# Python 金融之旅 —— “超级Excel”

"编程是一门艺术，而不是科学" - Donald Knuth

“Pandas是一个强大的工具，它可以让你在Python中使用Excel的功能”

## 目录



@Zhongfang Yuan(Harold)

金融工程专业大四学生

申请金融工程硕士学位并担任内容创作者

---

## 1. 你想转正吗?

- 为什么不用Excel?
- 我讨厌编程！
- Python？我不喜欢这个名字，听起来像一条巨蟒。

“Python是一条巨蟒，它会把你吞下去”

让我给你讲一个关于Rivers和Jae的故事。

很久很久以前，在一个遥远的银河系里...

毕业后，Rivers和Jae在香港摩根大通找到了实习机会，成为专门研究房地产行业的股票分析师。他们都有金融背景。Rivers不会编程，因为他害怕编程！而Jae在港中大（深圳）上的FIN3080课程中学习了Python。

他们在大学时是最好的朋友，所以他们选的课程都一样，除了FIN3080，一门港中深的量化分析课。

在他们实习结束时，只有一个人可以留下。

---

一天，他们的领导，Harold，走向他们，给他们带来了一条消息，说： 嘿，你们这两个聪明人！我的客户对恒大很生气，问我们为什么之前没有建议他买入……你们试着给我一些关于何时以及为什么会发生这种情况的分析。哦，也许还可以分析一下其他公司。我有个会议要去了，今天下午之前给我。

Rivers 转向Excel来分析数据，但他只能做到中学生水平的Excel操作。 同时，Jae 尝试用Python来解决问题。 由于这是一台新电脑，Jae下载并安装了他所需的Python包。

In [ ]:
```
# ! pip install pandas
# ! pip install matplotlib  # he wants to draw pictures
# ! pip install numpy
# ! pip install seaborn

# for mac
# ! pip3 install pandas
# ! pip3 install matplotlib  # he wants to draw pictures
# ! pip3 install numpy
# ! pip3 install seaborn
```

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns


import warnings
warnings.filterwarnings("ignore")

# 1. Read data
df = pd.read_excel('3333.HK.xlsx')
```

In [ ]:
```
df.columns = ['代码', '简称', '日期', '前收盘价', '开盘价', '最高价',
              '最低价', '收盘价', '成交量', '成交金额', '涨跌(元)',
              '涨跌幅(%)', '均价(元)','换手率(%)',  'A股流通市值', 'B股流通市值
              '总市值', 'A股流通股本', 'B股流通股本', '总股本', '市盈率', '市净

# Data type conversion and cleaning

df['代码'] = df['代码'].astype(str)
df['简称'] = df['简称'].astype(str)

df['日期'] = pd.to_datetime(df['日期'])

df['成交量'] = df['成交量'].replace('--', None)
df['成交金额'] = df['成交金额'].replace('--', None)
df['前收盘价'] = df['前收盘价'].replace('--', None)

df['成交量'] = df['成交量'].astype(float)
df['成交金额'] = df['成交金额'].astype(float)
df['前收盘价'] = df['前收盘价'].astype(float)

df.head()
```

Out[ ]:

| | 代码 | 简称 | 日期 | 前收盘价 | 开盘价 | 最高价 | 最低价 | 收盘价 | 成交量 | 成交金额 | ... | A股流通市值 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3333.HK | 中国恒大 | 2009-11-05 | NaN | 4.00 | 4.80 | 3.81 | 4.70 | 1.205691e+09 | 4.951375e+09 | ... | -- |
| **1** | 3333.HK | 中国恒大 | 2009-11-06 | 4.70 | 4.65 | 4.80 | 4.41 | 4.58 | 4.855230e+08 | 2.211907e+09 | ... | -- |
| **2** | 3333.HK | 中国恒大 | 2009-11-09 | 4.58 | 4.60 | 4.85 | 4.58 | 4.69 | 1.673497e+08 | 7.893678e+08 | ... | -- |
| **3** | 3333.HK | 中国恒大 | 2009-11-10 | 4.69 | 4.73 | 4.77 | 4.45 | 4.58 | 1.084150e+08 | 4.972369e+08 | ... | -- |
| **4** | 3333.HK | 中国恒大 | 2009-11-11 | 4.58 | 4.59 | 4.60 | 4.36 | 4.41 | 6.849600e+07 | 3.056243e+08 | ... | -- |

5 rows × 24 columns

数据看起来很混乱，但不用担心，Jae上过FIN3080，他见过更糟糕的情况。

In [ ]:

