COMPUTING FOR FINANCE IN PYTHON

Lecture III – Student-led Projects overview and kick-offs

MAFS5360 – Summer, 2024

By Hongsong Chou, Ph.D.

AGENDA

- Overview of projects
- Project descriptions
- Project kick-offs

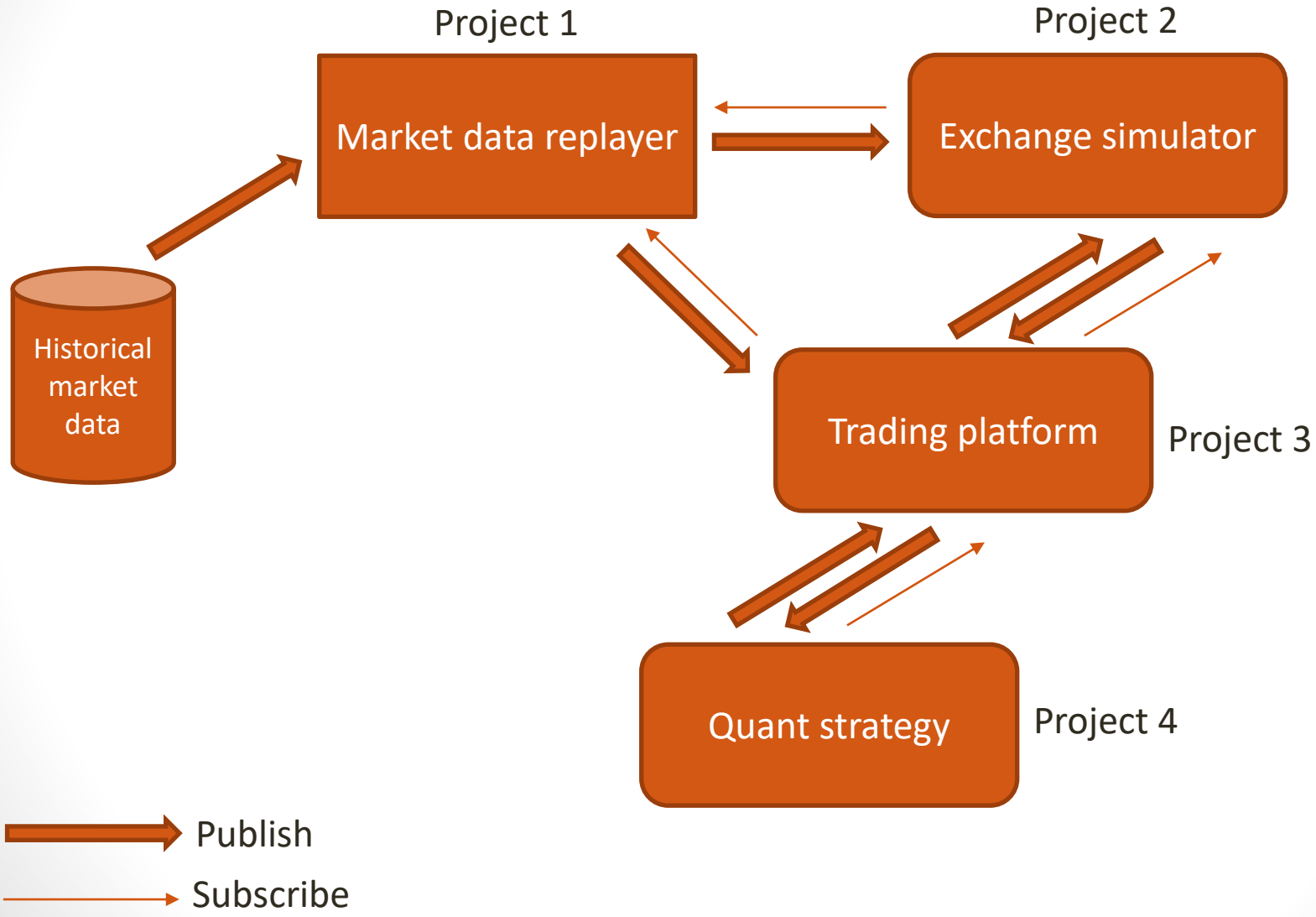
(TENTATIVE) OUTLINE OF THE COURSE

- Lecture 1 (June 20/25): Basic Python programming – Data Structure and object-oriented programming;
- Lecture 2 (June 27/July 2): Utilities in Python programming – File I/O and financial time series handling;
- Lecture 3 (July 4/9): Discussions on student-led projects, and project kick-offs;
- Lecture 4 (July 11/16): Machine learning through Python programming – packages and best practices;
- Lecture 5 (July 18/23): Parallel computing and performance optimization in Python programming;
- Office hour (July 25): In-class project supervision;
- Student-led presentations – Groups 1 & 2 (July 30);
- Student-led presentations – Group 3 (August 1).
- *Quiz 1 (June 25);*
- *Quiz 2 (July 2);*
- *Quiz 3 (July 9);*
- *Quiz 4 (July 16);*
- *Quiz 5 (July 23).*

AGENDA

- Overview of projects
- Project descriptions
- Project kick-offs

A QUANT TRADING SYSTEM IN PYTHON



AGENDA

- Overview of projects
- Project descriptions
- Project kick-offs

PROJECT 1

- Title: **Trade Data Processing and Re-publishing**
- Project main tasks:
  - Binary data file processing;
  - Subscription-based market data (re-)publishing;
  - Multi-processing for real-time data dissemination;
- Project dependence:
  - Upstream: none;
  - Downstream: projects 2, 3, 4;
- Project starting point:
  - Binary file processing code will be given;
  - A prototype for market data subscription will be given;
- Project focus:
  - How to design a “clock” to publish market data at the same speed as history;
  - How to ensure system stability when the number of subscribers increases.

PROJECT 2

- Title: **Exchange Simulator Prototyping**
- Project main tasks:
  - Establishment of an order book for an individual stock;
  - Updating order book status upon real-time market data;
  - Handling submitted orders;
  - Responding to order submitters with executions;
- Project dependence:
  - Upstream: projects 1 and 3;
  - Downstream: project 3;
- Project starting point:
  - A prototype code based on Python multi-processing utilities;
- Project focus:
  - How to design order books that update based on real-time market data from project 1 and cross orders submitted from project 3;
