In [ ]:
```
import pandas as pd
import numpy as np
import yfinance as yf
from forex_python.converter import CurrencyRates
import datetime as dt
from scipy import stats
```

In [ ]:
```
today
```

Out[ ]:
```
datetime.datetime(2024, 2, 16, 14, 21, 9, 230358)
```

In [ ]:
```
# get stock objects
tencent_hk = yf.Ticker('0700.HK')
rockstar_us = yf.Ticker('TTWO')
nintendo_jp = yf.Ticker('7974.T')

# get currency for today

today = dt.datetime(2024, 2, 16, 9, 21, 9, 230358)
c = CurrencyRates(today)

# get indexes
hk_index = yf.Ticker('^HSI')
us_index = yf.Ticker('^GSPC')
jp_index = yf.Ticker('^N225')
```

In [ ]:
```
# get 1 year data
hist_tencent_hk = tencent_hk.history(period='1y')
hist_tencent_hk.reset_index(inplace=True)
hist_tencent_hk = hist_tencent_hk[['Date', 'Close']].copy(deep=True)
hist_tencent_hk.rename(columns=str.lower, inplace=True)
hist_tencent_hk['date'] = pd.to_datetime(hist_tencent_hk['date']).dt.date

hist_rockstar_us = rockstar_us.history(period='1y')
hist_rockstar_us.reset_index(inplace=True)
hist_rockstar_us = hist_rockstar_us[['Date', 'Close']].copy(deep=True)
hist_rockstar_us.rename(columns=str.lower, inplace=True)
hist_rockstar_us['date'] = pd.to_datetime(hist_rockstar_us['date']).dt.date

hist_nintendo_jp = nintendo_jp.history(period='1y')
hist_nintendo_jp.reset_index(inplace=True)
hist_nintendo_jp = hist_nintendo_jp[['Date', 'Close']].copy(deep=True)
```

```
hist_nintendo_jp.rename(columns=str.lower, inplace=True)
hist_nintendo_jp['date'] = pd.to_datetime(hist_nintendo_jp['date']).dt.date

# 1 yr index data
hist_hk_index = hk_index.history(period='1y')
hist_hk_index.reset_index(inplace=True)
hist_hk_index = hist_hk_index[['Date', 'Close']].copy(deep=True)
hist_hk_index.columns = hist_hk_index.columns.str.lower()
hist_hk_index['date'] = pd.to_datetime(hist_hk_index['date']).dt.date

hist_us_index = us_index.history(period='1y')
hist_us_index.reset_index(inplace=True)
hist_us_index = hist_us_index[['Date', 'Close']].copy(deep=True)
hist_us_index.columns = hist_us_index.columns.str.lower()
hist_us_index['date'] = pd.to_datetime(hist_us_index['date']).dt.date

hist_jp_index = jp_index.history(period='1y')
hist_jp_index.reset_index(inplace=True)
hist_jp_index = hist_jp_index[['Date', 'Close']].copy(deep=True)
hist_jp_index.columns = hist_jp_index.columns.str.lower()
hist_jp_index['date'] = pd.to_datetime(hist_jp_index['date']).dt.date
```

```
c:\Users\tkkim\AppData\Local\Programs\Python\Python312\Lib\site-packages\yfinance\utils.py:775: FutureWarning: The 'uni
t' keyword in TimedeltaIndex construction is deprecated and will be removed in a future version. Use pd.to_timedelta in
stead.
  df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')
c:\Users\tkkim\AppData\Local\Programs\Python\Python312\Lib\site-packages\yfinance\utils.py:775: FutureWarning: The 'uni
t' keyword in TimedeltaIndex construction is deprecated and will be removed in a future version. Use pd.to_timedelta in
stead.
  df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')
c:\Users\tkkim\AppData\Local\Programs\Python\Python312\Lib\site-packages\yfinance\utils.py:775: FutureWarning: The 'uni
t' keyword in TimedeltaIndex construction is deprecated and will be removed in a future version. Use pd.to_timedelta in
stead.
  df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')
c:\Users\tkkim\AppData\Local\Programs\Python\Python312\Lib\site-packages\yfinance\utils.py:775: FutureWarning: The 'uni
t' keyword in TimedeltaIndex construction is deprecated and will be removed in a future version. Use pd.to_timedelta in
stead.
  df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')
c:\Users\tkkim\AppData\Local\Programs\Python\Python312\Lib\site-packages\yfinance\utils.py:775: FutureWarning: The 'uni
t' keyword in TimedeltaIndex construction is deprecated and will be removed in a future version. Use pd.to_timedelta in
stead.
  df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')
c:\Users\tkkim\AppData\Local\Programs\Python\Python312\Lib\site-packages\yfinance\utils.py:775: FutureWarning: The 'uni
t' keyword in TimedeltaIndex construction is deprecated and will be removed in a future version. Use pd.to_timedelta in
stead.
  df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')
```