```
df.columns = ['代码', '简称', '日期', '前收盘价', '开盘价', '最高价',
              '最低价', '收盘价', '成交量', '成交金额', '涨跌(元)',
              '涨跌幅(%)', '均价(元)','换手率(%)',  'A股流通市值', 'B股流通市值
              '总市值', 'A股流通股本', 'B股流通股本', '总股本', '市盈率', '市净

# Data type conversion and cleaning

df['代码'] = df['代码'].astype(str)
df['简称'] = df['简称'].astype(str)

df['日期'] = pd.to_datetime(df['日期'])

df['成交量'] = df['成交量'].replace('--', None)
df['成交金额'] = df['成交金额'].replace('--', None)
df['前收盘价'] = df['前收盘价'].replace('--', None)

df['成交量'] = df['成交量'].astype(float)
df['成交金额'] = df['成交金额'].astype(float)
df['前收盘价'] = df['前收盘价'].astype(float)

df.head()
```

Out[ ]:

| | 代码 | 简称 | 日期 | 前收盘价 | 开盘价 | 最高价 | 最低价 | 收盘价 | 成交量 | 成交金额 | ... | A股流通市值 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3333.HK | 中国恒大 | 2009-11-05 | NaN | 4.00 | 4.80 | 3.81 | 4.70 | 1.205691e+09 | 4.951375e+09 | ... | -- |
| **1** | 3333.HK | 中国恒大 | 2009-11-06 | 4.70 | 4.65 | 4.80 | 4.41 | 4.58 | 4.855230e+08 | 2.211907e+09 | ... | -- |
| **2** | 3333.HK | 中国恒大 | 2009-11-09 | 4.58 | 4.60 | 4.85 | 4.58 | 4.69 | 1.673497e+08 | 7.893678e+08 | ... | -- |
| **3** | 3333.HK | 中国恒大 | 2009-11-10 | 4.69 | 4.73 | 4.77 | 4.45 | 4.58 | 1.084150e+08 | 4.972369e+08 | ... | -- |
| **4** | 3333.HK | 中国恒大 | 2009-11-11 | 4.58 | 4.59 | 4.60 | 4.36 | 4.41 | 6.849600e+07 | 3.056243e+08 | ... | -- |

5 rows × 24 columns

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3412 entries, 0 to 3411
Data columns (total 24 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   代码      3412 non-null   object
 1   简称      3412 non-null   object
 2   日期      3412 non-null   datetime64[ns]
 3   前收盘价    3411 non-null   float64
 4   开盘价     3412 non-null   float64
 5   最高价     3412 non-null   float64
 6   最低价     3412 non-null   float64
 7   收盘价     3412 non-null   float64
 8   成交量     3046 non-null   float64
 9   成交金额    3046 non-null   float64
 10  涨跌(元)   3412 non-null   float64
 11  涨跌幅(%)  3412 non-null   float64
 12  均价(元)   3412 non-null   object
 13  换手率(%)  3412 non-null   object
 14  A股流通市值  3412 non-null   object
 15  B股流通市值  3412 non-null   object
 16  总市值     3412 non-null   float64
 17  A股流通股本  3412 non-null   object
 18  B股流通股本  3412 non-null   object
 19  总股本     3412 non-null   int64
 20  市盈率     3412 non-null   float64
 21  市净率     3412 non-null   float64
 22  市销率     3412 non-null   object
 23  市现率     3412 non-null   object
dtypes: datetime64[ns](1), float64(12), int64(1), object(10)
memory usage: 639.9+ KB
```

