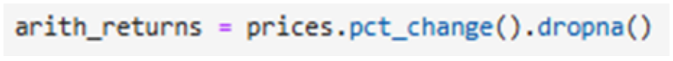
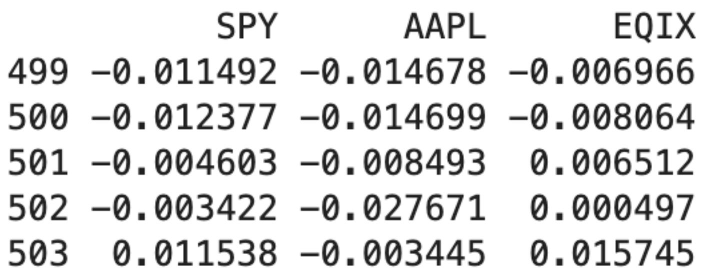
1.

Problem A Answer: for the question A, it is asked to calculate the arithmetic returns. The data we have is all stock price at the end of the day. So, in order to get the return we need to compute the return between price t and price t-1. In this case, I use the code:



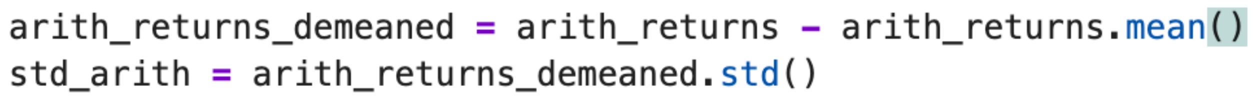
```
arith_returns = prices.pct_change().dropna()
```

This code help me computes change between a price on Pt and the previous Pt-1. And the last five row of the means are:



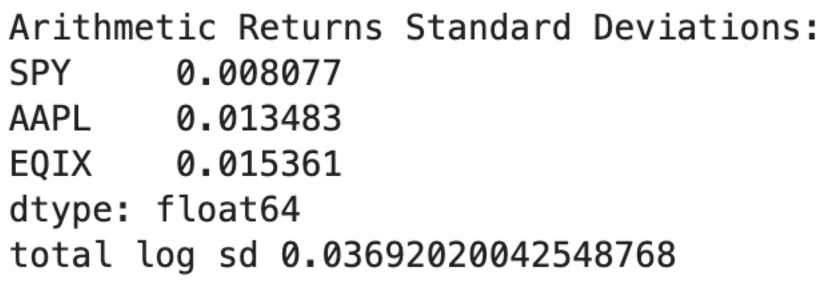
```
          SPY      AAPL      EQIX
499 -0.011492 -0.014678 -0.006966
500 -0.012377 -0.014699 -0.008064
501 -0.004603 -0.008493  0.006512
502 -0.003422 -0.027671  0.000497
503  0.011538 -0.003445  0.015745
```

Then I remove the mean from each series so that the mean becomes zero, and Compute the standard deviation for each stock's demeaned arithmetic returns.



```
arith_returns_demeaned = arith_returns - arith_returns.mean()
std_arith = arith_returns_demeaned.std()
```

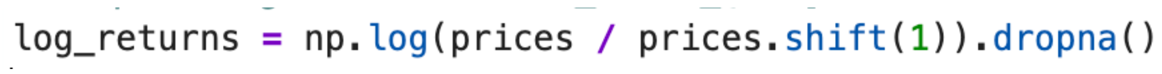
The result I get is:



```
Arithmetic Returns Standard Deviations:
SPY     0.008077
AAPL    0.013483
EQIX    0.015361
dtype: float64
total log sd 0.03692020042548768
```

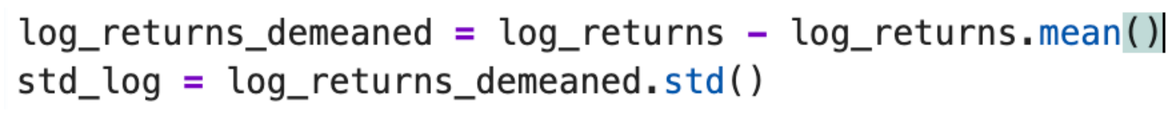
Problem B:

Same for the log return. I need to get the return from price t and price t-1. But with np.log function.