  - How to receive submitted orders from and send back executions to project 3, in real time.

PROJECT 3

- Title: **Quant Trading Platform Prototyping**
- Project main tasks:
  - Receiving market data and passing to subscribed strategies;
  - Receiving order submissions (if any) from strategies;
  - Utilities for sending a single stock and/or futures order to the exchange simulator and receiving corresponding execution or cancellation information from the simulator;
  - Passing executions or cancellation info to strategies;
- Project dependence:
  - Upstream: projects 1, 2 and 3;
  - Downstream: projects 2 and 3;
- Project starting point:
  - A design for data structure and prototype code for handling orders from strategies and executions from simulator;
- Project focus:
  - How to “synchronize” market data of two assets;
  - How to handle different types of orders (aggressive vs. passive).

PROJECT 4 (I)

- Title: **Single Stock and/or Single Stock Futures HFT Strategies**
- Project focus:
  - A single stock strategy, or a single stock futures strategy, or an arbitrage strategy can be designed using the other as forecasting signals;
  - For whichever strategy, the project will calculate trading signals based on real-time market data;
  - If deciding to trade, submit an order and receive executions;
  - Calculating and reporting strategy PnL and other metrics in real time and after market close;
- Project dependence:
  - Upstream: project 3;
  - Downstream: project 3;
- Project starting point:
  - A design for data structure for the strategy;
- Project focus:
  - How to build and back-testing a forecasting model;
  - How to communicate with trading platform;
  - How to calculate and update PnL and other strategy metrics.

PROJECT 4 (II)

- A few more comments on the quant strategy research:
  - Ten stock/stock futures pairs' market data are provided;
  - The tickers for stocks and corresponding futures are:

stocks	futures
"0050"	NYF
"2392"	GLF
"2498"	HCF
"2610"	DBF
"2618"	HSF
"3035"	IPF
"3264"	NEF
"3374"	QLF
"5347"	NLF
"6443"	RLF

  - For back-testing, three-month data are provided; you can use all the data except the last trading day to build the model;
  - For simulated trading, the date is the last trading day in both the data of the stock and the data of the futures (June 28, 2024 for most of the pairs); therefore, the model will be used "out-of-sample" for the last trading day.