In [ ]:
```
df.describe()
```

Out[ ]:

| | 日期 | 前收盘价 | 开盘价 | 最高价 | 最低价 | |
|---|---|---|---|---|---|---|
| **count** | 3412 | 3411.000000 | 3412.000000 | 3412.000000 | 3412.000000 | 3412.0 |
| **mean** | 2016-10-05 00:08:26.447831296 | 14.170833 | 14.194997 | 14.463860 | 13.876339 | 14.1 |
| **min** | 2009-11-05 00:00:00 | 0.530000 | 0.430000 | 0.590000 | 0.430000 | 0.5 |
| **25%** | 2013-04-18 18:00:00 | 3.540000 | 3.550000 | 3.630000 | 3.470000 | 3.5 |
| **50%** | 2016-10-04 12:00:00 | 5.040000 | 5.045000 | 5.170000 | 4.920000 | 5.0 |
| **75%** | 2020-03-23 06:00:00 | 28.465000 | 28.520000 | 29.000000 | 27.730000 | 28.4 |
| **max** | 2023-09-11 00:00:00 | 52.550000 | 53.030000 | 53.770000 | 51.620000 | 52.5 |
| **std** | NaN | 14.739339 | 14.770053 | 15.057941 | 14.415812 | 14.7 |

In [ ]:
```
df.sort_values(["代码", "日期"], inplace=True)
```

In [ ]:
```
df['收盘价'] = df['收盘价'].fillna(method='ffill')

df['日频收益'] = df['收盘价'].pct_change()
df['日频收益_log'] = df['收盘价'].apply(lambda x: np.log(x)).diff()

df['累计收益'] = (df['日频收益'] + 1).cumprod()
df['累计收益_2'] = np.exp(df['日频收益'].apply(lambda x: np.log(x+1)).cumsum
df['累计收益_3'] = df['日频收益'].apply(lambda x: np.log(x+1)).cumsum().appl

df.head()
```

Out[ ]:

| | 代码 | 简称 | 日期 | 前收盘价 | 开盘价 | 最高价 | 最低价 | 收盘价 | 成交量 | 成交金额 | ... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3333.HK | 中国恒大 | 2009-11-05 | NaN | 4.00 | 4.80 | 3.81 | 4.70 | 1.205691e+09 | 4.951375e+09 | ... | 15 |
| **1** | 3333.HK | 中国恒大 | 2009-11-06 | 4.70 | 4.65 | 4.80 | 4.41 | 4.58 | 4.855230e+08 | 2.211907e+09 | ... | 15 |
| **2** | 3333.HK | 中国恒大 | 2009-11-09 | 4.58 | 4.60 | 4.85 | 4.58 | 4.69 | 1.673497e+08 | 7.893678e+08 | ... | 15 |
| **3** | 3333.HK | 中国恒大 | 2009-11-10 | 4.69 | 4.73 | 4.77 | 4.45 | 4.58 | 1.084150e+08 | 4.972369e+08 | ... | 15 |
| **4** | 3333.HK | 中国恒大 | 2009-11-11 | 4.58 | 4.59 | 4.60 | 4.36 | 4.41 | 6.849600e+07 | 3.056243e+08 | ... | 15 |

