Hw1

Q1

(a)

- Alice : 建仓所需的最少本金为 $18 \times 500 \times 50$
- Bill : 建仓所需的最少本金为 $18 \times 500 \times 50$

(b)

Balance Sheet	
Alice	
Assets	Liabilities
13500	9000
	Net Worth
	4500

Balance Sheet	
Alice	
Assets	Liabilities
9000	4500
	Net Worth
	4500

(c) - Alice 收到margin call的条件为:

$$\frac{13500 - 500P}{500P} = 40\% \Rightarrow P = 19.29$$

- Bill 收到margin call的条件为:

$$\frac{500P - 4500}{500P} = 40\% \Rightarrow P = 15$$

Q2

(a)

$$P = \sum_{t=1}^{T} \frac{100c}{(1+y)^t} + \frac{100}{(1+y)^T} \tag{1}$$

到期收益率y可利用以上方程解得

(b)

平价发行时$P = 100$,代入式(1)得到

$$1 = \sum_{t=1}^{T} \frac{c}{(1+y)^t} + \frac{1}{(1+y)^T} \tag{1}$$

解得,

$$y = c$$

(c)

平价发行时$P > 100$,代入式(1)得到

$$c > (1 - \frac{1}{(1+t)^T}) \left\{ \frac{1 - \frac{1}{1+y}}{\frac{1}{1+y} - \frac{1}{(1+y)^{T+1}}} \right\} = y$$

Q3

(a)

逐个接收OrderSize,我们发现:

$$98 + 3000 < 4000 < 98 + 3000 + 4000$$

因此yield to maturity的值为 1.10%

(b)

$$\begin{aligned} P &= \sum_{t=1}^{T} \frac{100 \times \text{coupon rate}}{(1+y)^T} + \frac{100}{(1+y)^T} \\ &= \sum_{t=1}^{5} \frac{100 \times 2\%}{(1+1.10\%)^5} + \frac{100}{(1+1.10\%)^5} \\ &= 104.36 \end{aligned}$$

国债发行的价格为$\$104.36$

(c)

该投资人每份国债的总收益为

$$100 + 5 \times 100 \times 2\% = 110$$

因此,年化回报率为:

$$\mathrm{AR} = (\frac{110}{100})^{1/5} - 1 = 1.92\%$$

Q4

(a)

Bid-ask spread为$50.25 - 49.75 = 0.5$,不可能小于0,因为一旦出现价格高于限价卖出订单的买入订单,交易就会瞬间完成.较小的价差意味着市场竞争剧烈

(b)

恰好有100份$\$50.25$的Limited Sell Orders,因此该订单总价为: $100 \times 50.25 = 5025$

(c)

注意到$100 < 150 < 100 + 300$,因此新订单总价为:

$$100 \times 51.50 + (150 - 100) \times 54.75 = 7887.5$$

Q5

(a)

经excel计算,三类资产实际收益率的均值和标准差(样本标准差)如下表

Real Return	Mean	SSD
US Stock Market	8.77%	0.161
Long Term Treasury	4.67%	0.148
Gold	3.61%	0.214

PS : 计算实际收益率的公式为 $\mathrm{r} = \frac{1+\text{名义收益率}}{1+\text{通胀率}} - 1$

(b)

- 美国年化通胀率的均值为3.57%,标准差为0.0280
- 上述三类资产的实际收益率与通胀率的相关系数分别为:$-0.0877, -0.452, 0.262$

(c)

如下表:

$1	Nominal Value	Real Value
US Stock Market	122.66	25.66
Long Term Treasury	23.79	4.98
Gold	10.23	2.14

实际价值可以视为实际收益的累乘,实际上也是名义收益与通胀累乘的比,两种计算方法的结果是相同的

Q6

(a)