COMMON DATA STRUCTURE

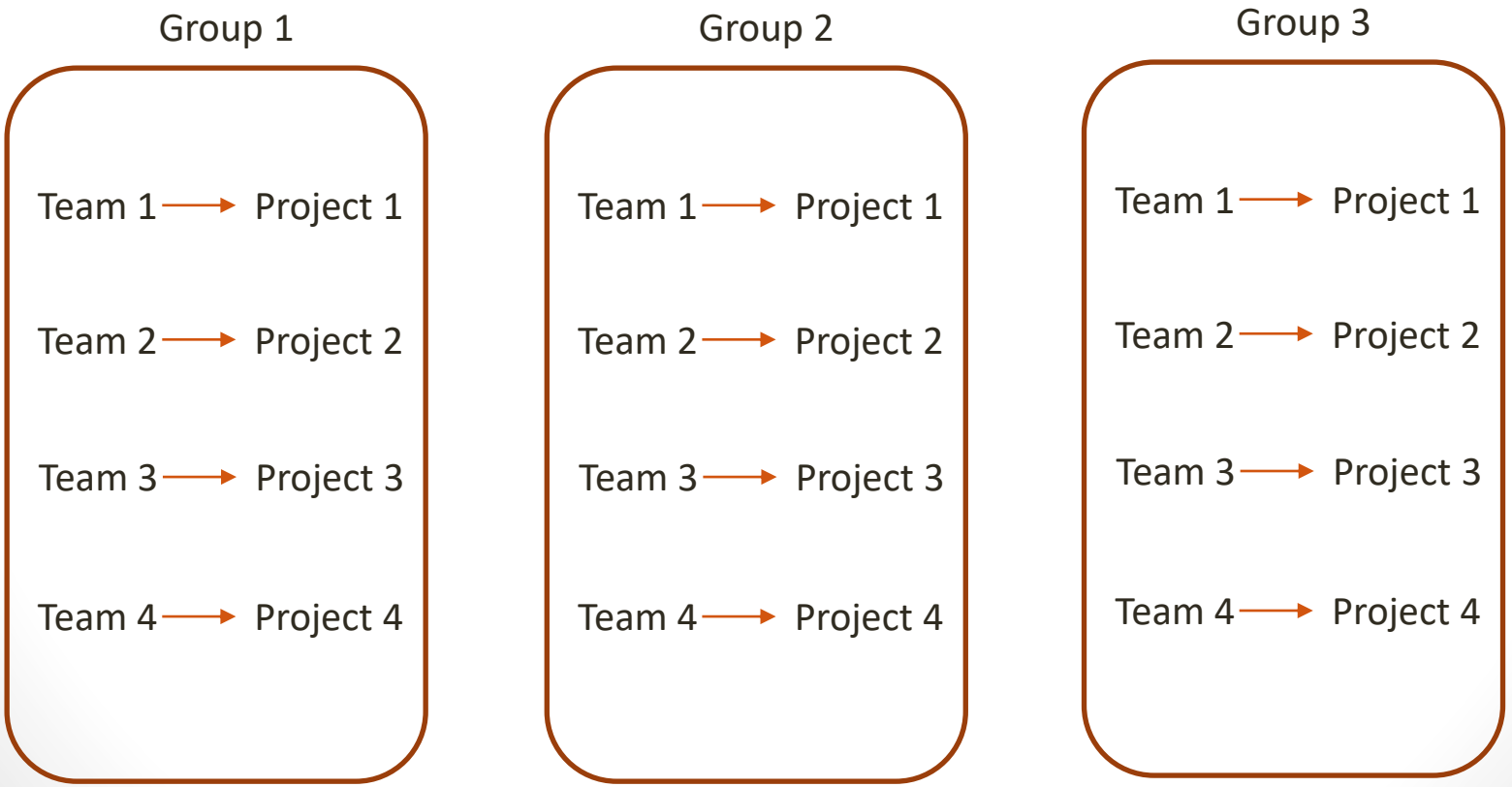
- Market data:
  - Order book snapshots (five levels only each on bid and ask);
  - Individual trades;
- Single asset order:
  - Basic info such as order ID, time stamp, order status, ticker, direction, price, size, etc.;
- Single asset execution:
  - Basic info such as execution ID, time stamp, ticker, price, size, etc.;
- Other key classes to be defined:
  - `MarketDataReplayer`;
  - `ExchangeSimulator`;
  - `TradingSystem`;
  - `TradingStrategy`;
- Communication messages:
  - A “market data subscriber” list held by “Market Data Replayer”;
  - A “trading system” list held by “Exchange Simulator”;
  - A “trading strategy” list held by “Trading System”.