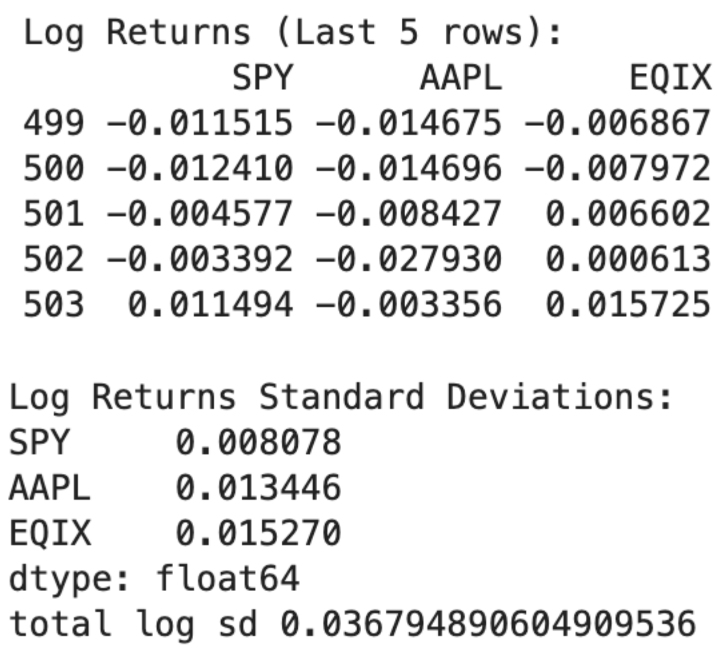
```
log_returns = np.log(prices / prices.shift(1)).dropna()
```

After this I remove the mean from each series and compute the standard deviation for each stock's demeaned log returns.



```
log_returns_demeaned = log_returns - log_returns.mean()
std_log = log_returns_demeaned.std()
```

The output I got is:



```
Log Returns (Last 5 rows):
          SPY      AAPL      EQIX
499 -0.011515 -0.014675 -0.006867
500 -0.012410 -0.014696 -0.007972
501 -0.004577 -0.008427  0.006602
502 -0.003392 -0.027930  0.000613
503  0.011494 -0.003356  0.015725

Log Returns Standard Deviations:
SPY     0.008078
AAPL    0.013446
EQIX    0.015270
dtype: float64
total log sd 0.036794890604909536
```

2.

Question A: To calculate the current value of the portfolio given the date: 1/3/2025. I define the date and target price. I find the row with the given date and calculate the portfolio value by multiplying shares by prices.

And the output I got is: Portfolio Value on 2025-01-03 = $251,862.50

Question B.a:
first we need to get the return: which follows the formula rt = pt/(pt-1) -1. And then subtract the mean return to ensure zero mean assumption.

$$R_t = \frac{P_t}{P_{t-1}} - 1 \quad R'_t = R_t - \bar{R}$$

To calculate var, we need to follow this function:

$$VaR(\alpha) = - PV * F_X^{-1}(\alpha) * \sqrt{\nabla R^T \Sigma \nabla R}$$

We have the PV and it is quite easy to get the critical quantile Fx-1(a). And we need to get the portfolio's standard deviation. $\sqrt{\nabla R^T \Sigma \nabla R}$ To get this standard deviation, we need to get the exponentially weighted covariance matrix, I use the following code to calculate the matrix. This is something we did last project. Code:

```
def calc_exp_cov(data, decay=0.97):
    weights = np.array([decay ** i for i in range(len(data))])[::-1]
    weights /= weights.sum()
    avg = np.average(data, axis=0, weights=weights)
    diff = data - avg
    cov_matrix = np.dot(weights * diff.T, diff)
    return pd.DataFrame(cov_matrix, index=data.columns, columns=data.columns)
```

Math formula:

$$w_{t-i} = (1 - \lambda)\lambda^{i-1}$$

$$\widehat{w_{t-i}} = \frac{w_{t-i}}{\sum_{j=1}^{n} w_{t-j}}$$

$$\widehat{\sigma_t^2} = \sum_{=1}^{n} w_{t-i}\left(x_{t-i} - \bar{x}\right)^2$$

$$\widehat{cov(x,y)} = \sum_{i=1}^{n} w_{t-i}\left(x_{t-i} - \bar{x}\right)\left(y_{t-i} - \bar{y}\right)$$