In [ ]:

```
"""hkd_to_usd = c.get_rate('HKD', 'USD', today)
jpy_to_usd = c.get_rate('JPY', 'USD', today)
usd_to_hkd = c.get_rate('USD', 'HKD', today)
usd_to_jpy = c.get_rate('USD', 'JPY', today)"""


#hardcoded as api is dead
hkd_to_usd = .1278
jpy_to_usd = 0.006657
usd_to_hkd = 7.8221
usd_to_jpy = 150.22
```

In [ ]:

```
usd_to_jpy
```

Out[ ]:

```
150.22
```

In [ ]:

```
# common dates

datelist = []
```

```
for date in hist_tencent_hk['date']:
    if date in hist_rockstar_us['date'].values and date in hist_nintendo_jp['date'].values and date in hist_hk_index['d
        datelist.append(date)
```

In [ ]:
```
# semi-accurate way to normalize dates
hist_nintendo_jp = hist_nintendo_jp[hist_nintendo_jp['date'].isin(datelist)]
hist_rockstar_us = hist_rockstar_us[hist_rockstar_us['date'].isin(datelist)]
hist_tencent_hk = hist_tencent_hk[hist_tencent_hk['date'].isin(datelist)]
hist_hk_index = hist_hk_index[hist_hk_index['date'].isin(datelist)]
hist_us_index = hist_us_index[hist_us_index['date'].isin(datelist)]
hist_jp_index = hist_jp_index[hist_jp_index['date'].isin(datelist)]
```

In [ ]:
```
# normalize stock prices
hist_nintendo_jp['close'] = hist_nintendo_jp['close'] * jpy_to_usd
hist_tencent_hk['close'] = hist_tencent_hk['close'] * hkd_to_usd

# normalize index
hist_hk_index['close'] = hist_hk_index['close'] * hkd_to_usd
hist_jp_index['close'] = hist_jp_index['close'] * jpy_to_usd
```

In [ ]:
```
hist_nintendo_jp['return_log'] = np.log(hist_nintendo_jp['close']) - np.log(hist_nintendo_jp['close'].shift(1)).dropna(
hist_rockstar_us['return_log'] = np.log(hist_rockstar_us['close']) - np.log(hist_rockstar_us['close'].shift(1)).dropna(
hist_tencent_hk['return_log'] = np.log(hist_tencent_hk['close']) - np.log(hist_tencent_hk['close'].shift(1)).dropna()

hist_nintendo_jp['return'] = hist_nintendo_jp['close'] / hist_nintendo_jp['close'].shift(1) - 1
hist_rockstar_us['return'] = hist_rockstar_us['close'] / hist_rockstar_us['close'].shift(1) - 1
hist_tencent_hk['return'] = hist_tencent_hk['close'] / hist_tencent_hk['close'].shift(1) - 1


hist_jp_index['return_log'] = np.log(hist_jp_index['close']) - np.log(hist_jp_index['close'].shift(1)).dropna()
hist_us_index['return_log'] = np.log(hist_us_index['close']) - np.log(hist_us_index['close'].shift(1)).dropna()
hist_hk_index['return_log'] = np.log(hist_hk_index['close']) - np.log(hist_hk_index['close'].shift(1)).dropna()


hist_jp_index['return'] = hist_jp_index['close'] / hist_jp_index['close'].shift(1) - 1
hist_us_index['return'] = hist_us_index['close'] / hist_us_index['close'].shift(1) - 1
hist_hk_index['return'] = hist_hk_index['close'] / hist_hk_index['close'].shift(1) - 1
```

