Trading signals

FINANCIAL TRADING IN PYTHON

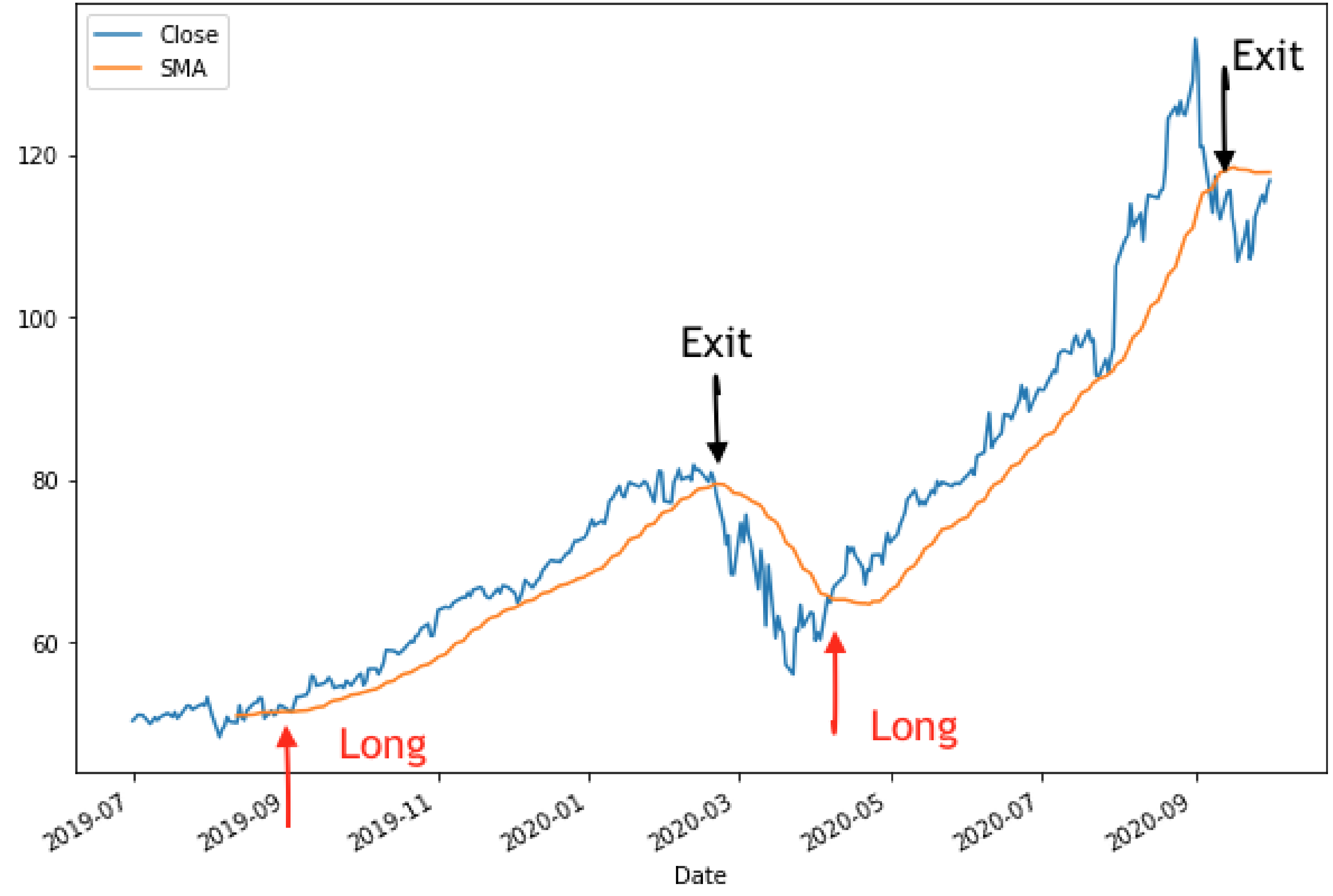
Chelsea Yang
Data Science Instructor

What are trading signals?

- Triggers to long or short financial assets based on predetermined criteria
- Can be constructed using:
  - One technical indicator
  - Multiple technical indicators
  - A combination of market data and indicators
- Commonly used in algorithmic trading

A signal example

- Signal: Price > SMA (long when the price rises above the SMA)



How to implement signals in bt

1. Get the data and calculate indicators
2. Define the signal-based strategy
   - `bt.algos.SelectWhere()`
   - `bt.algos.WeighTarget()`
3. Create and run a backtest
4. Review the backtest result

Construct the signal

```
# Get price data by the stock ticker
price_data = bt.get('aapl', start='2019-11-1', end='2020-12-1')
```

```
# Calculate SMA
sma = price_data.rolling(20).mean()
```

Alternatively, use `talib` to calculate the indicator:

```
# Calculate SMA
import talib
sma =  talib.SMA(price_data['Close'], timeperiod=20)
```

Define a signal-based strategy

```
# Define the signal-based strategy
bt_strategy = bt.Strategy('AboveEMA',
                          [bt.algos.SelectWhere(price_data > sma),
                           bt.algos.WeighEqually(),
                           bt.algos.Rebalance()])
```