After we calculate the exp_cov, we can get the sd of the portfolio with my following code:

```
assets = np.array([positions[s] * curr_prices[s] for s in stocks])
dollar_w = assets / port_value
p_var = dollar_w @ exp_cov @ dollar_w
p_std = np.sqrt(p_var)
```

First I converting asset position to dollar weights, and get the variance with this equation:

$$\sigma^2_{\text{portfolio}} = \mathbf{w}^\top \Sigma \mathbf{w}$$

Eventually get the standard deviation since it is square root of the variance.

After that to I find the 5th percentile of the standard normal distribution by finding the z score and the pdf

```
z_score = norm.ppf(0.05)
pdf_val = norm.pdf(z_score)
```

Combining everything following the previous formula.

$$VaR(\alpha) =- PV * F_X^{-1}(\alpha) * \sqrt{\nabla R^T \Sigma \nabla R}$$

```
var_norm = -z_score * p_std * port_value
es_norm = p_std * (pdf_val / 0.05) * port_value
```

I got the output:

```
Normal VaR: 3856.3183014814767
Normal ES: 4835.978727805546
```

Interpretation:

VaR (3856.32): This represents the potential maximum loss at a given confidence level (typically 95% or 99%) over a specific time horizon. It means there is a 5% chance that losses exceed this amount.

ES (4835.98): Also called Conditional VaR (CVaR), ES represents the expected loss in the worst-case 5% scenarios, providing a more comprehensive risk measure than VaR since it accounts for tail risk.

Question B.b:

First, get the portfolio value and remove the mean. This is same with the last part of the question.

$$R_t = \frac{P_t}{P_{t-1}} - 1 \quad R'_t = R_t - \bar{R}$$

Second, fit a Maximum Likelihood Estimation (MLE) to fit a Student's T distribution to the demeaned returns.

Third, transforming the return to uniform then to normal quantiles. For each asset, Convert each demeaned return r(0) into a cumulative probability u u using the T CDF.

$$u = F_{T,\nu,\mu,s}(r^{(0)})$$

And converting uniform variables to standard normal quantiles with transformation.

$$z = \Phi^{-1}(u)$$

```python
norm_q = U.apply(lambda col: norm.ppf(col))
spearman_corr = norm_q.corr(method="spearman")
```

Fourth, simulating joint returns via Gaussian Copula. This begin with simulate a multivariate normal distribution.

$$\mathbf{z}^{(sim)} \sim \mathcal{N}\left(\mathbf{0}, \Sigma_{\text{spearman}}\right)$$

Then convert simulated normal back to t-distributed return with following math equation:

$$u^{(sim)} = \Phi(z^{(sim)}) \quad r^{(sim)} = F^{-1}_{T,\nu,\mu,s}(u^{(sim)})$$

This maps the simulated normal variables back into the space of returns that follow the fitted T distribution.

Fifth, simulate the PnL simulation for each asset and add up to the total portfolio PnL. This follows the math equation:

$$\text{PnL}_i^{(sim)} = \text{Position}_i \times \text{Price}_{i,\text{last}} \times r_i^{(sim)} \quad \text{PnL}^{(sim)} = \sum_i \text{PnL}_i^{(sim)}$$

The **last** step will be calculate Var and ES,

$$\text{VaR} = \text{Percentile}_{5\%}(\text{PnL}^{(sim)}) \quad \text{ES} = \mathbb{E}[\text{PnL}^{(sim)} \mid \text{PnL}^{(sim)} \leq \text{VaR}]$$

Eventually I got the output:

```
Portfolio Value: $251,862.50
5% VaR = $4,369.76
5% ES  = $6,074.73
```

Interpretation: This means that with 95% confidence, the portfolio is not expected to lose more than $4,390.50, and if losses do exceed this threshold, the average loss is about $6,139.69.

