142.17	1143.00	1125.74	1136.88	1698247

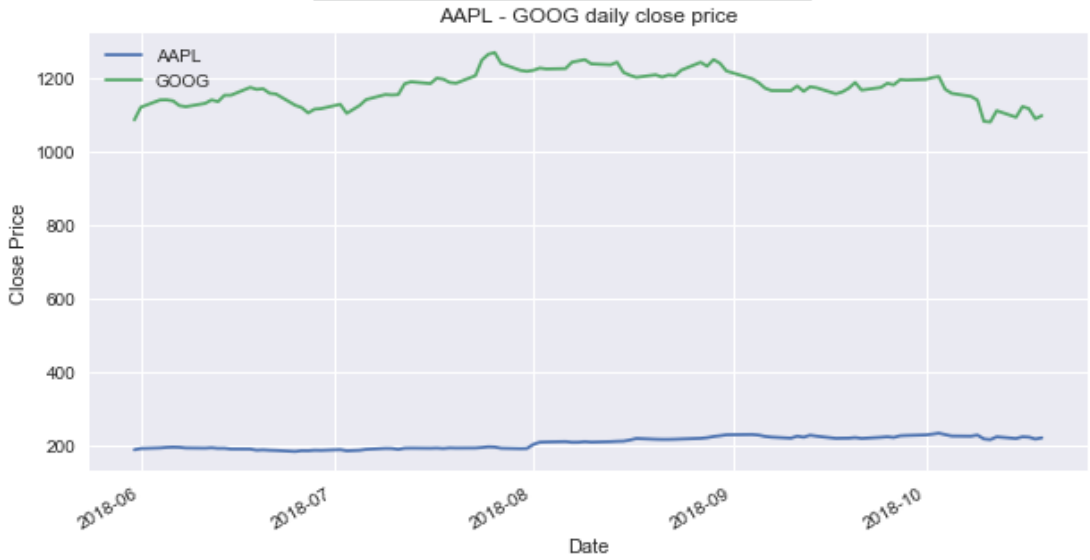
注：这里显示的日期不是一个时期或时间变量，所有我们要对它进行修改：

df2.index = pd.to_datetime(df2.index , format = '%Y-%m-%d')

df2.index

```
DatetimeIndex(['2018-05-31', '2018-06-01', '2018-06-04', '2018-06-05',
               '2018-06-06', '2018-06-07', '2018-06-08', '2018-06-11',
               '2018-06-12', '2018-06-13', '2018-06-14', '2018-06-15',
               '2018-06-18', '2018-06-19', '2018-06-20', '2018-06-21',
               '2018-06-22', '2018-06-25', '2018-06-26', '2018-06-27',
               '2018-06-28', '2018-06-29', '2018-07-02', '2018-07-03',
               '2018-07-05', '2018-07-06', '2018-07-09', '2018-07-10',
               '2018-07-11', '2018-07-12', '2018-07-13', '2018-07-16',
               '2018-07-17', '2018-07-18', '2018-07-19', '2018-07-20',
               '2018-07-23', '2018-07-24', '2018-07-25', '2018-07-26',
               '2018-07-27', '2018-07-30', '2018-07-31', '2018-08-01',
               '2018-08-02', '2018-08-03', '2018-08-06', '2018-08-07',
               '2018-08-08', '2018-08-09', '2018-08-10', '2018-08-13',
               '2018-08-14', '2018-08-15', '2018-08-16', '2018-08-17',
               '2018-08-20', '2018-08-21', '2018-08-22', '2018-08-23',
               '2018-08-24', '2018-08-27', '2018-08-28', '2018-08-29',
               '2018-08-30', '2018-08-31', '2018-09-04', '2018-09-05',
               '2018-09-06', '2018-09-07', '2018-09-10', '2018-09-11',
               '2018-09-12', '2018-09-13', '2018-09-14', '2018-09-17',
               '2018-09-18', '2018-09-19', '2018-09-20', '2018-09-21',
               '2018-09-24', '2018-09-25', '2018-09-26', '2018-09-27',
               '2018-09-28', '2018-10-01', '2018-10-02', '2018-10-03',
               '2018-10-04', '2018-10-05', '2018-10-08', '2018-10-09',
               '2018-10-10', '2018-10-11', '2018-10-12', '2018-10-15',
               '2018-10-16', '2018-10-17', '2018-10-18', '2018-10-19'],
              dtype='datetime64[ns]', freq=None)
```

```
plt.figure(figsize = (10,5))
df1['Close'].plot(label = 'AAPL')
df2['Close'].plot(label = 'GOOG')
plt.title('AAPL - GOOG daily close price')
plt.legend()
plt.ylabel('Close Price')
plt.show()
```



