40.317 Lecture 1

Dr. Jon Freeman

19 May 2020

Slido Event #9269

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Agenda

- Course introduction and “administrivia”
- ~~We play a trading game!~~
  A demonstration only, due to HBL
- On building a financial system
  - Overall process
  - Key characteristics and qualities

High-Level Introduction

We will address two important questions, in the first and second half respectively:

- How should we design a financial system? That is, what are the principles a well-designed financial system should obey?
- How do we calculate the theoretical value of a financial derivative?

Financial Systems Design

A useful financial system is always an example of "programming in the large."

So our guidelines for building financial systems will overlap with those for programming in the large.

We will use a simple working example which we will build on from week to week.

Valuing Derivatives

By “derivatives” we will mean options rather than futures.

That is, we will focus on derivatives whose theoretical value is a function of volatility (among other things).

Lecturer Introductions

- For the first half: Jon Freeman, jon_freeman@sutd.edu.sg
  - Eastspring disclaimer
- For the second half: Chung-I Lu ("Lu"), lu.chung.i@gmail.com
- (There will be no Teaching Assistant.)

Recommended Texts

- Hilpisch, Yves. Python for Finance, 2nd Edition. O'Reilly, 2018.
- Newman, Mark. Computational Physics. CreateSpace, 2013 revised edition.
- Wilmott, Paul. Paul Wilmott on Quantitative Finance, 2nd Edition. Wiley, 2013. (Volume 3 only.)

Assessment Methods

- Regular assignments
- Design project
- Case study
  - Written prep, then an in-class discussion
- (No quizzes)
- ~~Final exam~~

  No final exam this year, due to HBL
- Class participation
  - Moderated Q&A via Slido

Assessment Weights

Item	Weight
Weekly assignments	40%
Design project	25%
Case study	10%
Class participation	25%
(Survey completion	*2%)*

HBL Procedures and Requests

- We will use Slido for entering questions and taking surveys.
  - There will be no chat window in Zoom.
- Please keep your camera on.
- Please email me your preferred groups of 3 to 4 students, for Zoom breakout rooms in later lectures.

Some Notes on Python

We will be using Python 3 via Anaconda (currently 3.7). Things to be aware of:

- Python 3 is not compatible with Python 2.
  - The Web has lots of example code which only works in Python 2.
- Whitespace matters in Python – another reason why code on a Web page might break after you copy and paste it.

Some Notes on Python, continued

- Regarding Anaconda:
  - You can install Anaconda2 and Anaconda3 side by side.
  - It comes with Spyder, a friendly IDE.
  - Also be aware of conda, its built-in package manager.

- Jupyter notebooks will be our means of communication in Weeks 8-13.

Your Expected Knowledge of Python

I expect you to have an intermediate knowledge of Python, which includes:

- Lists, sets, dictionaries, and tuples
- *Nested* combinations of the above
- *Comprehensions* on the above

```
[w.lower() for w in ('MiXed', 'caSe')]
{i for i in range(20) if i % 2 == 1}
{w: len(w) for w in ['various', 'words']}
```

Expected Python Knowledge, cont'd.

- Unpacking

```
first, *middle, last = ['A', 'B', 'C', 'D']
```

- with statements

```
with open(my_text_file, 'r') as f:
    for line in f:
        print(line)
```