Question B.c:

First, get the portfolio value and remove the mean. This is same with the last part of the question.

$$R_t = \frac{P_t}{P_{t-1}} - 1 \quad R_t' = R_t - \bar{R}$$

Then I need to compute the historical portfolio PnL for each time t:

$$\text{PnL}_t = \sum_{i=1}^{N} (r_{i,t} \times \text{Exposure}_i)$$

Eventually calculate the historical VAR and Expected Shortfall with the formula we use before:

$$\text{VaR} = \text{Percentile}_{5\%}(\text{PnL}^{(sim)}) \quad \text{ES} = \mathbb{E}[\text{PnL}^{(sim)} \mid \text{PnL}^{(sim)} \leq \text{VaR}]$$

Eventually I got the output:

```
Historical 5% VaR: $4,575.03
Historical 5% ES: $6,059.39
```

The Var means that, based on past data, there is a 5% chance the portfolio will lose at least $4,575.03 in a given time period. The ES presents the expected loss in the worst 5% of cases, meaning that if losses exceed the VaR threshold, the average loss would be $6,059.39

Question C:

All three methods measure the 5% VaR and ES, but they do so in different ways. The Normal Distribution with Exponentially Weighted Covariance (EWMA) approach assumes returns follow a normal pattern and puts more weight on recent data, making it quick to compute. However, it often underestimates extreme losses because it does not account for the heavier tails that sometimes occur in real markets. This underestimation is clear from its lower 5% VaR of $3,856.32 and ES of $4,835.98. In contrast, the t-Distribution with a Gaussian Copula uses a fat-tailed distribution and captures nonlinear dependencies between assets, giving a more realistic view of tail risk but at a higher computational cost. This method produced a 5% VaR of $4,369.76and the highest ES of $ 6,074.73, reflecting its strong focus on extreme outcomes. Finally, the Historical Simulation method calculates VaR and ES directly from actual past returns, so it is straightforward and shows real market behavior. Its estimates (5% VaR of $4,575.03 and ES of $6,059.39) are close to the t-Copula results, indicating similar recognition of significant loss events. However, this method can fail if the future does not resemble the past. In summary, EWMA is fast but underestimates tail risk, the t-Copula method better captures extreme scenarios at a higher computational cost, and Historical Simulation is highly intuitive but heavily reliant on historical data.

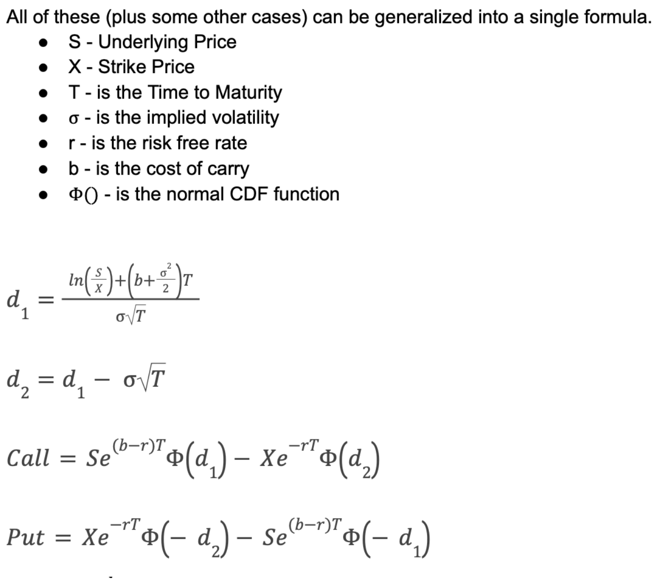
Question 3.

Question A: in this question it asked for implied volatility. Because the market implied volatility is not directly observed, we need to use option prices, which we can observed, to value the overall market implied volatility. So the first step is to value the present value of the option. Since this is an European option, it can only exercise at the end, I use Black-Scholes Formulas instead of a binary tree model. I define the bs_call and bs_put function.

```python
def bs_call(spot, strike, time, rate, vol):
    d1 = (log(spot / strike) + (rate + 0.5 * vol**2) * time) / (vol * sqrt(time))
    d2 = d1 - vol * sqrt(time)
    return spot * norm.cdf(d1) - strike * exp(-rate * time) * norm.cdf(d2)
```

```python
def bs_put(spot, strike, time, rate, vol):
    d1 = (log(spot / strike) + (rate + 0.5 * vol**2) * time) / (vol * sqrt(time))
    d2 = d1 - vol * sqrt(time)
    return strike * exp(-rate * time) * norm.cdf(-d2) - spot * norm.cdf(-d1)
```

The code above follows the following formula:



All of these (plus some other cases) can be generalized into a single formula.