AGENDA

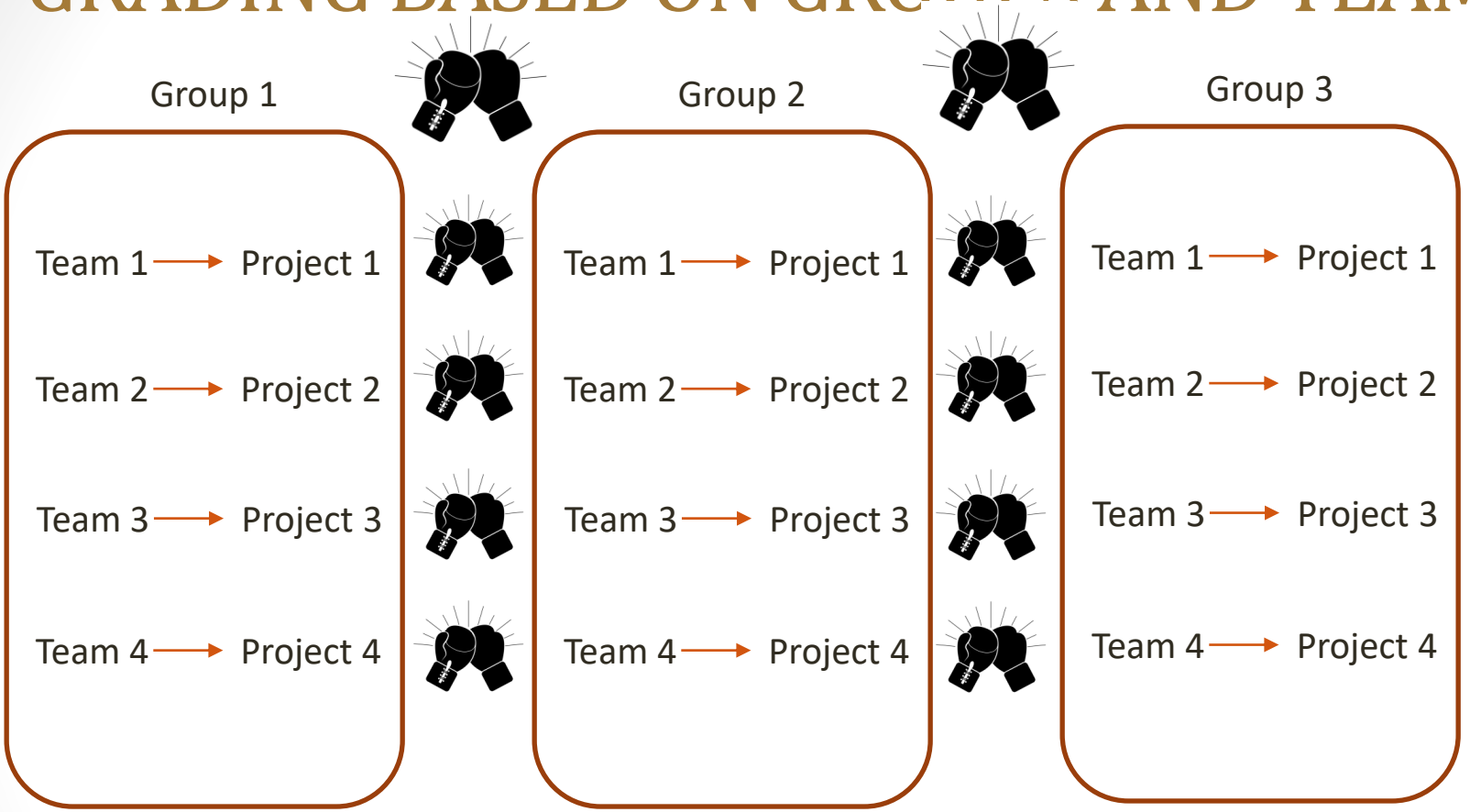
- Overview of projects
- Project descriptions
- Project kick-offs