In [ ]:
```
hist_nintendo_jp.dropna(inplace=True)
hist_nintendo_jp.reset_index(drop=True, inplace=True)

hist_rockstar_us.dropna(inplace=True)
hist_rockstar_us.reset_index(drop=True, inplace=True)
```

```
hist_tencent_hk.dropna(inplace=True)
hist_tencent_hk.reset_index(drop=True, inplace=True)

hist_hk_index.dropna(inplace=True)
hist_hk_index.reset_index(drop=True, inplace=True)

hist_us_index.dropna(inplace=True)
hist_us_index.reset_index(drop=True, inplace=True)

hist_jp_index.dropna(inplace=True)
hist_jp_index.reset_index(drop=True, inplace=True)
```

In [ ]:
```
n_invest = 10000000/3
r_invest = 10000000/3
t_invest = 10000000/3
```

In [ ]:
```
portfolio = hist_nintendo_jp[['date']].copy(deep=True)
```

In [ ]:
```
portfolio['returns'] = hist_nintendo_jp['return'] + hist_rockstar_us['return'] + hist_tencent_hk['return']
```

In [ ]:
```
portfolio_mean = portfolio['returns'].mean()
protfolio_std = portfolio['returns'].std(ddof=1)
```

Expected Returns

In [ ]:
```
n_er = np.exp(hist_nintendo_jp['return_log'].mean()) * ((n_invest*usd_to_jpy)/hist_nintendo_jp.loc[0,'close']) * jpy_to
r_er = np.exp(hist_rockstar_us['return_log'].mean()) * ((r_invest)/hist_rockstar_us.loc[0,'close'])
t_er = np.exp(hist_tencent_hk['return_log'].mean()) * ((t_invest*usd_to_hkd)/hist_tencent_hk.loc[0,'close']) * hkd_to_u

print(f'Nintendo expected return: ${n_er:,.2f}')
print(f'Rockstar expected return: ${r_er:,.2f}')
print(f'Tencent expected return: ${t_er:,.2f}')
```

```
Nintendo expected return: $96,231.06
Rockstar expected return: $29,797.60
Tencent expected return: $72,501.59
```

variance

In [ ]:
```
n_var = hist_nintendo_jp['return_log'].var()
r_var = hist_rockstar_us['return_log'].var()
t_var = hist_tencent_hk['return_log'].var()

print(f'Nintendo expected variance: {n_var:,.4f}')
print(f'Rockstar expected variance: {r_var:,.4f}')
print(f'Tencent expected variance: {t_var:,.4f}')

jp_var = hist_jp_index['return_log'].var()
us_var = hist_us_index['return_log'].var()
hk_var = hist_hk_index['return_log'].var()
```

```
Nintendo expected variance: 0.0002
Rockstar expected variance: 0.0003
Tencent expected variance: 0.0005
```

beta

In [ ]:
```
nintendo_beta = np.cov(hist_nintendo_jp['return_log'], hist_jp_index['return_log'])/jp_var
rockstar_beta = np.cov(hist_rockstar_us['return_log'], hist_us_index['return_log'])/us_var
tencent_beta = np.cov(hist_tencent_hk['return_log'], hist_hk_index['return_log'])/hk_var

print(f'Nintendo beta: {nintendo_beta[0,1]:,.4f}')
print(f'Rockstar beta: {rockstar_beta[0,1]:,.4f}')
print(f'Tencent beta: {tencent_beta[0,1]:,.4f}')
```

```
Nintendo beta: 0.6603
Rockstar beta: 1.0396
Tencent beta: 1.2256
```

VaR Historical

In [ ]:
```
n_to_n_covar = np.cov(hist_nintendo_jp['return_log'], hist_nintendo_jp['return_log'])
n_to_r_covar = np.cov(hist_nintendo_jp['return_log'], hist_rockstar_us['return_log'])
n_to_t_covar = np.cov(hist_nintendo_jp['return_log'], hist_tencent_hk['return_log'])

r_to_n_covar = np.cov(hist_rockstar_us['return_log'], hist_nintendo_jp['return_log'])
r_to_r_covar = np.cov(hist_rockstar_us['return_log'], hist_rockstar_us['return_log'])
r_to_t_covar = np.cov(hist_rockstar_us['return_log'], hist_tencent_hk['return_log'])

t_to_n_covar = np.cov(hist_tencent_hk['return_log'], hist_nintendo_jp['return_log'])
t_to_r_covar = np.cov(hist_tencent_hk['return_log'], hist_rockstar_us['return_log'])
```

```
t_to_t_covar = np.cov(hist_tencent_hk['return_log'], hist_tencent_hk['return_log'])

covarmatrix = np.array([[n_to_n_covar[0,1], n_to_r_covar[0,1], n_to_t_covar[0,1]],
                        [r_to_n_covar[0,1], r_to_r_covar[0,1], r_to_t_covar[0,1]],
                        [t_to_n_covar[0,1], t_to_r_covar[0,1], t_to_t_covar[0,1]]])

triple_valueatrisk = covarmatrix.dot(np.array([(1/3), (1/3), (1/3)]))
triple_valueatrisk = triple_valueatrisk.dot(np.array([(1/3), (1/3), (1/3)]))

print(f'Triple value at risk: {triple_valueatrisk:,.4f}')


triple_std = np.sqrt(triple_valueatrisk)
print(f'Triple standard deviation: {triple_std:,.4f}')
```

```
Triple value at risk: 0.0001
Triple standard deviation: 0.0116
```