- S - Underlying Price
- X - Strike Price
- T - is the Time to Maturity
- σ - is the implied volatility
- r - is the risk free rate
- b - is the cost of carry
- Φ() - is the normal CDF function

$$d_1 = \frac{ln\left(\frac{S}{X}\right)+\left(b+\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

$$Call = Se^{(b-r)T}\Phi\left(d_1\right) - Xe^{-rT}\Phi\left(d_2\right)$$

$$Put = Xe^{-rT}\Phi\left(-d_2\right) - Se^{(b-r)T}\Phi\left(-d_1\right)$$

But without the e^(b-r)T, because this is for stock that paid for dividend.

After define the bs call and bs put function, I calculate the implied volatility with this function:

```python
def iv_call(spot, strike, time, rate, market):
    f = lambda vol: bs_call(spot, strike, time, rate, vol) - market
    return brentq(f, 1e-6, 5.0)
```

brentq(f, 1e-6, 5.0) is a numerical solver that finds the root of f(vol) = 0, i.e., it finds the vol where the volatility(vol) that can explain the observed option price. With this calculation I got the output as following: IV = 33.51%

Question B: in this part I calculate for the Delta, Vega, and Theta for call option.

- Delta:
    - For Delta call, the formula I use is $\Delta_{\text{call}} = \Phi(d_1)$, the reason I don't use the formula that is shown in the pdf (
        - Call: $e^{(b-r)T}\Phi(d_1)$

) is because the second formula is for dividend paying stock, however, this company does not pay for dividend. In this case, the first formula will fit this scenario. The code I apply is following:

```python
def delta_call(spot, strike, time, rate, vol):
    d1 = (log(spot / strike) + (rate + 0.5 * vol**2) * time) / (vol * sqrt(time))
    return norm.cdf(d1)
```

- This get the d1 first and from d1 get the delta. The result for Delta call is 0.6659
- For Delta put, it is similar the value will be $\Phi(d_1) - 1$ result the delta put = -0.3341

- Vega:
  - For Vega, it is same as Delta. The formula in pdf does not work because it fits better with the dividend paying stock. The formula I use is $\nu = S_0\phi(d_1)\sqrt{T}$. The code I apply with this is:

```
def vega_call(spot, strike, time, rate, vol):
    d1 = (log(spot / strike) + (rate + 0.5 * vol**2) * time) / (vol * sqrt(time))
    return spot * norm.pdf(d1) * sqrt(time)
```

  Get the d1 first and times it with spot and sqrt of time. Result: vega =5.6407

- Theta:
  - For Theta call, it get the d1 and d2 first. The formula for d1 and d2 are same as above: $d_1 = \frac{ln\left(\frac{S}{X}\right)+\left(b+\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$ $d_2 = d_1 - \sigma\sqrt{T}$. And the formula use is:

$$: -\frac{S_0\phi(d_1)\sigma}{2\sqrt{T}} - rKe^{-rT}N(d_2)$$

Code I use:

```
def theta_call(spot, strike, time, rate, vol):
    d1 = (log(spot / strike) + (rate + 0.5 * vol**2) * time) / (vol * sqrt(time))
    d2 = d1 - vol * sqrt(time)
    return - (spot * norm.pdf(d1) * vol) / (2 * sqrt(time)) - rate * strike * exp(-rate * time) * norm.cdf(d2)
```

  Get the d1 and d2, and follow the above formula. The Theta call = -5.5446.