5 rows × 29 columns

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3412 entries, 0 to 3411
Data columns (total 29 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   代码        3412 non-null   object
 1   简称        3412 non-null   object
 2   日期        3412 non-null   datetime64[ns]
 3   前收盘价      3411 non-null   float64
 4   开盘价       3412 non-null   float64
 5   最高价       3412 non-null   float64
 6   最低价       3412 non-null   float64
 7   收盘价       3412 non-null   float64
 8   成交量       3046 non-null   float64
 9   成交金额      3046 non-null   float64
 10  涨跌(元)     3412 non-null   float64
 11  涨跌幅(%)    3412 non-null   float64
 12  均价(元)     3412 non-null   object
 13  换手率(%)    3412 non-null   object
 14  A股流通市值    3412 non-null   object
 15  B股流通市值    3412 non-null   object
 16  总市值       3412 non-null   float64
 17  A股流通股本    3412 non-null   object
 18  B股流通股本    3412 non-null   object
 19  总股本       3412 non-null   int64
 20  市盈率       3412 non-null   float64
 21  市净率       3412 non-null   float64
 22  市销率       3412 non-null   object
 23  市现率       3412 non-null   object
 24  日频收益      3411 non-null   float64
 25  日频收益_log  3411 non-null   float64
 26  累计收益      3411 non-null   float64
 27  累计收益_2    3411 non-null   float64
 28  累计收益_3    3411 non-null   float64
dtypes: datetime64[ns](1), float64(17), int64(1), object(10)
memory usage: 773.2+ KB
```

In [ ]:
```
df.describe()
```

Out[ ]:

| | 日期 | 前收盘价 | 开盘价 | 最高价 | 最低价 | |
|---|---|---|---|---|---|---|
| **count** | 3412 | 3411.000000 | 3412.000000 | 3412.000000 | 3412.000000 | 3412.0 |
| **mean** | 2016-10-05 00:08:26.447831296 | 14.170833 | 14.194997 | 14.463860 | 13.876339 | 14.1 |
| **min** | 2009-11-05 00:00:00 | 0.530000 | 0.430000 | 0.590000 | 0.430000 | 0.5 |
| **25%** | 2013-04-18 18:00:00 | 3.540000 | 3.550000 | 3.630000 | 3.470000 | 3.5 |
| **50%** | 2016-10-04 12:00:00 | 5.040000 | 5.045000 | 5.170000 | 4.920000 | 5.0 |
| **75%** | 2020-03-23 06:00:00 | 28.465000 | 28.520000 | 29.000000 | 27.730000 | 28.4 |
| **max** | 2023-09-11 00:00:00 | 52.550000 | 53.030000 | 53.770000 | 51.620000 | 52.5 |
| **std** | NaN | 14.739339 | 14.770053 | 15.057941 | 14.415812 | 14.7 |

In [ ]:
```
df.sort_values(["代码", "日期"], inplace=True)
```

有许多方法可以计算投资回报，需要注意的是，我们需要考虑股息和股票价格的变化。

In [ ]:
```
df['收盘价'] = df['收盘价'].fillna(method='ffill')

df['日频收益'] = df['收盘价'].pct_change()
df['日频收益_log'] = df['收盘价'].apply(lambda x: np.log(x)).diff()

df['累计收益'] = (df['日频收益'] + 1).cumprod()
df['累计收益_2'] = np.exp(df['日频收益'].apply(lambda x: np.log(x+1)).cumsum
df['累计收益_3'] = df['日频收益'].apply(lambda x: np.log(x+1)).cumsum().appl

df.head()
```

Out[ ]:

| | 代码 | 简称 | 日期 | 前收盘价 | 开盘价 | 最高价 | 最低价 | 收盘价 | 成交量 | 成交金额 | ... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3333.HK | 中国恒大 | 2009-11-05 | NaN | 4.00 | 4.80 | 3.81 | 4.70 | 1.205691e+09 | 4.951375e+09 | ... | 15 |
| **1** | 3333.HK | 中国恒大 | 2009-11-06 | 4.70 | 4.65 | 4.80 | 4.41 | 4.58 | 4.855230e+08 | 2.211907e+09 | ... | 15 |
| **2** | 3333.HK | 中国恒大 | 2009-11-09 | 4.58 | 4.60 | 4.85 | 4.58 | 4.69 | 1.673497e+08 | 7.893678e+08 | ... | 15 |
| **3** | 3333.HK | 中国恒大 | 2009-11-10 | 4.69 | 4.73 | 4.77 | 4.45 | 4.58 | 1.084150e+08 | 4.972369e+08 | ... | 15 |
| **4** | 3333.HK | 中国恒大 | 2009-11-11 | 4.58 | 4.59 | 4.60 | 4.36 | 4.41 | 6.849600e+07 | 3.056243e+08 | ... | 15 |

5 rows × 29 columns

In [ ]:
```
df["close_ma_5"] = df.groupby("代码")["收盘价"].transform(lambda x : x.roll
df["close_ma_30"] = df.groupby("代码")["收盘价"].transform(lambda x : x.rol
```

In [ ]:
```
import matplotlib.pyplot as plt