注：利用上节课所学的知识，把两个图放在一个图上面。

二、使用 fig.add_subplot() 和 ax

可以在一个图上面显示一个数据，但同时画很多个图。

例 1：

```
fig = plt.figure(figsize = (20,25))

ax = fig.add_subplot(4,2,1)
df1.Open.plot(label = 'Open')
plt.legend()
plt.ylabel('Price')

ax = fig.add_subplot(4,2,2)
df1.High.plot(label = 'High')
plt.legend()
plt.ylabel('Price')

ax = fig.add_subplot(4,2,2)
df1.Low.plot(label = 'Low')
plt.legend()
plt.ylabel('Price')

ax = fig.add_subplot(4,2,4)
df1.Close.plot(label = 'Close')
plt.legend()
plt.ylabel('Price')
```

plt.show()

注：

① fig 指的是所有的图所在的这一片的区域。

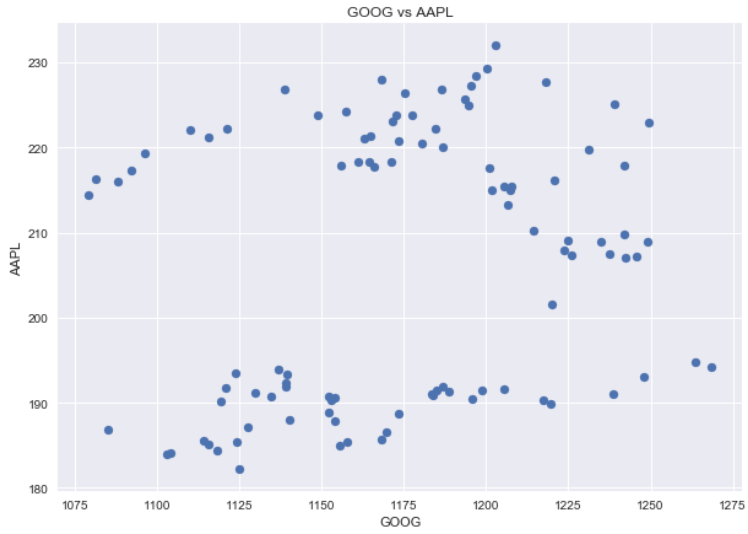
② fig.add_subplot()是增加一个小图的意思，括号内第一个参数表示的是一共要画多少个小图，第二个参数指的是要在每一行画多少个图，最后一个参数指的是现在画的图是第几张图。把两个第三个参数改成一样的，就能把两个这个两个数据画在一个图上了。

③ ax 也是一个区域的概念，它指的是每一个小图确切的范围是哪里。

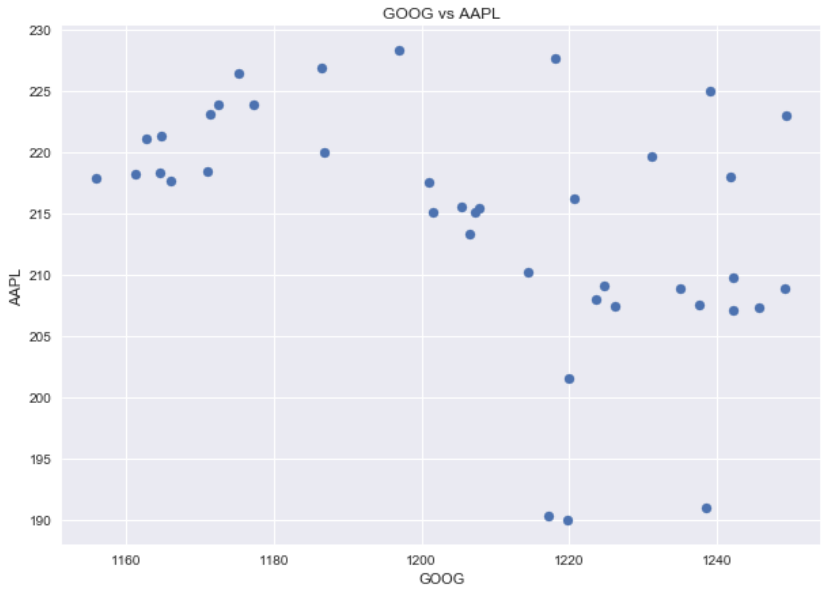
例 2：

```
plt.figure(figsize = (10,7))
plt.scatter(df2.Close, df1.Close)
plt.title('GOOG vs AAPL')
plt.xlabel('GOOG')
plt.ylabel('AAPL')
```

```
plt.show()
```