  - For put, it follows $$\Theta_{\text{put}} = -\frac{S \cdot \phi(d_1) \cdot \sigma}{2\sqrt{T}} + r \cdot X \cdot e^{-rT} \cdot \Phi(-d_2)$$
  - And the result I get is -2.6186

```
Call Delta = 0.6659
Put Delta = -0.3341
Call Vega  = 5.6407
Call Theta = -5.5446
Put Theta = -2.6186
```

-

- Delta call : ~0.6659. Interpretation: For a $1 increase in the stock price, we expect the call option's price to increase by $0.6659, approximately. For Delta put, a $1 increase in the stock price, we expect the put option's price to decerase by $0.3341
- Vega: ~5.6407. Interpretation: For each 1 percentage point increase in implied volatility, we expect the call option's price to go up by about $0.0564. Equivalently, for a 1% (0.01) change in σ, the call changes by $5.64 * 0.01 = 0.0564$.
- Theta call: ~-5.5446. Interpretation: This is the annual rate of change with respect to time. Practitioners often divide by 365, so the daily time decay would be. -5.5446/365 = −0.0152 per day. In other words, each passing day (in calendar days, if we use 365) costs about 1.5 cents in the call option's value purely from time decay, ignoring changes in the underlying or volatility. For theta put it is the same but -2.6186/365 = -0.007174 per day.

```
Actual Price Change = 0.0565
Vega Approx Change = 0.0564
```

- I take the implied vol iv (about 0.3351) and increase it by 0.01. I compute the new call price (price1) vs. the old call price (price0) and observe the actual difference.
- I compare this difference with what Vega predicted. When I raise implied volatility by 1 percentage point, the call's price increase 0.0565. Vega predicted an increase of $0.0564, which is extremely close—demonstrating that vega is an excellent local (small change) approximation for the effect of volatility on the option price.

Question C:

- First I calculate the put price with the function I define earlier and I got the price:
-
```
Put Price = 1.2593
```
- For put call parity, I follow the formula: $C + Xe^{-rT} = P + S$. the code I use is:

```
put_val = bs_put(spot, strike, maturity, rate, iv)
lhs = mkt_call + strike * exp(-rate * maturity)
rhs = put_val + spot
```

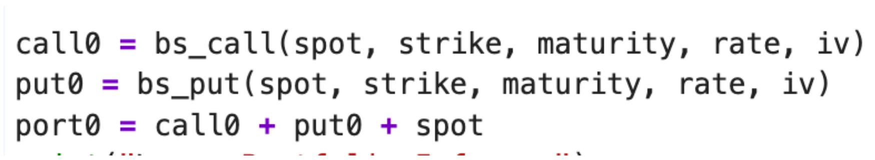
- Then the result I got from it is

```
Put Price = 1.2593
LHS (Call + PV) = 32.2593
RHS (Put + Spot)= 32.2593
Difference      = 0.00000000
```

The difference is effectively zero, so put–call parity holds.

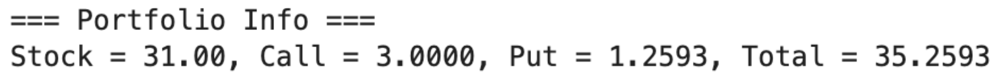
Question D:

- First I use the call and put function that I defined before to calculate the call price and put price. And add them up together.



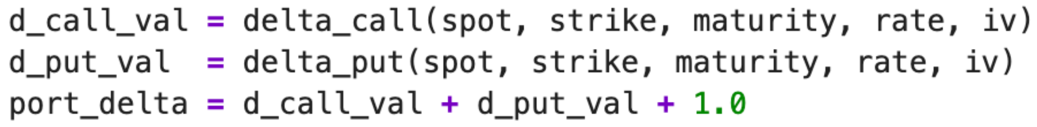
```
call0 = bs_call(spot, strike, maturity, rate, iv)
put0 = bs_put(spot, strike, maturity, rate, iv)
port0 = call0 + put0 + spot
```

-
- Here is the output:



```
=== Portfolio Info ===
Stock = 31.00, Call = 3.0000, Put = 1.2593, Total = 35.2593
```