# In Windows:
# plt.rcParams['font.family'] = 'SimHei'

plt.rcParams['font.family'] = ['Arial Unicode MS']
plt.rcParams['figure.figsize'] = [5, 3]
```

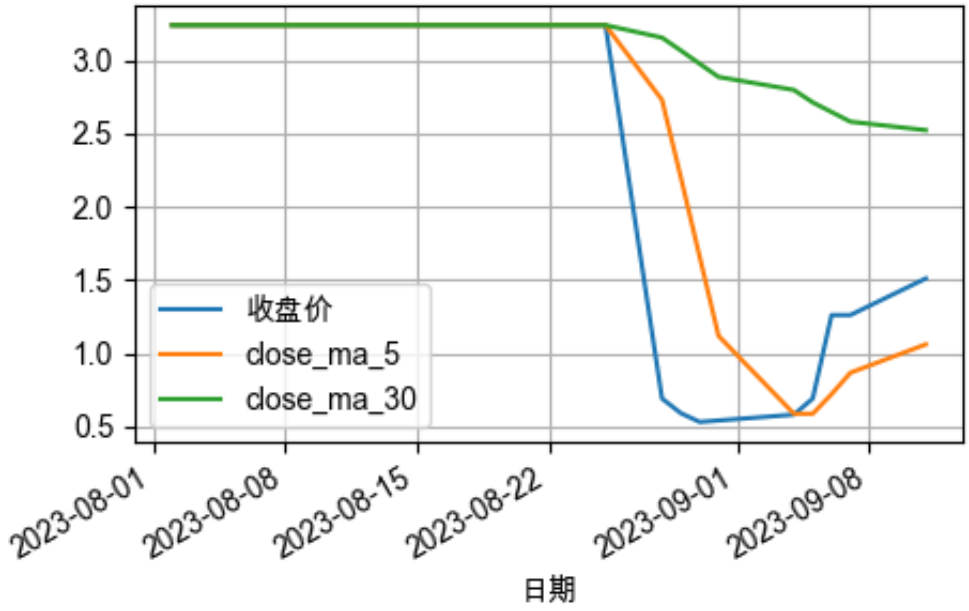
In [ ]:
```
df.set_index("日期", inplace=True)

df[["收盘价", "close_ma_5", "close_ma_30"]].plot()

plt.xlabel("日期")