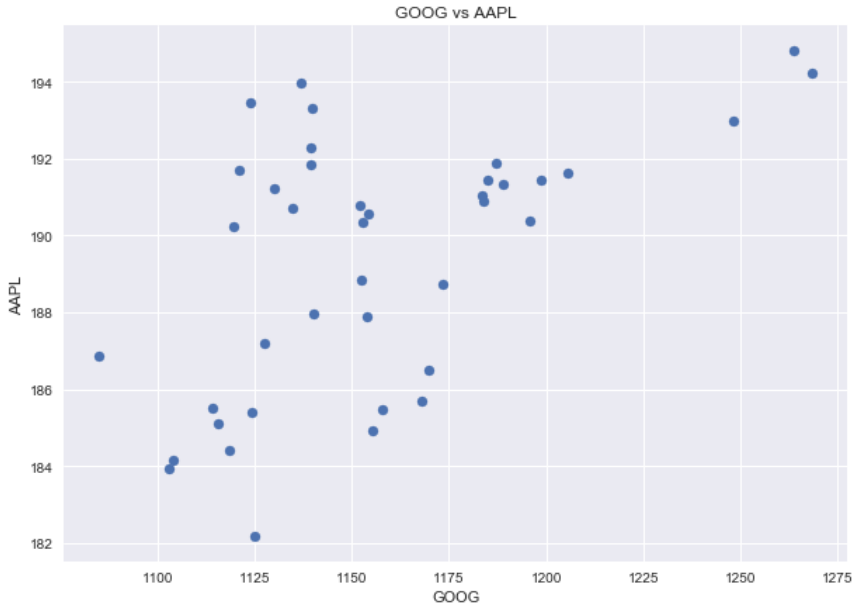



```
plt.figure(figsize = (10,7))
plt.scatter(df2.Close[40:80], df1.Close[40:80])
plt.title('GOOG vs AAPL')
plt.xlabel('GOOG')
plt.ylabel('AAPL')
plt.show()
```



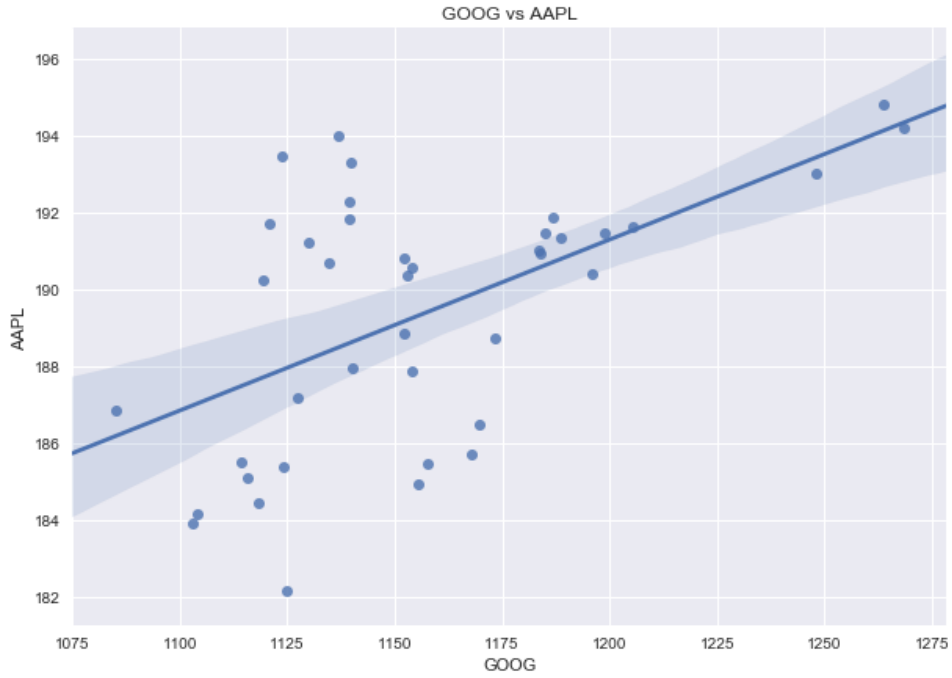
注：这两个数据 40-80 天的比较。

```
plt.figure(figsize = (10,7))
plt.scatter(df2.Close[:40], df1.Close[:40])
plt.title('GOOG vs AAPL')
plt.xlabel('GOOG')
plt.ylabel('AAPL')
plt.show()
```



注：两个数据前 40 前的比较。

```
plt.figure(figsize = (10,7))
sns.regplot(df2.Close[:40], df1.Close[:40])
plt.title('GOOG vs AAPL')
plt.xlabel('GOOG')
plt.ylabel('AAPL')
plt.show()
```



