## Strategy Overview

The mean-reversion trading strategy implemented in the provided script is based on the Ornstein-Uhlenbeck (OU) process. This statistical method is used to model the mean-reverting behavior of financial time series, such as stock prices. The strategy aims to exploit temporary deviations from a long-term mean, expecting that prices will revert to the mean over time.

## Key Concepts of the Strategy

### Mean-Reversion

Mean-reversion is the theory that prices and returns eventually move back towards the mean or average. This mean or average can be the historical average price, the average return, or another relevant average.

### Ornstein-Uhlenbeck Process

The OU process is a type of stochastic differential equation used to model mean-reverting behavior. It is characterized by three parameters:

- Mean reversion speed ($\mu$): The speed at which the price reverts to the mean.
- Long-term mean ($\theta$): The average price level to which the price reverts.
- Volatility ($\sigma$): The degree of variation or dispersion of the price.

### Parameters Estimation

Historical price data of the stock (AAPL in this case) is used to estimate the parameters of the OU process. These parameters help in calculating optimal entry and exit points for trades.

### Entry and Exit Levels

Using the estimated parameters, the script calculates optimal entry ($a^*$) and exit ($b^*$) levels. These levels determine when to open and close positions based on the mean-reversion principle.

## Detailed Steps of the Strategy

### Connect to IBKR

The script starts by connecting to the Interactive Brokers Trader Workstation (IBKR TWS) using the `ib_insync` library.

## Fetch Historical Data

The script fetches historical price data for the specified stock (AAPL) over the past 5 days, with 1-minute intervals. This data is used for parameter estimation and to check trading conditions.

## Estimate OU Parameters

The historical data is used to estimate the parameters of the OU process: mean reversion speed ($\mu$), long-term mean ($\theta$), and volatility ($\sigma$). This is done by maximizing the likelihood of the observed data under the OU model.

## Calculate Entry and Exit Levels

Based on the estimated parameters, the script calculates the optimal entry ($a^*$) and exit ($b^*$) levels. These levels indicate the price points at which it is statistically advantageous to enter and exit trades.

## Check Trading Condition

The script continuously fetches the latest price data and checks if the price has deviated significantly from the mean (i.e., if the price movement exceeds a specified threshold).

## Place Market Orders

If the trading condition is met (i.e., price has deviated enough from the mean), the script places a market order to open a position. It uses a market order to ensure immediate execution at the current market price. After placing the order, the script prints the details of the executed trade, including the price at which the order was filled.

## Hold the Position

The script holds the position for a specified duration (5 minutes in this case).

## Close the Position

After the holding period, the script places a market order to close the position. Again, it uses a market order to ensure immediate execution. The script prints the details of the closing trade and calculates the profit or loss from the trade.

## Repeat the Process

The script waits for a specified interval (60 seconds) before fetching new data and checking trading conditions again. This ensures that the strategy continuously monitors the market and makes trades based on the latest data.

## Example Walkthrough

### Connect to IBKR

The script connects to the IBKR TWS using the provided host and port details.

### Fetch Historical Data

It retrieves 5 days of historical price data for AAPL, with 1-minute bars.

### Estimate Parameters

The OU parameters are estimated based on the historical data. Suppose the estimated parameters are:

- $\mu = 0.01$ (mean reversion speed)
- $\theta = 150$ (long-term mean price)
- $\sigma = 2$ (volatility)

### Calculate Entry and Exit Levels

Using the estimated parameters, the script calculates the optimal entry ($a^*$) and exit ($b^*$) levels. Suppose the calculated levels are:

- $a^* = 145$ (entry level)
- $b^* = 155$ (exit level)

### Check Condition and Place Order

The latest price data shows that the price has dropped to 144. This triggers the condition to open a position because the price is below the entry level ($a^*$). The script places a market order to sell 200 shares of AAPL. The trade is executed at a price of 144.50.

### Hold the Position

The script holds the position for 5 minutes.

### Close the Position

After 5 minutes, the script places a market order to buy back 200 shares of AAPL. The trade is executed at a price of 143.50. The script calculates the profit as $(144.50 - 143.50) \times 200 = \$200$.

### Repeat

The script waits for 60 seconds before fetching new data and checking conditions again.

## Detailed Breakdown of the Code

### Imports and Initialization

The necessary libraries are imported, and the connection to IBKR is established. `nest_asyncio.apply()` is used to allow nested use of `asyncio.run` in Jupyter notebooks.

### Function Definitions

- `Now()`: Returns the current time as a string for logging.
- `fetch_historical_data()`: Fetches historical data for the given contract.
- `estimate_ou_parameters()`: Estimates the OU parameters ($\mu$, $\theta$, $\sigma$) from the historical data.
- `calculate_entry_exit_levels()`: Calculates the optimal entry and exit levels ($a^*$, $b^*$) based on the OU parameters.
- `place_market_order()`: Places a market order for the specified action (BUY/SELL) and quantity.
- `get_account_balance()`: Retrieves the current account balance.

### Main Strategy Logic

The main logic is wrapped in an asynchronous `main()` function to handle asynchronous data fetching and order placement. Parameters for the strategy, such as the check window, bar size, contract, discount rate, transaction cost, stop-loss percentage, trade amount, hold time, and wait time, are defined. Historical data is fetched, and OU parameters are estimated. Entry and exit levels are calculated. The initial account balance is retrieved and printed. The script enters a continuous loop where it fetches the latest price data, checks the trading condition, and places market orders if the condition is met. The script prints detailed logs of each step, including order placement and trade details.

## Conclusion

This mean-reversion strategy leverages the Ornstein-Uhlenbeck process to identify optimal trading opportunities based on the statistical properties of price movements. By continuously monitoring the market and placing trades based on calculated entry and exit levels, the strategy aims to profit from temporary price deviations that are expected to revert to the mean. The use of market orders ensures immediate execution, allowing the strategy to take advantage of favorable price movements quickly.