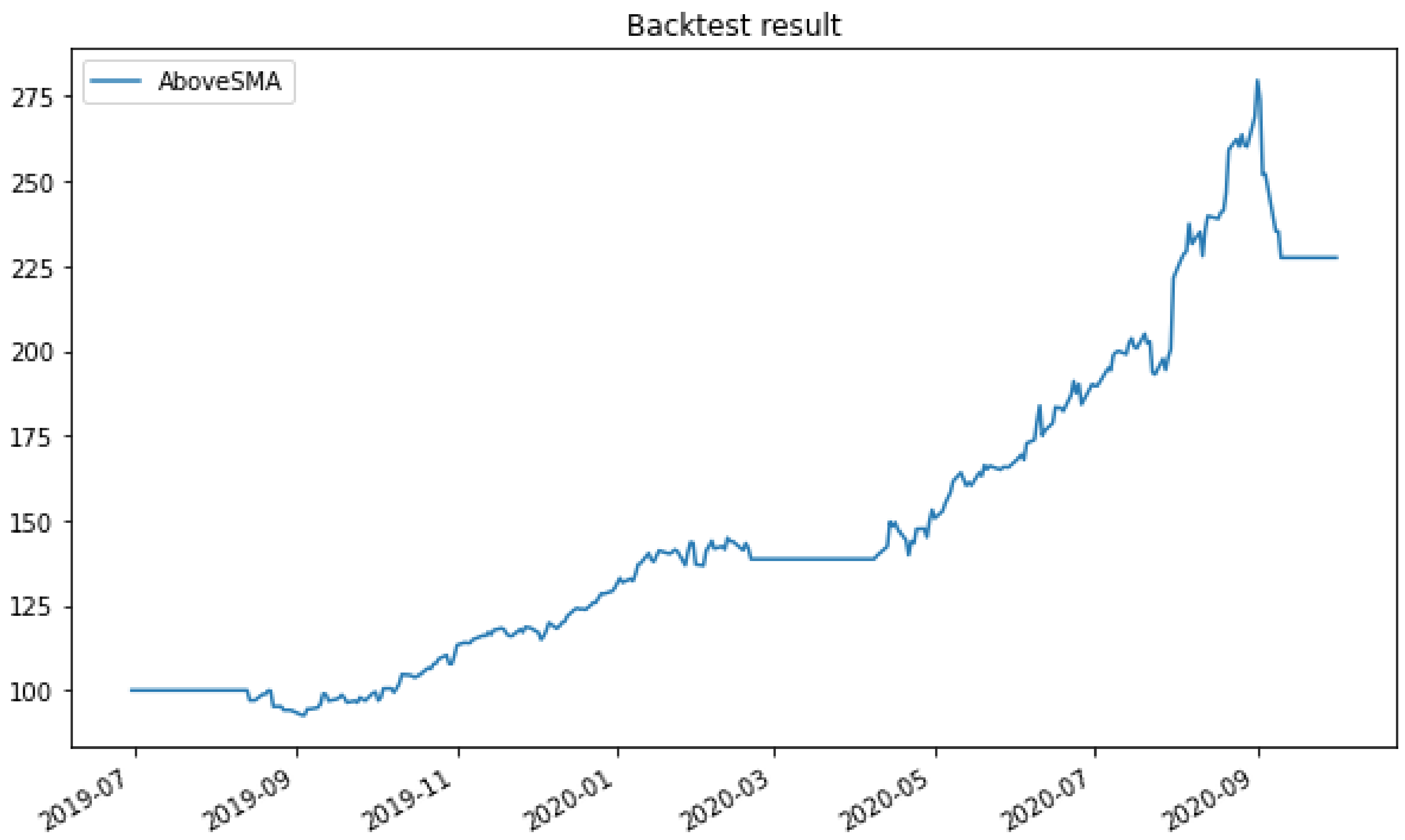
- For simplicity, we assume:
  - Trade one asset at a time
  - No slippage or commissions
    - Slipage: the difference between the expected price of a trade and the price at which the trade is executed
    - Commission: fees charged by brokers when executing a trade

Backtest the signal based strategy

```
# Create the backtest and run it
bt_backtest = bt.Backtest(bt_strategy, price_data)
bt_result = bt.run(bt_backtest)
```

Plot the backtest result

```
# Plot the backtest result
bt_result.plot(title='Backtest result')
```



Let's practice!

FINANCIAL TRADING IN PYTHON

Trend-following strategies

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

Two types of trading strategies

Trend-following

- Bet the price trend will continue in the same direction
- Use trend indicators such as moving averages, ADX, etc to construct trading signals

Mean reversion

- Bet the price tends to reverse back towards the mean
- Use indicators such as RSI, Bollinger Bands, etc, to construct trading signals

MA crossover strategy

The trend is your friend.