plt.legend(["收盘价", "close_ma_5", "close_ma_30"])
# 设置绘图标题
plt.title("close price 000005.SZ")
# 格子
plt.grid()
plt.show()
```


In [ ]:
```
df_recent = df[df.index > '2023-08-01']
df_recent[["收盘价", "close_ma_5", "close_ma_30"]].plot()
plt.grid()
plt.show()
```




In [ ]:
```
df_recent[df_recent.index >= '2023-08-24'][["收盘价", '涨跌幅(%)','换手率(%)
```

Out[ ]:

| 日期 | 收盘价 | 涨跌幅(%) | 换手率(%) | 成交金额 | 市盈率 | 市净率 | 总股本 | close_ma_ |
|---|---|---|---|---|---|---|---|---|
| 2023-08-24 | 3.24 | 0.0000 | -- | NaN | -0.1838 | -0.0400 | 13204300900 | 3.24 |
| 2023-08-25 | 3.24 | 0.0000 | -- | NaN | -0.1838 | -0.0400 | 13204300900 | 3.24 |
| 2023-08-28 | 0.69 | -78.7879 | 13.9933 | 582346873.0 | -0.0587 | -0.0077 | 13204300900 | 2.73 |
| 2023-08-29 | 0.59 | -14.2857 | 3.9602 | 165698771.0 | -0.0503 | -0.0066 | 13204300900 | 2.20 |
| 2023-08-30 | 0.53 | -10.0000 | 1.9645 | 71924595.0 | -0.0453 | -0.0059 | 13204300900 | 1.65 |
| 2023-08-31 | 0.54 | 1.8519 | 1.6063 | 60117438.0 | -0.0461 | -0.0060 | 13204300900 | 1.11 |
| 2023-09-04 | 0.58 | 7.2727 | 1.3593 | 52910752.0 | -0.0495 | -0.0065 | 13204300900 | 0.58 |
| 2023-09-05 | 0.69 | 18.6441 | 1.9047 | 81113253.0 | -0.0587 | -0.0077 | 13204300900 | 0.58 |
| 2023-09-06 | 1.26 | 82.8571 | 9.6291 | 633395325.0 | -0.1074 | -0.0141 | 13204300900 | 0.72 |
| 2023-09-07 | 1.26 | 0.0000 | 7.6003 | 691153067.0 | -0.1074 | -0.0141 | 13204300900 | 0.86 |
| 2023-09-11 | 1.51 | 20.3125 | 4.019 | 368973167.0 | -0.1292 | -0.0170 | 13204300900 | 1.06 |

In [ ]:
```
df_2007 = pd.read_excel('2007.HK.xlsx')
df_2007.columns = ['代码', '简称', '日期', '前收盘价', '开盘价', '最高价',
                   '最低价', '收盘价', '成交量', '成交金额', '涨跌(元)',
                   '涨跌幅(%)', '均价(元)','换手率(%)',  'A股流通市值', 'B股流通市值
                   '总市值', 'A股流通股本', 'B股流通股本', '总股本', '市盈率', '市净

df_2007['代码'] = df_2007['代码'].astype(str)
df_2007['简称'] = df_2007['简称'].astype(str)
df_2007['日期'] = pd.to_datetime(df_2007['日期'])
df_2007['成交量'] = df_2007['成交量'].replace('--', None)
df_2007['成交金额'] = df_2007['成交金额'].replace('--', None)
df_2007['前收盘价'] = df_2007['前收盘价'].replace('--', None)
df_2007['成交量'] = df_2007['成交量'].astype(float)
df_2007['成交金额'] = df_2007['成交金额'].astype(float)
df_2007['前收盘价'] = df_2007['前收盘价'].astype(float)

df_2007.set_index("日期", inplace=True)
df_2007_select = df_2007[df_2007.index >= '2023-08-29'][["收盘价", '涨跌幅(
df_2007_select
```

Out[ ]:

| 日期 | 收盘价 | 涨跌幅(%) | 换手率(%) | 成交金额 | 市盈率 | 市净率 | 总股本 |
|---|---|---|---|---|---|---|---|
| 2023-08-29 | 1.86 | 12.3457 | 2.9593 | 7.145545e+08 | -3.7122 | 0.1131 | 27637858596 |
| 2023-08-30 | 1.80 | -3.2967 | 2.2952 | 5.652364e+08 | -0.4033 | 0.1436 | 27637858596 |
| 2023-08-31 | 1.82 | 1.1364 | 2.0757 | 5.179268e+08 | -0.4079 | 0.1452 | 27637858596 |
| 2023-09-04 | 2.08 | 14.6067 | 6.174 | 1.741268e+09 | -0.4675 | 0.1663 | 27988507946 |
| 2023-09-05 | 2.06 | -0.9804 | 2.9799 | 8.378436e+08 | -0.4688 | 0.1670 | 27988507946 |
| 2023-09-06 | 2.49 | 20.7921 | 8.5585 | 2.809730e+09 | -0.5663 | 0.2021 | 27988507946 |
| 2023-09-07 | 2.19 | -12.2951 | 6.9157 | 2.292240e+09 | -0.4966 | 0.1773 | 27988507946 |
| 2023-09-11 | 2.11 | -3.7383 | 4.7779 | 1.386162e+09 | -0.4781 | 0.1711 | 27988507946 |

In [ ]:
```
import matplotlib.pyplot as plt

df_recent = df_recent[df_recent.index >= '2023-08-29'][["收盘价", '涨跌幅(%

plt.figure(figsize=(10, 6))
plt.plot(df_2007_select.index, df_2007_select['收盘价'], label='碧桂园')
plt.plot(df_recent.index, df_recent['收盘价'], label='恒大')
plt.xlabel('日期')
plt.ylabel('收盘价')
plt.legend()
plt.title('Comparison of Close Prices')