GROUPING AND TEAMING-UP

- The whole class will be split to three groups, each of which will work on the same things: projects 1 to 4;
- For each group, members will be split roughly evenly to four teams, each of which will work on one of the four projects:



GRADING BASED ON GROUPS AND TEAMS



- Each group will get a grade from 0 to 100;
- Each team will get a grade from 0 to 100;
- Each team member's grade will be weighted sum (60%/40%) of group grade and team grade;
- Key criteria: (1) completeness; (2) code design and readability; (3) code performance (esp. speed).

Effective collaboration within group will be KEY!

TENTATIVE GROUPING

Group 1 (15 members)

AU, Man Yi Sigmund
DING, Boyang
DING, Guobin
DU, Ruimiao
HAN, Ru
JIANG, Xiaoyue
LAU, Wesley Kwokleung
QIAN, Yuhua
WANG, Chenyue
YANG, Tianhao
ZHANG, Zihan
BAO, Shijie
TAO, Xiyuan
HUANG, Yuxin
LI, Junyan

Group 2 (14 members)

CHEN, Qichuan
CHEN, Sihan
CUI, Hao
HUANG, Yuxuan
LAM, Hoi Tung
LI, Hanjin
LIN, Haoyu
LIU, Zongxuan
WEI, Wentao
WONG, Hou Tung
WONG, Sing Kit
WU, Qixin
YANG, Fan
ZHOU, Zihan

Group 3 (15 members)

ZENG, Chenxi
ZHANG, Zhiyun
ZHOU, Yufeng
ZHAO, Tianjun
HU, Junrong
LIN, Heyi
LU, Junyu
WU, Zilin
YANG, Kuo
GAO, Jie
ZHU, Wenjing
ZOU, Jie
WANG, Yiqi
HUANG, Xiaofeng
SINGH, Har Shwinder

- *As of July 9, 2024, we have 44 students registered for this course；*
- *Three group leaders will receive extra 5 to 10 bonus points for final grade;*
- *As of July 9, 2024, **Bao Shijie** and **Tao Xiyuan** will be co-group-leaders for Group 1, and **Singh Har Shwinder** will be group leader for Group 3.*

AGENDA

- Overview of projects
- Project descriptions
- Project kick-offs
- Extra - The scikit-learn package and machine learning in Python

SCIKIT-LEARN PACKAGE OVERVIEW (I)

- Rather powerful package that offers many utilities in data analysis, regression, classification, model selection, feature engineering, etc.;
- It is the place for many relatively simple and straightforward (not less powerful, though) machine learning approaches to solving problems;
- In many ways, scikit-learn package is comparable to R in terms of data and statistical analysis, yet faster (in my view) than R;
- Scikit-learn is built on top of numpy, scipy and other efficient and optimized scientific computing infrastructure;
- Unless one is considering large-scale machine learning tasks or sophisticated deep learning problems, scikit-learn should be the go-to machine learning package in Python; in many ways, it helps populate machine learning concepts among traditional data analytics communities.

SCIKIT-LEARN PACKAGE OVERVIEW (II)

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization.
— Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.
— Examples

Source: https://scikit-learn.org/stable/index.html

SCIKIT-LEARN REGRESSION EXAMPLE

- (See attached example: `Scikit_Example_ModelComparison.py`)

SCIKIT-LEARN MODEL SELECTION

- (See attached example: `Scikit_Example_FeatureSelection.py`)

THAT'S ALL FOR THIS LECTURE.

THANK YOU!