- Two EMA crossover:
  - Long signal: the short-term EMA crosses above the long-term EMA
  - Short signal: the short-term EMA crosses below the long-term EMA

Calculate the indicators

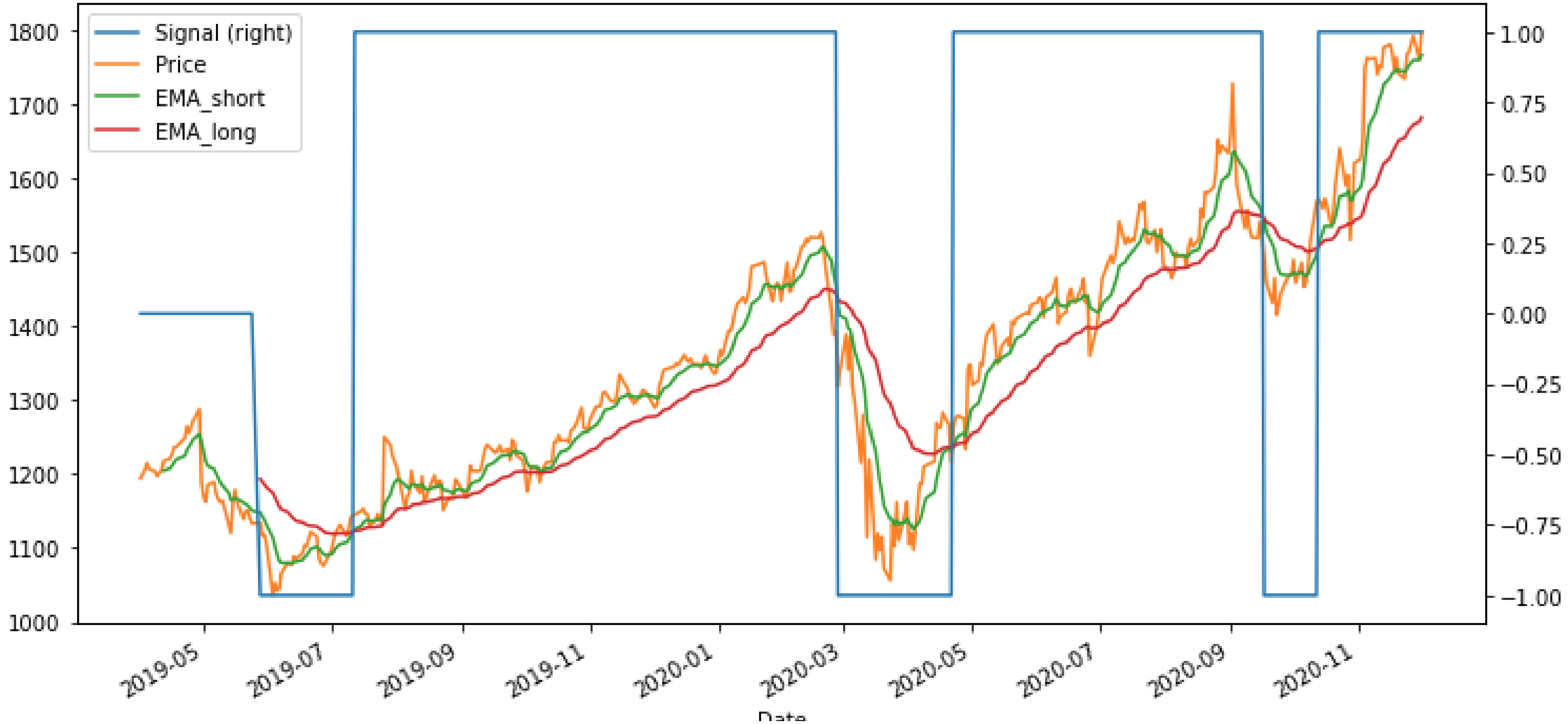
```
import talib
# Calculate the indicators
EMA_short = talib.EMA(price_data['Close'],
                      timeperiod=10).to_frame()
EMA_long = talib.EMA(price_data['Close'],
                     timeperiod=40).to_frame()
```

Construct the signal

```
# Create the signal DataFrame
signal = EMA_long.copy()
signal[EMA_long.isnull()] = 0

# Construct the signal
signal[EMA_short > EMA_long] = 1
signal[EMA_short < EMA_long] = -1
```

Plot the signal

```
# Plot the signal, price and MAs
combined_df = bt.merge(signal, price_data, EMA_short, EMA_long)
combined_df.columns = ['Signal', 'Price', 'EMA_short', 'EMA_long']
combined_df.plot(secondary_y=['Signal'])
```