# Add a vertical line at August 31st
plt.axvline(x=pd.to_datetime('2023-09-04'), color='red', linestyle='--',
plt.legend()
plt.show()
```


◆**一周行情回顾**：本周（2023/09/04-2023/09/10）申万房地产指数下跌 1.10%，跑输上证综指 0.57pct，在各类板块中位列第 21/31，恒生地产建筑业指数上涨 2.13%，跑赢恒生综指 2.93pct。本周涨幅前 3 的地产公司分别为：融创中国（163.83%）、中国恒大（132.73%）、融信中国（74.42%）；本周跌幅前 3 的地产公司分别为：德信中国（-25.71%）、天房发展（-18.84%）、弘阳地产（-18.34%）。本周恒生物业服务及管理板块上涨 4.47%，跑赢恒生综指 5.27pct，在各类板块中位列第 1/13。本周涨幅前 3 的物业公司分别为：恒大物业（15.94%）、新城悦服务（14.84%）、融创服务（14.29%）；本周跌幅前 3 的物业公司分别为：合景悠活（-12.35%）、弘阳服务（-5.10%）、建业新生活（-4.33%）。

最终，Jae给老板留下了深刻印象，留在了公司，而Rivers离开了公司。 这个故事告诉我们，学习Python是非常重要的。

1. Python Pandas 的基础操作

---

## 1. 版本控制

使用 `requirements.txt` 文件来维护依赖和版本。这确保了在不同环境下工作或与他人共享代码时的一致性。

```
plaintext
pandas==1.5.3
matplotlib==3.7.1
numpy==1.24.3
scikit-learn==1.3.0
statsmodels==0.14.0
requests==2.31.0
seaborn==0.11.2
```

```
! pip install -r requirements.txt
# 或者
! pip install pandas matplotlib numpy
```

例如，用 `pd.concat` 替换 `pd.append`：

之前

df = df.append(new_data)

之后

df = pd.concat([df, new_data], axis=0)

---

---

## 2. 本地路径

```
plaintext
不良实践（绝对路径）
使用绝对路径会给我们亲爱的助教带来非常非常难受的生活
```

```
file_path =
"C:/Users/ILOVECODING/MYCAT/username/Documents/data.csv"
```

更好的实践（相对路径）

```
file_path = "./data/data.csv"
df = pd.read_csv(file_path)
df = pd.read_csv('data.csv')
```

---

---

## 3. 变量命名

```
plaintext
不良实践
使用绝对路径会给我们亲爱的助教带来非常非常难受的生活
```

```
a = pd.read_csv('data1.csv')
b = pd.read_csv('data2.csv')
df = pd.read_csv('data3.csv')
i_do_not_know_what_should_i_name_this_variable = pd.concat([df,
a], axis=0) # axis = 0 表示垂直追加数据，这是我们在大多数情况下想要的
```

或许相对较好的实践（我的方法）

```
df_stock_prices = pd.read_csv('data1.csv')
df_daily_returns = pd.read_csv('data2.csv')
df_combine_price_and_return =  pd.concat([df_stock_prices,
df_daily_returns], axis=0)
```

---

---

### 4. 为你的代码添加注释，为你的代码添加注释，为你的代码添加注释

```
def win_rate_test(sig_df, rets_df, win, negative=False):
    """
    函数: 指标胜率测试 (Function: Indicator Win Rate Test)
    :param sig_df: 单因子序列 (Parameter: Single-factor Time
Series)
    :param rets_df: 单标的收益率序列 (Parameter: Returns Time
Series for a Single Asset)
    :param negative: 多空头信号，默认为空头 (Parameter: Long/Short
Signal, Default is Short)

    返回: DProfVol_window_size的DataFrame (Return: DataFrame of
Win Rates for Different Window Sizes)
    """