In [ ]:
```
import pandas as pd

def calculate_monthly_returns(file_path):
    # □□□□□□□□□□□□□□□sheet□□□□□□□□
    monthly_data_by_sheet = {}

    # □□□□sheet□□□□□
    with pd.ExcelFile(file_path) as xls:
        for sheet_name in xls.sheet_names:
            # print(f"□□□□ {sheet_name} sheet...")
            # □□□□sheet□□□□□
            df = pd.read_excel(xls, sheet_name)
            # print("□□□□□□□□□:")
            # print(df.head())

            # □□□□□□□□□□□□□
            df['□□'] = pd.to_datetime(df['□□'])
            # print("□□□□□□□□□:")
            # print(df.head())

            # □□□□□□□□□□□
            start_date = pd.Timestamp('2018-12-01')
            end_date = pd.Timestamp('2024-02-29')  # □□□□□
            df = df[(df['□□'] >= start_date) & (df['□□'] <= end_date)]
            # print("□□□□□□□□□□□□:")
            # print(df.head())

            # □□□□□□□□□□□□□□□□□
            monthly_data = df.groupby(df['□□'].dt.to_period('M')).apply(lambda x: x[x['□□'] == x['□□'].max()])
            # print("□□□□□□□□:")
            # print(monthly_data.head())

            # □□□□□□□□□□
            monthly_data = monthly_data[['□□', '□□□□(□)']]
            monthly_data.set_index(monthly_data['□□'].dt.to_period('M'), inplace=True)  # □□□□□□□□
            monthly_data.drop(columns=['□□'], inplace=True)  # □□□□□□
            # print("□□□□□□□□□□□:")
            # print(monthly_data.head())

            # □□□□□□□
            monthly_data['□□□□'] = monthly_data['□□□□(□)'].shift(1)  # □□□□□□□□□
            monthly_data['□□□□□'] =
```

```
(monthly_data['单位净值(元)'] - monthly_data['上月□□']) / monthly_data['□□□□']
        # print("□□□□□□□□□□□□:")
        # print(monthly_data.head())

        # □□□□□□□
        monthly_data.drop(columns=['□□□□'], inplace=True)
        # print("□□□□□□□□□□□□:")
        # print(monthly_data.head())

        # □□□sheet□□□□□□□□□□□□□□
        monthly_data_by_sheet[sheet_name] = monthly_data['□□□□□']

    return monthly_data_by_sheet

# □□□□□□□□□□□□
def calculate_covariance_matrix(file_path):
    # □□□□□□□□□□□
    monthly_returns_by_sheet = calculate_monthly_returns(file_path)

    # □□□DataFrame
    df = pd.DataFrame(monthly_returns_by_sheet)
    print(df)

    # □□□□□□□□□□□□□□□□□
    covariance_matrix = df.corr()

    return covariance_matrix

# □□□□□□□□□
file_path = '□□□□□□.xlsx'
monthly_returns_by_sheet = calculate_monthly_returns(file_path)
# □□□□□□□□□□□□□□
for sheet_name, monthly_returns in monthly_returns_by_sheet.items():
    print(f"□□□□□: {sheet_name}")
    print("□□□□□□□:", monthly_returns.mean())
    print("□□□□□□□□:", monthly_returns.std())
    print()

covariance_matrix = calculate_covariance_matrix(file_path)
print("□□□□□□□□□□□□:")
print(covariance_matrix)
```

```
□□□□□: 005960.OF
□□□□□□□: 0.005558680328667218
□□□□□□□□: 0.04483560117613314

□□□□□: 000008.OF
□□□□□□□: 0.006594566740407141
□□□□□□□□: 0.05657293402371166

        005960.OF  000008.OF
□□
```

```
2018-12       NaN       NaN
2019-01  0.036307  0.000883
2019-02  0.116005  0.186574
2019-03  0.081224  0.100513
2019-04 -0.026454 -0.040803
...           ...       ...
2023-10 -0.041242 -0.027211
2023-11 -0.007169  0.002753
2023-12 -0.021218 -0.019845
2024-01 -0.028782 -0.128104
2024-02  0.079996  0.130057

[63 rows x 2 columns]
□□□□□□□□□□□□□:
           005960.OF  000008.OF
005960.OF   1.000000   0.848266
000008.OF   0.848266   1.000000
```

从计算结果中发现,嘉实 500 中证 ETF 联接 A的收益和风险均显著高于博时量化价值A,两只基金收益相关性较强

(b)

中航首钢绿能 REIT【SZ180801】是封闭式基金,原因有:

- 网站明确说明了是封闭式基金
- 申购状态为不可申购
- 固定投资组合
- 折溢价率为正

注意到:

- 市场交易价格为 13.6690 元
- 单位净值为 10.1308 元 将这些数据代入计算公式，即可得到折溢价率:

$$\text{折溢价率} = \frac{P - NAV}{NAV} \times 100\% = 34.92\%$$

根据以上计算过程,封闭式基金折价是指基金的市场交易价格低于其单位净值的情况。

(c)

当基金份额价格为19元时,

$$NAV = 2000/100 = 20$$

由于市场价格低于NAV，这意味着份额被低估了。对冲策略管理人可以采取以下步骤:

- 以市场价格买入份额
- 等待市场价格上涨到接近NAV时卖出份额

当基金份额价格为22元时,管理人可以选择卖出份额
当基金份额价格为20元时,份额的价格正确反映了基金的净资产值,因此没有套利机会