Define the strategy with the signal

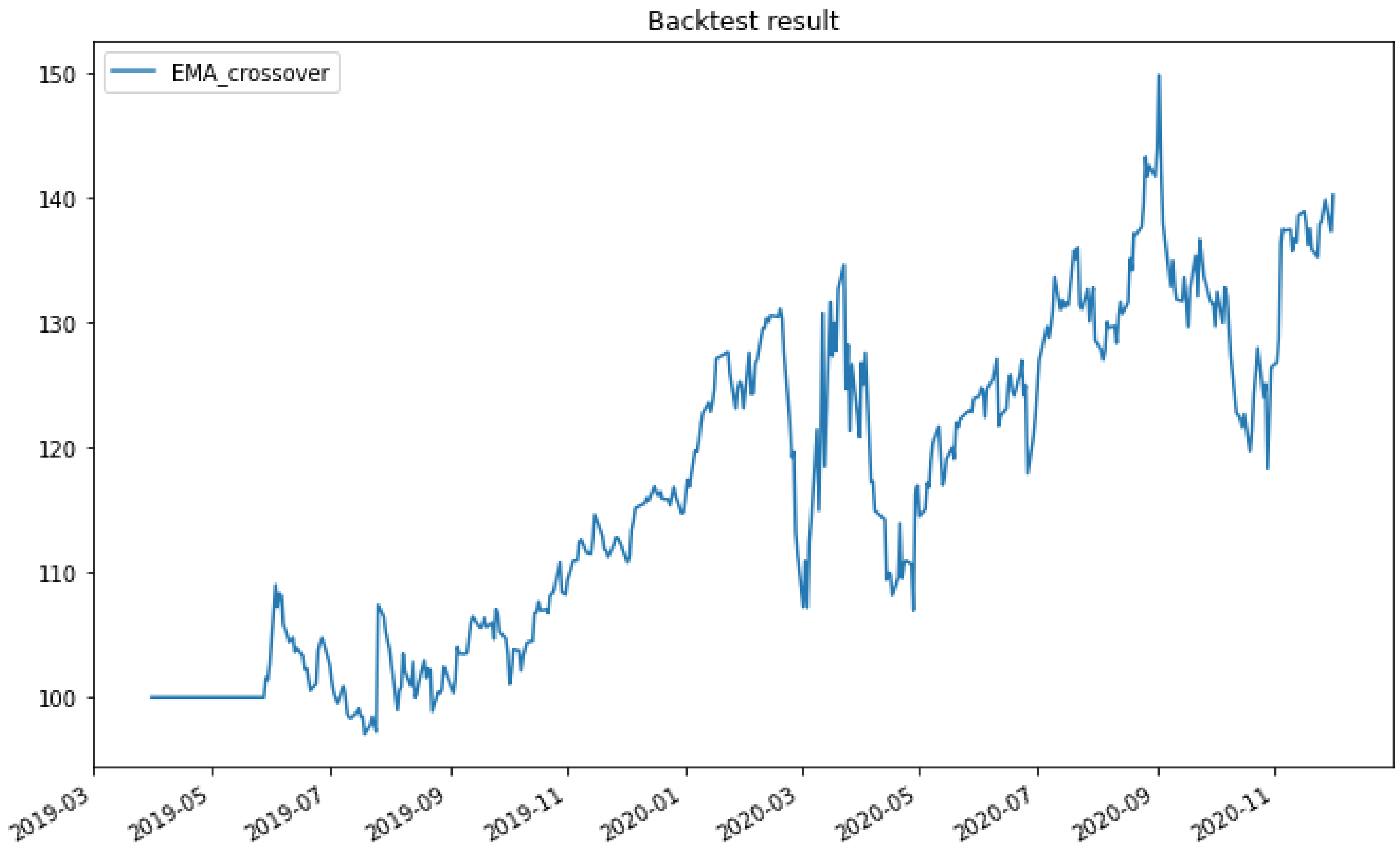
```
# Define the strategy
bt_strategy = bt.Strategy('EMA_crossover',
                          [bt.algos.WeighTarget(signal),
                           bt.algos.Rebalance()])
```

Backtest the signal based strategy

```
# Create the backtest and run it
bt_backtest = bt.Backtest(bt_strategy, price_data)
bt_result = bt.run(bt_backtest)
```

Plot backtest results

```
# Plot the backtest result
bt_result.plot(title='Backtest result')
```



Let's practice!

FINANCIAL TRADING IN PYTHON

Mean reversion strategy

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

RSI-based mean reversion strategy

Buy the fear and sell the greed

- RSI-based mean reversion strategy:
  - Short signal: RSI > 70
    - Suggests the asset is likely overbought and the price may soon reverse
  - Long signal: RSI < 30
    - Suggests the asset is likely oversold and the price may soon rally

Calculate the indicator

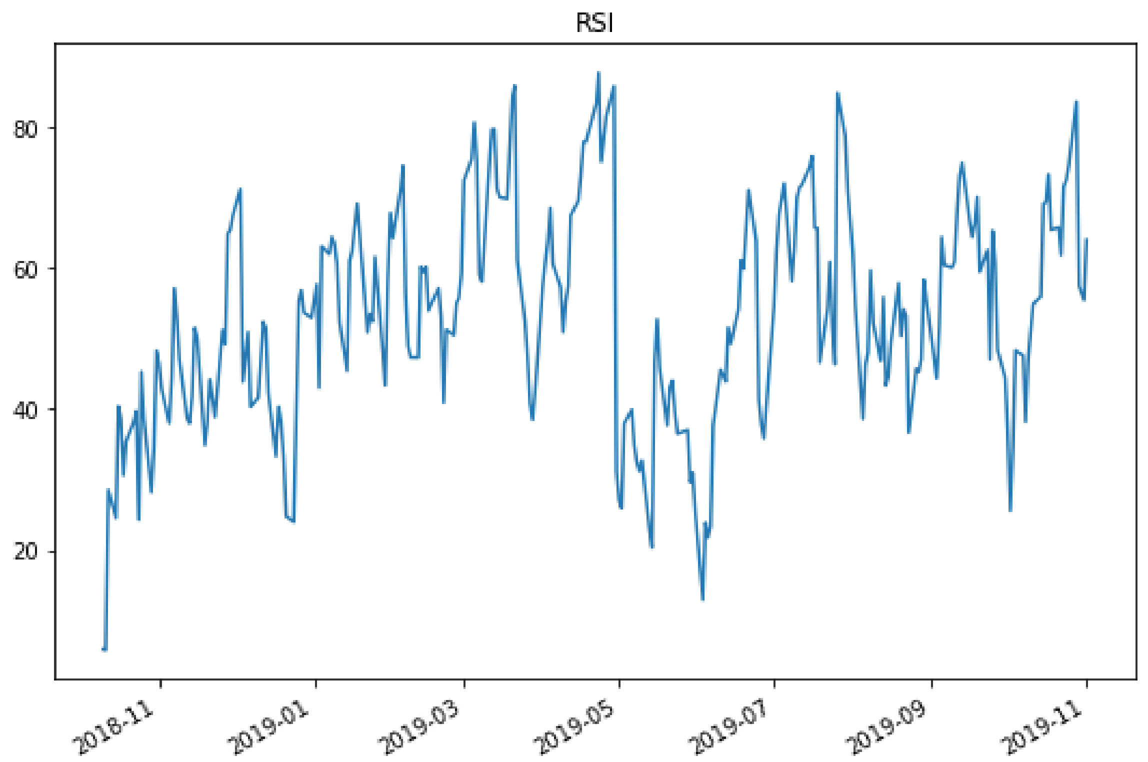
```
import talib
# Calculate the RSI
stock_rsi = talib.RSI(price_data['Close']).to_frame()
```

Construct the signal

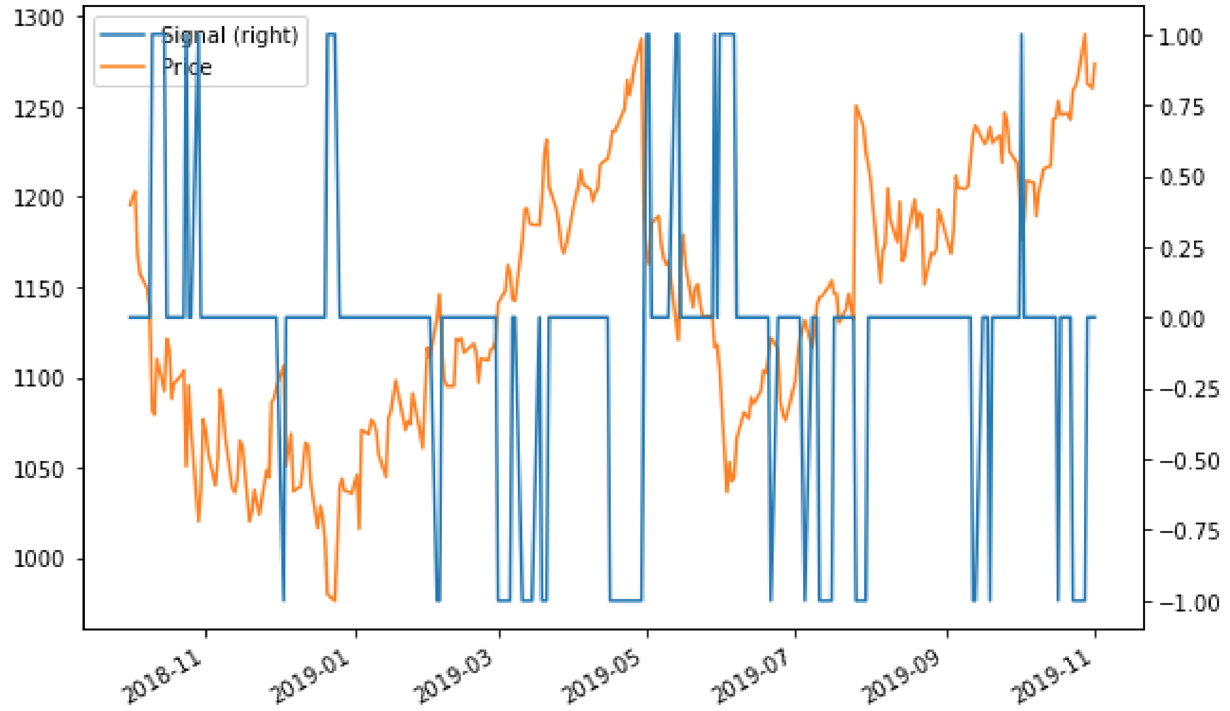
```
# Create the same DataFrame structure as RSI
signal = stock_rsi.copy()
signal[stock_rsi.isnull()] = 0

# Construct the signal
signal[stock_rsi < 30] = 1
signal[stock_rsi > 70] = -1
signal[(stock_rsi <= 70) & (stock_rsi >= 30)] = 0
```

Plot the signal

```
# Plot the RSI
stock_rsi.plot()
plt.title('RSI')
```




```
# Merge data into one DataFrame
combined_df = bt.merge(signal, stock_data)
combined_df.columns = ['Signal', 'Price']
# Plot the signal with price
combined_df.plot(secondary_y = ['Signal'])
```



Define the strategy with the signal

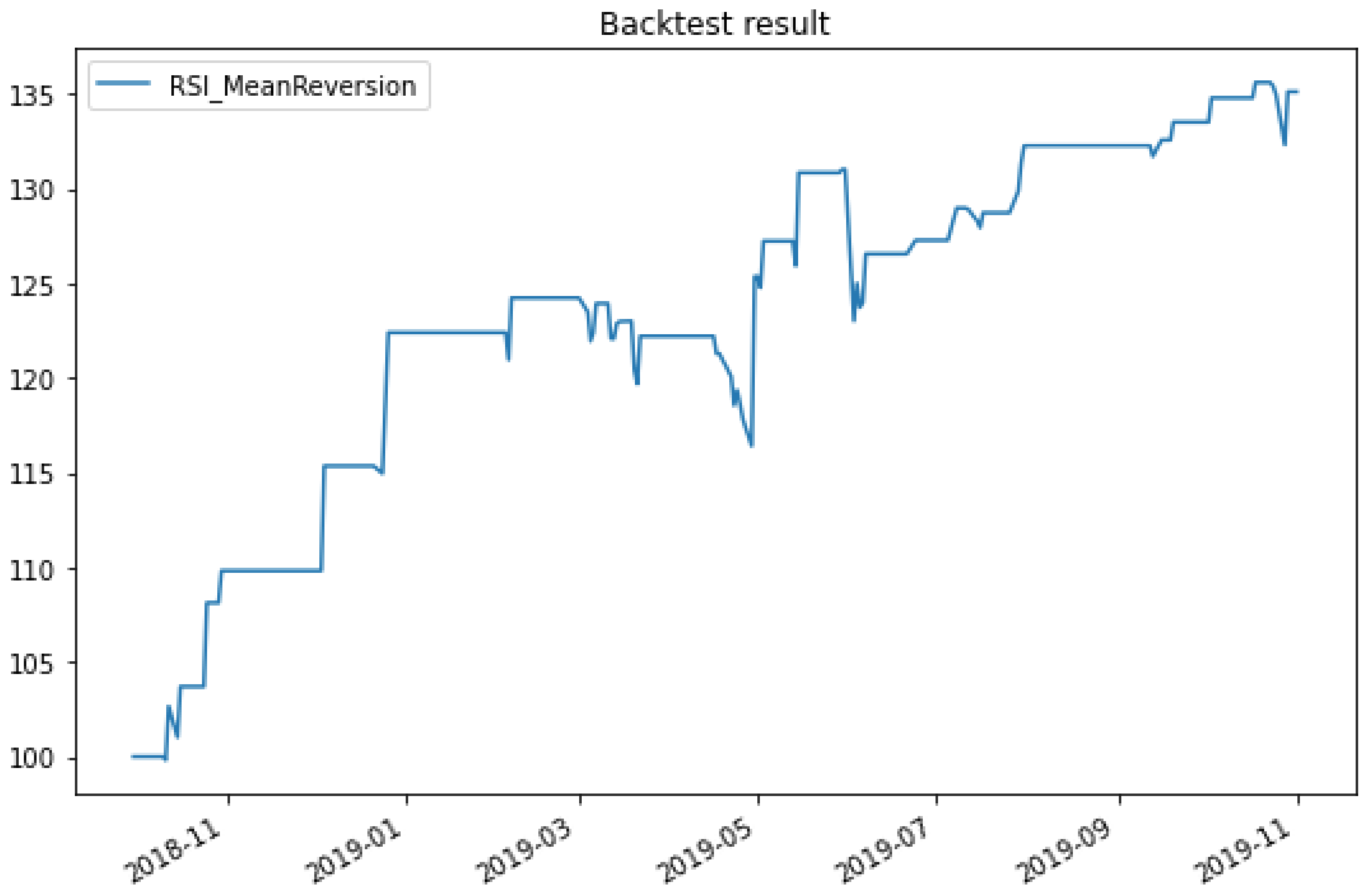
```
# Define the strategy
bt_strategy = bt.Strategy('RSI_MeanReversion',
                          [bt.algos.WeighTarget(signal),
                           bt.algos.Rebalance()])
```

Backtest the signal-based strategy

```
# Create the backtest and run it
bt_backtest = bt.Backtest(bt_strategy, price_data)
bt_result = bt.run(bt_backtest)
```

Plot backtest result

```
# Plot the backtest result
bt_result.plot(title='Backtest result')
```



Let's practice!

FINANCIAL TRADING IN PYTHON

Strategy optimization and benchmarking

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

How to decide the values of input parameters?