- Basic familiarity with iterators

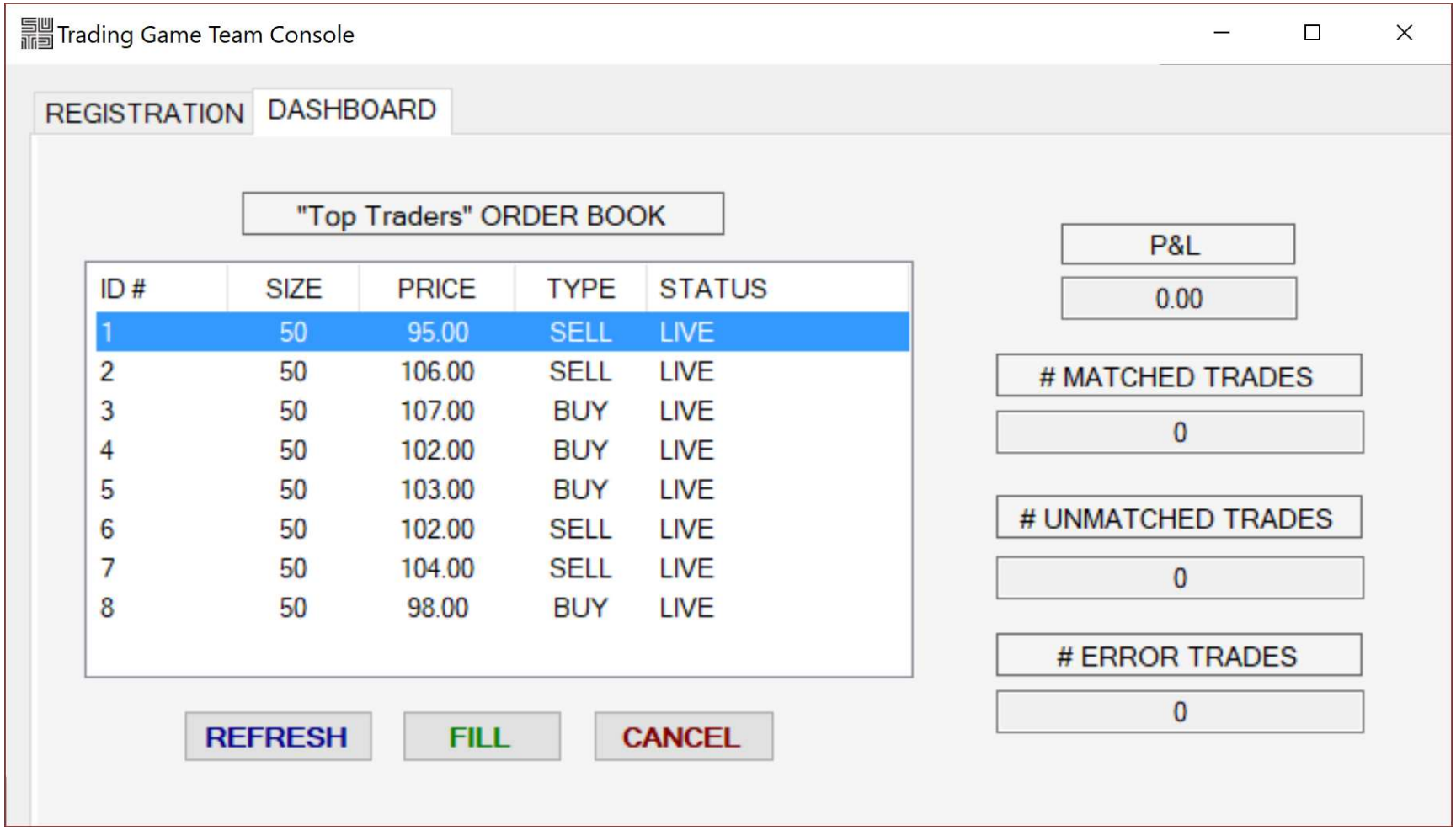
```
with open(my_text_file, 'r') as f:
    for line_num, line in enumerate(f):
        print(str(line_sum+1) + ': ' + line,
              end='')
```

Expected Python Knowledge, cont'd.

- Other basic aspects of "Pythonic" style, e.g. lazy stream processing:

```
import pyodbc
sql = 'select col1, col2, col3 from mytable'
# Not Pythonic:
with db_conn.cursor() as cursor:
    rows = cursor.execute(sql).fetchall()
    d = {r[0]: (r[1], r[2]) for r in rows}
# Pythonic:
with db_conn.cursor() as cursor:
    d = {r[0]: (r[1], r[2]) for r in
         cursor.execute(sql)}
```

Trading Game Demonstration



A BETTER WORLD BY DESIGN.

Why Demonstrate this Game?

After the first half of this class, you will know enough principles, tools, and techniques to build the same (type of) application yourself.

Building a Financial System

At a high level, how do we go about achieving a successful outcome?

{Correct, *Secure*, Fast, *Fair*, Friendly}

⇨ Trusted

⇨ Adopted

The three emphasised qualities are especially important for financial systems.

– (Notice blockchain helps with all three.)

Building a Financial System, cont'd.

The first half of this course will help you learn to build systems with these qualities.

The dimension of time is often ignored or discounted. The process is in fact like this:

{Correct, *Secure*, Fast, *Fair*, Friendly, Maintainable}

⇨ Trusted

⇨ Adopted

Building a Financial System, cont'd.

What are the key *characteristics* of a financial system, and the key *qualities* it therefore ought to have?

- First and foremost, high inherent complexity
  - ⇨ Well-modeled problem domain
  - ⇨ Well-architected overall
  - ⇨ Well-engineered internally
  - ⇨ [Well-governed]

Building a Financial System, cont'd.

- Over time, even higher complexity due to change requests ("feature creep")
  - ⇨ Maintainable
    - ⇨ Well-architected and well-engineered (again)
    - ⇨ [Well-governed, again]

- Transactional
  - ⇨ Persistent
  - ⇨ Atomic: each transaction must be processed completely or not at all

Building a Financial System, cont'd.

- Many independent, autonomous, often competing users/agents
  - ⇨ Asynchronous operation
    - ⇨ Correct and Fair
      (in Europe, fairness is required by law)
  - ⇨ Fast: both high throughput and low latency
  - ⇨ Secure
  - ⇨ Friendly: clear and helpful user interfaces

Building a Financial System, cont'd.

- Market-driven
  - ⇨ Asynchronous operation (again)
  - ⇨ Fast (again)

- Mission-critical – uptime must be close to 100%
  - ⇨ Resilient to component failure; to incorrect user inputs; and to missing, delayed, or incorrect market data
  - ⇨ Friendly error messages

Building a Financial System, cont'd.

- Heavily regulated
  - ⇨ Secure (again)
  - ⇨ Auditable
    - ⇨ Explainable behaviour, hence generally not suitable for deep learning

Thank you

A BETTER WORLD BY DESIGN.

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN