TBP-04 and 05: Lecture Summary

Overview

This document summarizes the lectures on backtesting and live trading strategies. This lecture introduces the IBridgePy platform and its implementation. After this lecture, you should be able to create trading strategies using IBridgePy and do live trading.

The lecture covers the following topics:

- How to set up a hedge fund?
- Major components of algo trading desk
- Various approaches to build strategies in Python
- Introduction to IBridgePy
- Handling errors in live trading
- Steps to build algo trading platform
- Roadmap: An idea to live trading
- Backtesting using historical data from IB
- What are the considerations when implementing HFT?
- Local computer vs Cloud computing

How to start a hedge fund?

- Research to find good strategies (fundamental analysis vs quantitative analysis)
- Translate strategy into Python code if you don’t want to trade manually
- Set up a server to run Python code
- Summary trading results
- Improve performance
- Show track records to investors
- Set up investment accounts (advisor account vs pooled account)
- Manage client portfolio

Major components of algo trading desk

- Successful strategies
- Python codes translated from the strategies
- A computer (local or cloud)
- A Python platform/environment to backtest and live trade
- Internet connection
- Broker accounts
- Customers

Various approaches to build strategies in Python

	Example	Advantages	Disadvantages
Web-based trading platforms	Blueshift, Backtrader	Huge data No installation Backtest	- No privacy - Coding limitations - Limited trading products
Low-level Python wrappers	IBpy, IB_insync	Live trade Basic to advanced functions	- hard to learn - No backtest - No company for support
Backtesting engines	Zipline	Backtest features Great community	- No live trading (there is a GitHub fork to do live trading) - Limited trading products - The future of development is not clear since Quantopian has stopped its operations.
Broker's native API	IB, TD Ameritrade, RESTful, JSON	Live trade Some supports	- Hard to learn - No backtesting - It takes a long time to code
Self-hosted solutions	IBridgePy	Privacy Easy to use Backtest and live	- Backtest needs data from providers - A dedicated server

<u>Introduction to IBridgePy</u>

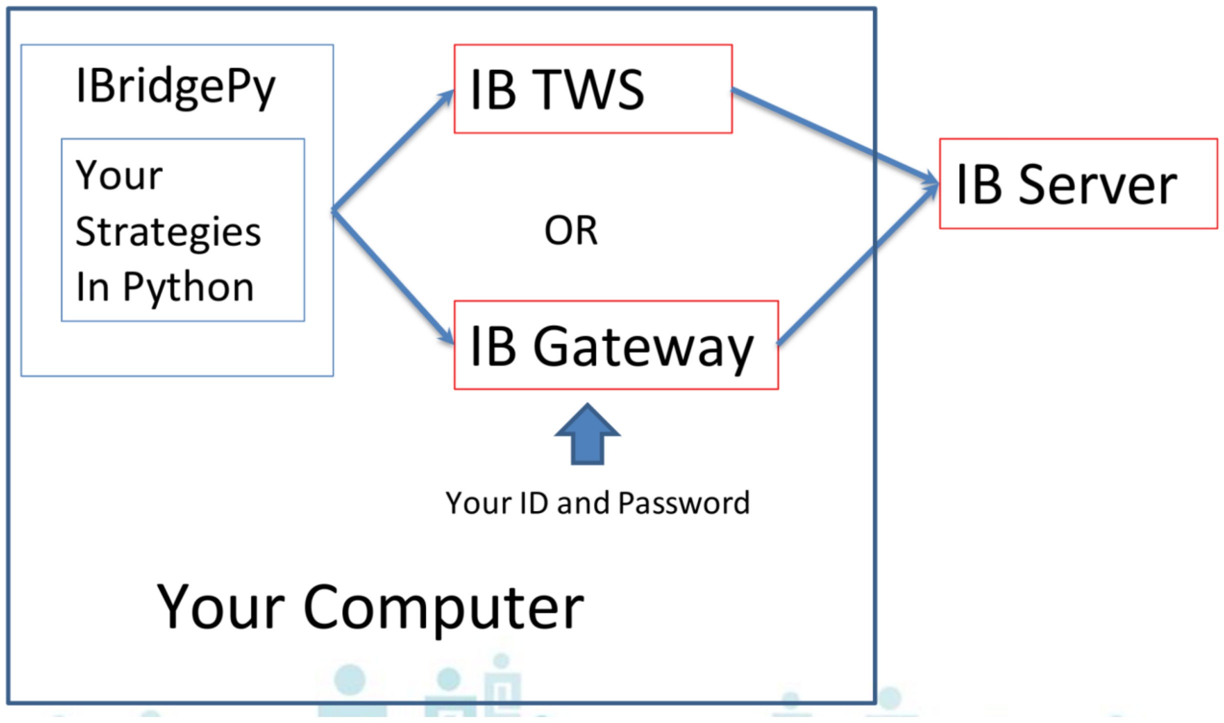
IBridgePy is a Python platform that provides backtesting and live trade features using different brokers such as Interactive Brokers, TD Ameritrade, and Robinhood.

Features of IBridgePy

- Manage multiple accounts at the same time

- Execute multiple trading strategies at the same time
- Run Quantopian-style algorithms

Interaction of IBridgePy with IB



The code structure of IBridgePy

Here are the main functions:

- initialize()   - It is a built-in method to claim variables. It will only be run once
- handle_data() - It is also a built-in method where trading decisions are made. Two inputs are given here (context, data). "Context" contains the variables claimed in initialize(). "data" contains account information and near real-time quotes received from IB
- show_real_time_price() - To get real time price
- request_historical_data() - To get historical price
- order() - To place order

Usage of these functions is available in the Python file provided in the LMS portal.

Here's a video with a live demo: Live market demo

Premium features of IBridgePy

- Get_contract_details: For option chains

- Get_scanner_results : stock scanner
- Get_option_greeks: For trading options

__Handling errors in live trading__

What errors could occur?

- User has no permission to access IB’s data -- Call IB representative
- The market is not open -- the real-time price is not fresh
- Request historical data too often -- minimise requests
- Not enough margin to place orders -- use try-catch
- Order status is unknown -- use order_status_monitor

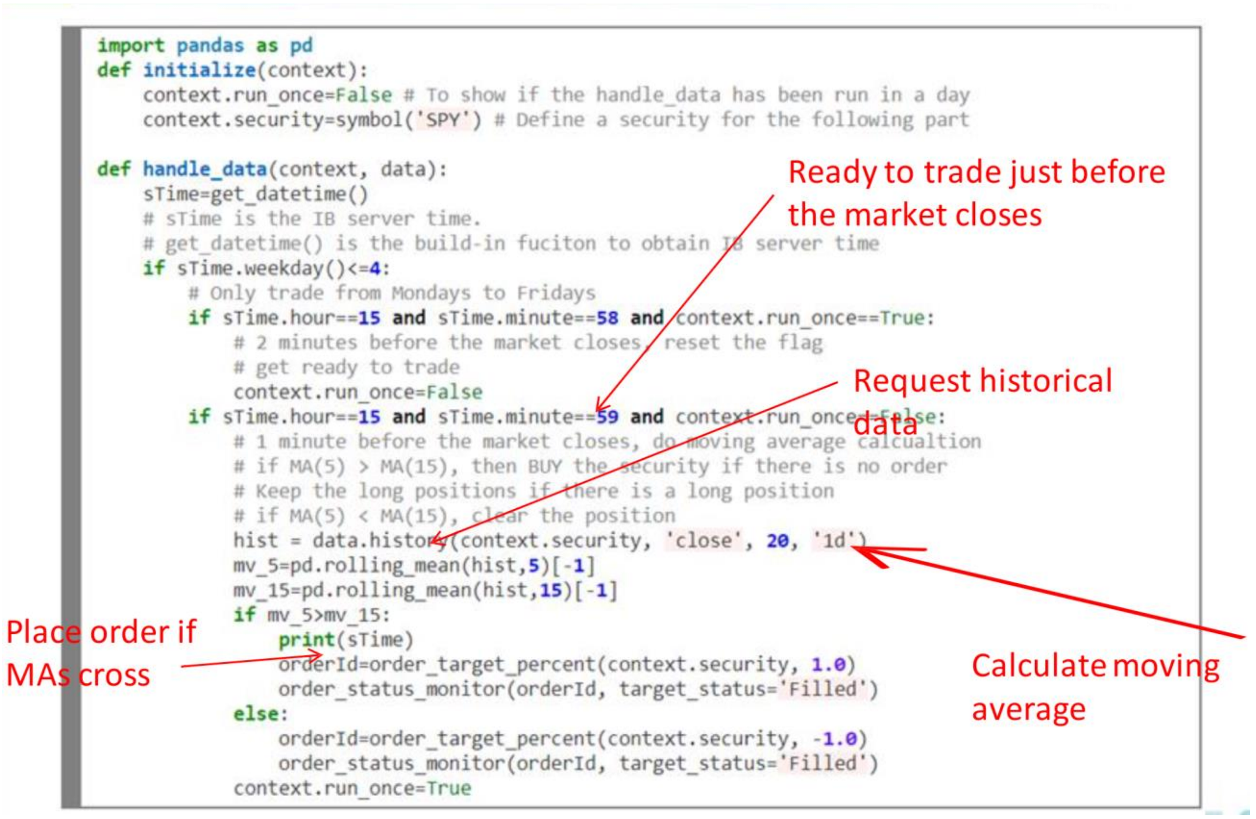
Smart routing

Interactive Brokers offers its customers a software product referred to as "Smart" order routing. Smart Routing software continually scans competing markets and automatically routes orders directly to the best ECN or market centre -- based on price and factors such as the availability of automatic order execution.

__Steps to build an algo trading platform__

- What contracts do you want to trade?
- When to make a trading decision? Frequency?
- What info is needed to make trading decisions?
- What strategy to place orders?
- When to Stop loss and take profits?
- When to exit?
- Continuously vs run-as-needed?

Example - Moving average crossover



```
import pandas as pd
def initialize(context):
    context.run_once=False # To show if the handle_data has been run in a day
    context.security=symbol('SPY') # Define a security for the following part

def handle_data(context, data):
    sTime=get_datetime()
    # sTime is the IB server time.
    # get_datetime() is the build-in fuciton to obtain IB server time
    if sTime.weekday()<=4:
        # Only trade from Mondays to Fridays
        if sTime.hour==15 and sTime.minute==58 and context.run_once==True:
            # 2 minutes before the market closes, reset the flag
            # get ready to trade
            context.run_once=False
        if sTime.hour==15 and sTime.minute==59 and context.run_once==False:
            # 1 minute before the market closes, do moving average calcualtion
            # if MA(5) > MA(15), then BUY the security if there is no order
            # Keep the long positions if there is a long position
            # if MA(5) < MA(15), clear the position
            hist = data.history(context.security, 'close', 20, '1d')
            mv_5=pd.rolling_mean(hist,5)[-1]
            mv_15=pd.rolling_mean(hist,15)[-1]
            if mv_5>mv_15:
                print(sTime)
                orderId=order_target_percent(context.security, 1.0)
                order_status_monitor(orderId, target_status='Filled')
            else:
                orderId=order_target_percent(context.security, -1.0)
                order_status_monitor(orderId, target_status='Filled')
            context.run_once=True
```

Improvements to Moving Average Crossover strategy

1. Handle holidays
2. Handling market early close
3. Send out an email notification
4. Handle market exceptions during live trading
5. Expansion: change it to a minute bar strategy

Roadmap: An idea to live trading

- Come up with an idea.
- Collect related data
- Build a mathematical model to test
- Draw graph for visualisation
- Program in IBridgePy
- Backtest the strategy
- Forward-test the strategy
- Live trade with a small portfolio
- Live trade with an entire portfolio

Example - Buy low, sell high.

If today's close price is lower than yesterday's close price, buy SPY using all cash. Otherwise, sell off all positions.

Demo available in the material provided: buy_low_sell_high_EPAT.ipynb

<u>Backtesting using historical data from IB through IBridgePy</u>

- Open TEST_ME.py
- Change accountCode
- Pick startTime and endTime of backtesting
- Backtest result analysis: two log files (BalanceLog, TransactionLog) are saved in the Output folder.
- tools/portfolio_sharpe_ratio.py
- Demo: TEST_ME_easiest_no_hist_ingest.py

- Declare histIngectionPlan, an IBridgePy reserved word
    - If handle_data is used, must ingest minute data at least
    - If schedule_function and/or request_historical_data is used, the timeframe of requested historical data should be adjusted accordingly
    - Hist data of all contracts in the code should be declared
    - IBridgePy will fetch hist data from IB server following histInjectionPlan
- Demo: TEST_ME_with_hist_ingest.py
- Demo: TEST_ME_by_local_file.py

<u>What are the considerations when implementing HFT?</u>

- Internet Latency
- High-performance CPU
- Enough memory
- Fast algorithm
- Stable tick-by-tick data
- Low transaction cost

<u>Local computer vs Cloud</u>

Local computer	Cloud computer
Low cost	Monthly charge
Self-managed utilities (power, internet, etc.)	99.99% uptime

Local development	Push code to remote
Large storage disk	Small storage (paid)
Full resources	Sharing computing (paid)
Full privacy	Full privacy

Additional resources

IBridgePy official website
IBridgePy documentation
IBridgePy YouTube channel
Quantra course: Automated Trading with IBridgePy using Interactive Brokers Platform