- Question:
  - Which is a better SMA lookback period to use in the signal "Price > SMA"?
- Solution: strategy optimization
  - Try a range of input parameter values in backtesting and compare the results

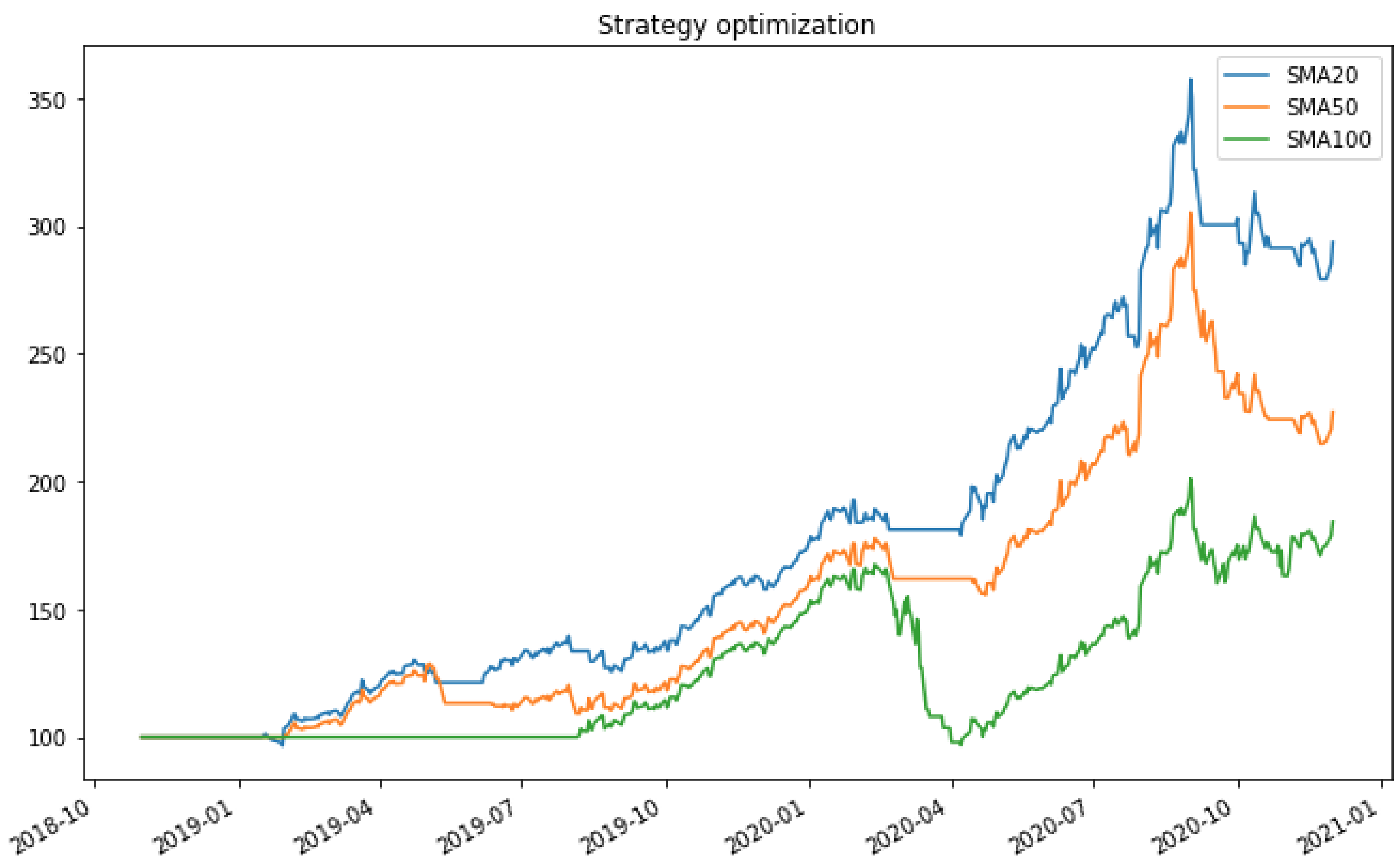
Strategy optimization example

```
def signal_strategy(ticker, period, name, start='2018-4-1', end='2020-11-1'):
    # Get the data and calculate SMA
    price_data = bt.get(ticker, start=start, end=end)
    sma = price_data.rolling(period).mean()
    # Define the signal-based strategy
    bt_strategy = bt.Strategy(name,
                              [bt.algos.SelectWhere(price_data>sma),
                               bt.algos.WeighEqually(),
                               bt.algos.Rebalance()])
    # Return the backtest
    return bt.Backtest(bt_strategy, price_data)
```

[1] www.datacamp.com/courses/writing-functions-in-python

Strategy optimization example

```
ticker = 'aapl'
sma20 = signal_strategy(ticker,
                        period=20, name='SMA20')
sma50 = signal_strategy(ticker,
                        period=50, name='SMA50')
sma100 = signal_strategy(ticker,
                         period=100, name='SMA100')

# Run backtests and compare results
bt_results = bt.run(sma20, sma50, sma100)
bt_results.plot(title='Strategy optimization')
```



What is a benchmark?