    # 反转收益率数据框
    rev_rets_df = rets_df[::-1]
    # 计算总信号数
    total = sig_df.sum().values[0]
    # 初始化一个空列表以记录胜率
    record = []
    # 计算滚动收益并反转它们
    rolling_rets = rev_rets_df.rolling(win).apply(lambda x:
x.add(1).prod() - 1)
    rolling_rets = rolling_rets[::-1]
    # 初始化一个变量来计算赢得的次数
    win_num = 0
    # 遍历信号数据框中的每个日期
    for date in sig_df.index:
        if (sig_df.loc[date].values[0] == 1):
            # 检查信号是多头（正）还是空头（负）
            if (negative == False) and (rolling_rets.loc[date]
<= 0):
                win_num += 1
            elif (negative == True) and (rolling_rets.loc[date]
>= 0):
                win_num += 1
    # 计算当前窗口大小的胜率
    win_ratio = win_num / total
    # 将胜率添加到记录列表中
    record.append(win_ratio)
    # 将触发总次数添加到记录列表中
    record.append(total)

    return record
```

---

## 5. 把Pandas想象成一个带有个性化功能的Excel；先熟悉它，然后深入思考。

```
import pandas as pd
import numpy as np

# 创建一个示例股票价格数据框
data = {'Date': pd.date_range(start='2022-01-01', periods=10,
freq='D'),
        'AAPL': [150, 151, 150, 153, 150, 155, 150, 157, 153,
150],
        'GOOG': [2800, 2810, 2820, 2830, 2840, 2850, 2860, 2870,
2880, 2890]}

stock_prices_df = pd.DataFrame(data)

# 创建一个示例交易量数据框
data = {'Date': pd.date_range(start='2022-01-01', periods=10,
freq='D'),
        'AAPL_volume': [1000, 1200, 1000, 1000, 1900, 1600,
1800, 1600, 1200, 1900],
        'GOOG_volume': [1000, 1100, 1200, 1050, 1150, 1250,
1300, 1400, 1500, 1450]}

volume_df = pd.DataFrame(data)

# 按'Date'列对数据框进行升序排序
stock_prices_df.sort_values(by='Date', inplace=True)

# 将stock_prices_df和volume_df根据'Date'列合并
merged_df = pd.merge(stock_prices_df, volume_df, on='Date')
print("\n合并后的数据框：")
print(merged_df)
```

groupby是一个很好用的函数。

```python
df_grouped = df.groupby('stock_code') # df_grouped 是一个groupby
对象
for stock_code, df_stock in df_grouped:
    print(stock_code)
    print(df_stock)
    break
```

# 按'AAPL'分组并计算'GOOG'的平均值

grouped_df = merged_df.groupby('AAPL')['GOOG'].mean() print("\n分组后的数字是所有Google价格的平均值：") print(grouped_df)

# 使用apply()应用自定义聚合函数

def custom_agg(x): return x.sum() / len(x)

# def custom_agg(x):

# return np.square(x.sum()) / len(x)

agg_result = merged_df.groupby('AAPL')['GOOG'].apply(custom_agg) print("\n使用apply()的自定义聚合：") print(agg_result)

# 将'AAPL'列向后移动1个周期

merged_df['AAPL_shifted'] = merged_df['AAPL'].shift(periods=1) print("\n移动后的'AAPL'列：") print(merged_df)

# 计算'AAPL'的滚动平均值，窗口大小为3

merged_df['AAPL_rolling_mean'] = merged_df['AAPL'].rolling(window=3).mean()

# 删除移动的'AAPL'列

merged_df.drop(columns=['AAPL_shifted'], inplace=True) print(merged_df)

小贴士：Google、Stack Overflow、复制粘贴，不（至少在完成3080课程前控制使用）chatgpt：）

---

## 3. 高级主题

- 我们在这里做的除了Excel功能还有什么?
- 这里的理念是什么?
- 因素分析?
- 市场时机?

让我给你讲一个故事。

# 很久很久以前，在一个遥远的银河系里...