注：使用 sns.regplot()能够自动画上最贴近它们趋势的一根线，并且还会画上它们的置信区间。

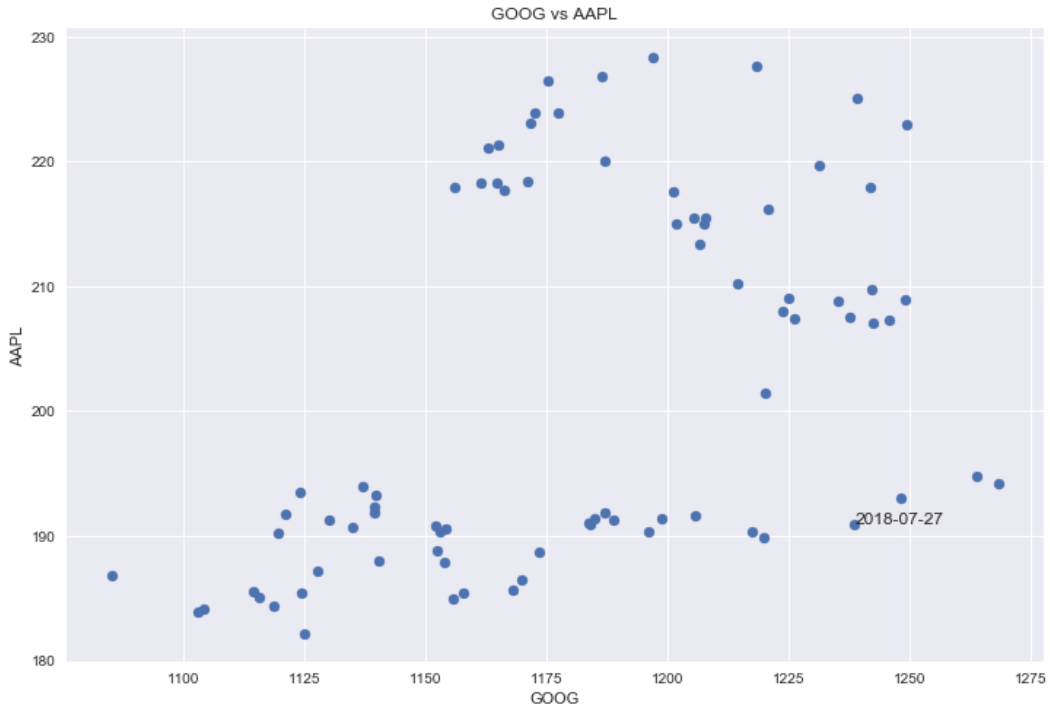
三、使用 ax.annotate 对图表进行注释

例：

```
fig = plt.figure(figsize = (12,8))
ax = fig.add_subplot(1,1,1)
x = df2.Close[:80]
y = df1.Close[:80]
plt.scatter(x,y)
plt.title('GOOG vs AAPL')
plt.xlabel('GOOG')
plt.ylabel('AAPL')

ax.annotate(str(df2.index[40])[:10],(x[40],y[40]))

plt.show()
```



注：

① 在注释时用 ax 的目的是把注释的内容准确地画到图标上，

② ax.annotate()里面需要填入两个内容，逗号左边填入的是注释的内容，逗号右边填入注释的位置。

③ 只添加日期不添加时间信息时，首先把它变成一个字符串，再对字符串进行切片。

```
fig = plt.figure(figsize = (12,8))
ax = fig.add_subplot(1,1,1)
x = df2.Close[:80]
y = df1.Close[:80]
plt.scatter(x,y)
plt.title('GOOG vs AAPL')
plt.xlabel('GOOG')
plt.ylabel('AAPL')

for i in range(80):
    ax.annotate(str(df2.index[i])[:10],(x[i],y[i]))

plt.show()
```

注：想要在每个点上都进行标注，可以使用循环。

今日作业：

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

读取课件中的'AAPL_intraday_data.csv' ，完成以下练习。

Q1. 读取 csv，并且把这个 dataframe 叫做 df1。day1 是 df1 的前 390 行，表示的是一天的数据。

```
In [496]: print(len(day1))
          day1.head()
```

390

Out[496]:

	Time	Open	High	Low	Close	Volume
0	09:31:00	221.0755	221.17	220.64	220.73	986771
1	09:32:00	220.8000	221.80	220.71	221.78	161516
2	09:33:00	221.6700	221.77	221.19	221.19	130556
3	09:34:00	221.2400	221.36	220.80	220.84	127409
4	09:35:00	220.9701	221.07	220.33	220.44	152806

Q2. 使用 day1 画图，画 Close price 的折线图，使用两种方法来更改 X 轴坐标

【提示】把 Time 更改成时间变量的时候要注意两个点：

① 时间的写法： '%Y' 表示年份，' %m' 表示月份， '%d' 表示日期，' %H ' 表示小时，' %M ' 表示分钟，' %S ' 表示秒。

② 使用 pd.to_datetime(' ', format=).dt.time 和

pd.to_datetime(' ', format=).dt.date，

③ 比较和 pd.to_datetime(' ', format= ）的不同

Q3. 使用 day1 同时画两张图，左图：最高价和最低价的折线图。右图：算出每个时间点最高价最低价之间的差，把这个差用柱状图画出来