A standard or point of reference against which a strategy can be compared or assessed.

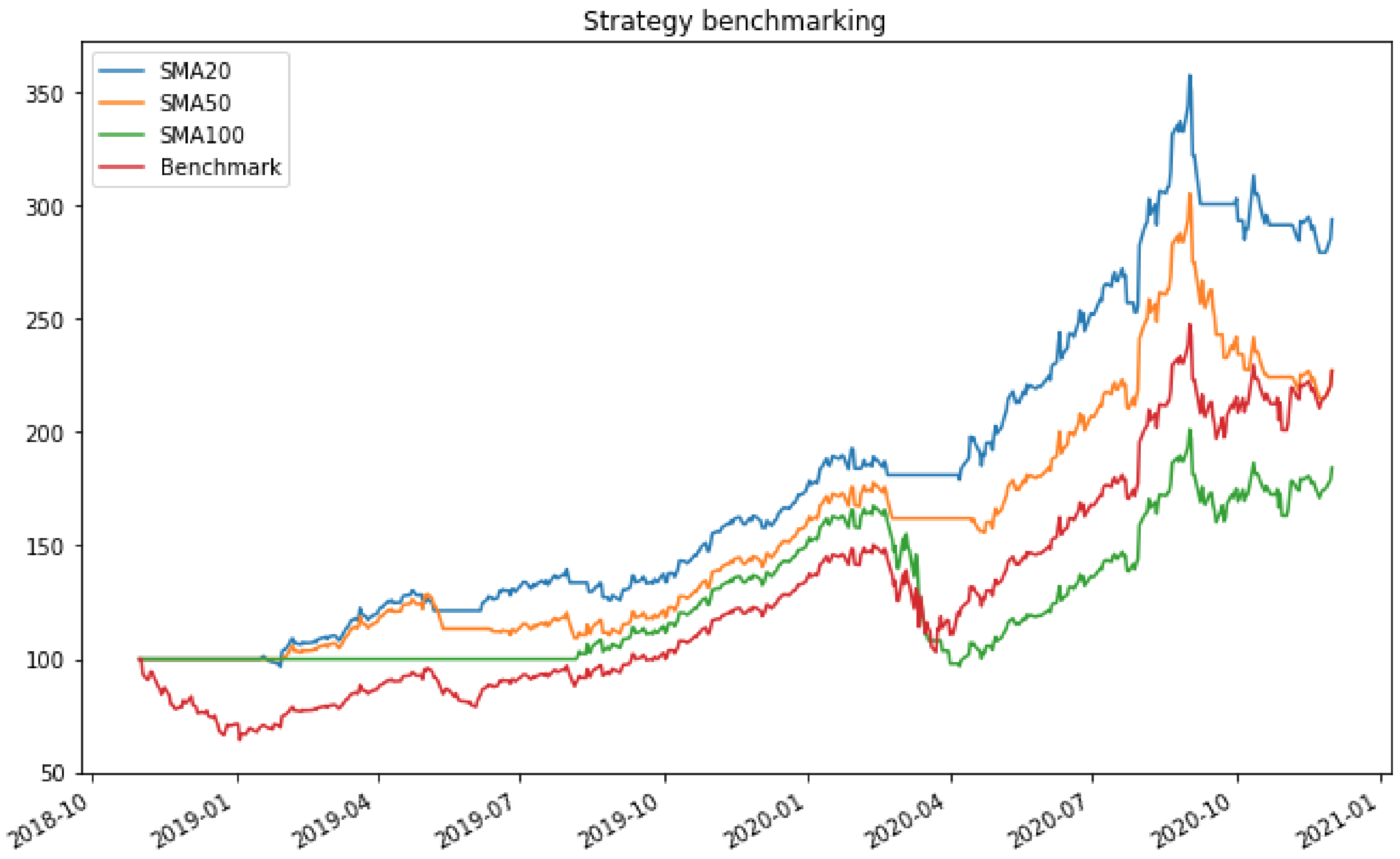
- Example:
  - A strategy that uses signals to actively trade stocks can use a passive buy and hold strategy as a benchmark.
  - The S&P 500 index is often used as a benchmark for equities.
  - U.S. Treasuries are used for measuring bond risks and returns.

Benchmarking example

```
def buy_and_hold(ticker, name, start='2018-11-1', end='2020-12-1'):
    # Get the data
    price_data = bt.get(ticker, start=start_date, end=end_date)
    # Define the benchmark strategy
    bt_strategy = bt.Strategy(name,
                              [bt.algos.RunOnce(),
                               bt.algos.SelectAll(),
                               bt.algos.WeighEqually(),
                               bt.algos.Rebalance()])
    # Return the backtest
    return bt.Backtest(bt_strategy, price_data)
```

Benchmarking example

```
benchmark = buy_and_hold(ticker, name='benchmark')
# Run all backtests and plot the resutls
bt_results = bt.run(sma20, sma50, sma100, benchmark)
bt_results.plot(title='Strategy benchmarking')
```



Let's practice!

FINANCIAL TRADING IN PYTHON