Week 2: FRA - Financial Risk Analytics - Python Cheat Sheet

1. Returns and Risk:

- Importing necessary libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- Calculating simple returns:

```
def calculate_simple_returns(prices):
    return (prices / prices.shift(1)) - 1

returns = calculate_simple_returns(prices)
```

- Calculating logarithmic returns:

```
def calculate_log_returns(prices):
    return np.log(prices / prices.shift(1))

log_returns = calculate_log_returns(prices)
```

- Calculating portfolio returns:

```
portfolio_weights = np.array([0.4, 0.6])  # Example weights
portfolio_returns = np.dot(returns, portfolio_weights)
```

- Calculating portfolio risk (variance and standard deviation):

```
portfolio_variance = np.dot(portfolio_weights.T, np.dot(cov_matrix, portfolio_weights))
portfolio_std_dev = np.sqrt(portfolio_variance)
```

2. Market Risk Case Hands-on:

- Importing necessary libraries:

```
import pandas as pd
import numpy as np
import scipy.optimize as sco
```

- Loading and preprocessing data:

```
data = pd.read_csv('stock_prices.csv')
returns = data.pct_change()
```

- Calculating mean returns and covariance matrix:

```
mean_returns = returns.mean()
cov_matrix = returns.cov()
```

- Defining the objective function for portfolio optimization:

```
def portfolio_variance(weights, cov_matrix):
    return np.dot(weights.T, np.dot(cov_matrix, weights))
```

- Optimizing portfolio weights:

```
num_assets = len(data.columns)
initial_weights = np.array([1/num_assets] * num_assets)  # Example initial weights

constraints = [{'type': 'eq', 'fun': lambda x: np.sum(x) - 1}]  # Constraint: weights sum to 1
bounds = tuple((0, 1) for _ in range(num_assets))  # Bounds: weights between 0 and 1

optimal_weights = sco.minimize(portfolio_variance, initial_weights, args=(cov_matrix,),
                               method='SLSQP', bounds=bounds, constraints=constraints)
```

- Calculating optimal portfolio returns and risk:

```
optimal_portfolio_returns = np.dot(mean_returns, optimal_weights.x)
optimal_portfolio_variance = np.dot(optimal_weights.x.T, np.dot(cov_matrix, optimal_weights.x))
optimal_portfolio_std_dev = np.sqrt(optimal_portfolio_variance)
```

3. Market Risk Optimization:

- Importing necessary libraries:

```
import pandas as pd
import numpy as np
import cvxpy as cp
```

- Defining the optimization problem using CVXPY:

```
num_assets = len(data.columns)
weights = cp.Variable(num_assets)
returns = np.array(mean_returns)
risk = cp.quad_form(weights, np.array(cov_matrix))

objective = cp.Maximize(returns @ weights - 0.5 * risk)
constraints = [cp.sum(weights) == 1, weights >= 0]
problem = cp
```