1 Day Historical VaR

In [ ]:
```
hvar_1day_asreturn = stats.norm.ppf(.95)*triple_std
print(f'Historical value at risk (return) 1 day: {hvar_1day_asreturn:,.4f}')

hvar_1day_asvalue = hvar_1day_asreturn * 10000000
print(f'Historical value at risk( value) 1 day: ${hvar_1day_asvalue:,.2f}')
```

```
Historical value at risk (return) 1 day: 0.0191
Historical value at risk( value) 1 day: $190,672.16
```

5 Day Historical VaR

In [ ]:
```
hvar_5day_asreturn = hvar_1day_asreturn * np.sqrt(5)
print(f'Historical value at risk (return) 5 days: {hvar_5day_asreturn:,.4f}')

hvar_5day_asvalue = hvar_5day_asreturn * 10000000
print(f'Historical value at risk( value) 5 days: ${hvar_5day_asvalue:,.2f}')
```

```
Historical value at risk (return) 5 days: 0.0426
Historical value at risk( value) 5 days: $426,355.91
```

Other historical VaR based on position?

In [ ]:
```
portfolio
```

Out[ ]:

	date	close	returns
0	2023-02-21	192.651482	-0.044522
1	2023-02-22	190.493943	-0.041471
2	2023-02-24	187.620193	-0.038803
3	2023-02-27	187.765069	-0.011554
4	2023-02-28	186.024329	-0.035651
...	...	...	...
220	2024-02-08	263.687508	0.013057
221	2024-02-09	249.453384	-0.079424
222	2024-02-14	251.836248	0.036747
223	2024-02-15	253.014276	0.001006
224	2024-02-16	249.746261	-0.008875

225 rows × 3 columns

In [ ]:
```
pos = round(portfolio.shape[0] * .05)
oh_var = portfolio.loc[pos,'returns'] * 10000000
print(f'Historical value at risk (value) 1 day: ${oh_var:,.2f}')
```

```
Historical value at risk (value) 1 day: $-423,129.42
```

Guassian VaR

In [ ]:
```
g_var = (-1.65 * protfolio_std) * 10000000
print(f'Gaussian value at risk: ${g_var:,.2f}')
```

```
Gaussian value at risk: $-575,555.80
```

Log Normal

In [ ]:
```
np.exp(1)
```

Out[ ]:
```
2.718281828459045
```

In [ ]:
```
log_norm_var = 10_000_000 * (1- np.exp(1) ** (portfolio_mean - (1.645 * protfolio_std)))
print(f'Log normal value at risk: ${log_norm_var:,.2f}')
```

```
Log normal value at risk: $527,774.60
```

Expected Shortfall

In [ ]:
```
es_df = portfolio.copy(deep=True)
es_df['returns_asc'] = es_df['returns'].sort_values(ascending=True)

es_oh_var = es_df.loc[pos, 'returns']
```

In [ ]:
```
def tail_loss(x):
    if x > es_oh_var:
        return 0
    else:
        return 1
```

In [ ]:
```
es_df['tail_loss'] = es_df['returns_asc'].apply(tail_loss)
```

In [ ]:
```
es_df['tail_loss'].value_counts()
```

Out[ ]:
```
tail_loss
0    205
1     20
Name: count, dtype: int64
```

In [ ]:
```
es = es_df[es_df['tail_loss'] == 1]['returns_asc'].mean() * 10000000
print(f'Expected shortfall: ${es:,.4f}')
```

```
Expected shortfall: $-572,092.0533
```