-
- Second, compute the daily volatility: $\sigma_{\text{daily}} = \frac{0.25}{\sqrt{255}} \quad \sigma_{20d} = \sigma_{\text{daily}} \times \sqrt{20}$
- Then I calculate the delta for the entire portfolio. The total delta will be the delta from call + delta from put + delta for stock price. For call and put delta, I use the function I define before, and delta for stock price will be 1. Since the stock has a delta of 1 (a $1 change in stock price changes its value by $1), the total portfolio delta is:



```
d_call_val = delta_call(spot, strike, maturity, rate, iv)
d_put_val  = delta_put(spot, strike, maturity, rate, iv)
port_delta = d_call_val + d_put_val + 1.0
```

-
The result I got is delta = 1.3319
- For Theta, I calculated the theta for call and put option and add them together. The result I get: Theta = -8.1632
- To calculate the VAR for Delta-Normal Approximation. Estimating the Profit & Loss will be the first step. The mean_pnl = theta_port * dt. The standard deviation of the pnl formula will be std_PNL = |delta port| * stock price * vol_20d. Then I can compute the VAR. I first calculate the z value with z = norm.ppf(0.05).

$$\alpha = 0.05 \Rightarrow VaR_{ret} \approx 1.65\sigma_P$$

-
- Then use the code: var_dn = -(mean_pnl + z * std_pnl) to calculate the value at risk:

$$\text{VaR}_{DN} = -\left(\text{mean\_PnL} + z \cdot \text{std\_PnL}\right)$$

-

- In this formula mean_pnl follows this formula:

$$\text{mean\_PnL} = \Theta_{\text{port}} \times dt$$

- And std_PNL follows:

$$\text{std\_PnL} = |\Delta_{\text{port}}| \times \text{Stock Price} \times \sigma_{20d}$$

- The result I get: Mean PnL = -0.6403, Std PnL = 2.8907 and VaR (95%) = 5.3951
- The Expected shortfall follows this formula:

$$ES_{\alpha}(X) = -\mu + \sigma \frac{f\left(\Phi^{-1}(\alpha)\right)}{\alpha}$$

-
- And I get: ES (95%) = 6.6030
- Total output:

```
Delta = 1.3319, Theta = -8.1632
Mean PnL = -0.6403, Std PnL = 2.8907
VaR (95%) = 5.3951, ES (95%) = 6.6030
```

-
- Interpretation:
- The Delta-Normal Approximation suggests this position has a Delta of 1.3319, meaning its value changes by about 1.33 units for each 1-unit move in the underlying. The Theta of -8.1632 indicates the position loses around 8.16 units of value each day as time passes. The average PnL is -0.6403, meaning on average it slightly loses money, and the standard deviation of 2.8907 shows the daily profits and losses can vary by about 2.89 units. The 95% VaR of 5.3951 means there is only a 5% chance losses will be higher than 5.40 units, while the 95% ES of 6.6030 tells us that in those worst 5% cases, the average loss is around 6.60 units.

Question 3 d.e: To complete a monte carlo simulation, I generates 100,000 random draws from a normal distribution with mean 0 and standard deviation equal to the 20-day volatility. Then I simulate stock prices with the following formula:

$$S_{\text{sim}} = S_0 \times \exp(r)$$ in the formula: $$r \sim \mathcal{N}(0, \sigma_{20d}^2)$$

And I then I adjust the time to maturity. Tnew = t-20/255. After that I re price the option with the following code:

```
call_sim = np.array([bs_call(s, strike, time_new, rate, iv) for s in spot_sim])
put_sim  = np.array([bs_put(s, strike, time_new, rate, iv) for s in spot_sim])
```

This code simulated stock price, the call and put are re-priced. Then I calculate the portfolio value and PnL with the following formula:

$$P_{\text{sim}} = \text{Call Price}_{\text{sim}} + \text{Put Price}_{\text{sim}} + S_{\text{sim}}$$

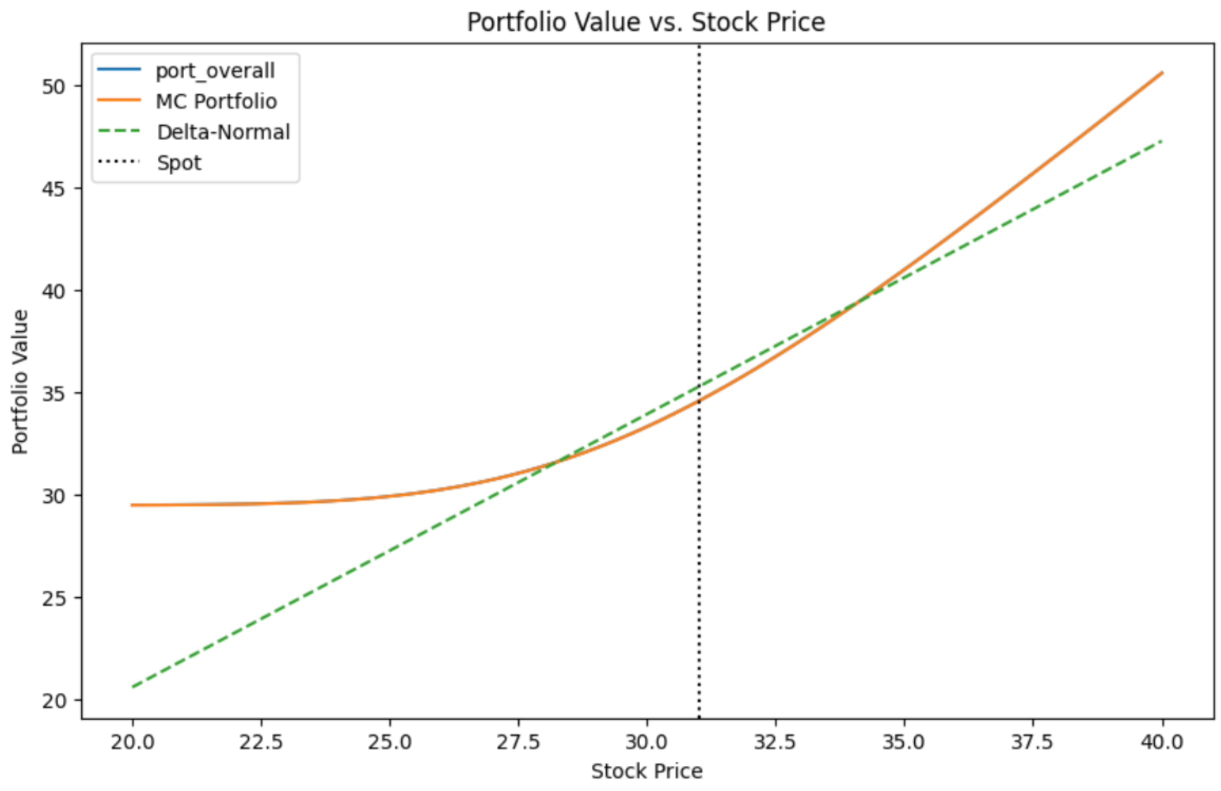
$$\text{PnL} = P_{\text{sim}} - P_0$$

Eventually get the var and ES just like before and got the result:

```
Sims = 100000, Mean PnL = -0.2083
VaR (95%) = 4.1383, ES (95%) = 4.5823
```

Interpretation: This result comes from running 100,000 Monte Carlo simulations of possible profit and loss outcomes. On average, the position loses 0.2083 units (Mean PnL = -0.2083). The 95% VaR of 4.1383 means there's only a 5% chance that losses will be bigger than 4.1383 units. Finally, the 95% ES of 4.5823 shows that, in those worst 5% scenarios, the average loss is about 4.58 units.

Question E:



The Delta-Normal approximation and Monte Carlo simulation differ significantly in how they assess portfolio risk. The Delta-Normal method is simple to compute and works best for small price changes, but it assumes a linear relationship and ignores second-order effects like Gamma. Because of this, it can become highly inaccurate when stock prices move significantly. As shown in the figure, options do not behave linearly, so relying on Delta-Normal for option-heavy portfolios introduces a large bias.

In contrast, Monte Carlo simulation generates many price paths to capture the non-linear characteristics of the portfolio, which keeps its estimated price changes close to the true values. However, it requires more computation time. It take longer time to process it.

Overall, Monte Carlo is more accurate for portfolios with non-linear exposures, while the Delta-Normal method is easier to compute but less reliable for portfolios containing options.