

AI FOR REAL-WORLD DATA AND AUTOMATED DECISIONS
SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING, SINGAPORE POLYTECHNIC

THE TRADING LEAGUE: PLAYER'S GUIDE (OPTIONAL)

Duration: About an hour to start; then whenever you like

Lecture: Topics 2 and 3

Name: _____

Admin No.: _____

Group: _____

Objectives

By the end of the lab, students will be able to

- Play an optional game that uses the loop, checks and rolling averages from Labs 2 and 3
- Walk the prompt, generate, verify, refine loop once, with a plan before the prompt
- Explain why code that runs is not code that is right, and what the three locked checks add
- Enter a class competition scored on prices nobody has seen

Activities

- Fetching five years of SPY prices into Colab and profiling them
- Scoring buy-and-hold and a ready-made crossover rule, with and without costs
- Changing one number in the crossover with Gemini: plan, prompt, generate, verify, refine
- Submitting the first entry; the scoreboard is public throughout

Review

- Labs 2 and 3 are complete

Equipment:

- Laptop with a web browser
- Google account (for Google Colab)
- The lab notebook: open the link on the next page (also on the course page on Brightspace)
- Files (the notebook fetches them itself): `Trading_League_STUB.ipynb` (linked from the scoreboard page), `ep0418.py`; prices come from Yahoo Finance through `yfinance`
- The class scoreboard: <https://stanleysp-eng.github.io/EP0418-labs/>

Open the notebook in Google Colab, then File > Save a copy in Drive (if asked, click Make a copy):

Trading_League_STUB.ipynb
https://colab.research.google.com/github/StanleySP-eng/EP0418-labs/blob/main/labs/Trading_League_STUB.ipynb

1 The game, in plain words

- No real money. The scorer pretends \$1 went into SPY, a fund that owns a slice of the 500 biggest American companies, five years ago, and replays history to see what your rule would have made of it. Every evening the rule chooses for tomorrow: keep the share (1), hold cash (0), or bet it falls (-1).
- A rule is a sentence that makes the choice for you. Example: keep the share while the price is above its 50-day average.
- You get five years of past prices to try rules on. The scoreboard shows how every rule did on them.
- Entries close at the end of Week 7. Then the market runs for four weeks nobody has seen, and every rule is scored on those. Results at Lab 7.
- The rule to beat is buy-and-hold: keep the share and do nothing. Each switch shrinks the pretend pot by 0.1%, like a broker's fee.
- Enter as often as you like; the latest entry counts. Nothing is graded. Two podiums: best hidden four weeks, best write-up.
- Paper trading only. Nothing here is investment advice.

2 Where the labs appear

Idea from the labs	Where you meet it in the game
Three ways data arrives (CSV, JSON, live API)	The first cell: yfinance calls a live API, gets JSON, saves a CSV.
Five questions to ask of new data	Task 1, one line each. The strange one is the gap between dates.
“It ran without an error” is the weakest evidence	Task 2: the crossover runs fine both times; costs change the answer. Task 3: the three checks.
Plan first	Task 3 starts by asking Gemini for a plan in plain words, no code.
The loop: prompt, generate, verify, refine	Task 3, once, on one changed number. Task 4 later, on your own idea.
A good prompt: data, goal, constraint, check	Every prompt in the notebook; the constraint is always “no later rows”.
Refine means saying exactly what was wrong	Task 6, the write-up: the mistake Gemini made and how you caught it.

3 Your first sitting (about an hour)

Task	What you do	What you write down
1 Look at the data (15 min)	Run the five profiling questions on SPY.	The strangest number and why it is there.
2 Doing nothing, and costs	Score buy-and-hold. Run the ready-made crossover with and without costs, slow and jumpy.	Buy-and-hold's Sharpe ratio; the trades each crossover made; how many percentage points costs ate from the jumpy one's return.
3 Your first entry	Ask Gemini for a plan, then for the code with one number changed. Click Accept and run on Gemini's green cell, run the check cell (three PASS lines), run the score cell, then Submit.	Your name on the scoreboard.

The Submit cell works from anywhere, any time before entries close.

4 The weeks after

When	Task	Why then
Any time	4 Your own rule from the ideas table	shift() and rolling averages from Lab 3 are the parts rules are made of.
After Lab 5	Try a volatility filter as Task 4	Lab 5 teaches z-scores.
After Lab 6	5 The rehearsal, once; 6 the write-up	Lab 6 teaches how to beat a baseline honestly.
End of Week 7	Last Submit	Entries close; the exact time is on the scoreboard.
Lab 7	Both boards on the screen	Who moved, which way, and why.

5 Ideas for Task 4

Each is honest, each has a known way of losing, and each is a sentence Gemini can turn into code. Pick one you can explain to a friend.

Idea	The rule	Where it loses
Trend	Hold when the price is above its moving average; cash below.	Choppy markets: it buys every wobble's top and sells its bottom.
Breakout	Hold after the close beats the highest close of the last n days.	False breakouts; it trades often, and costs eat it.
Mean reversion	Buy after a fall of more than x% in a week; sell after a rise.	A real crash: it buys all the way down.
Volatility filter	Hold, but go to cash when recent daily moves are unusually large.	It sits out the rebound after the fear.

6 The three traps, and the check for each

- Peeking at the future (lookahead). A rule that uses tomorrow's price looks wonderful on the past and is useless tomorrow. In code it looks innocent: `shift(-1)`, a centred rolling window, a whole-file average. `no_lookahead` chops the end off the file and reruns your rule; if an earlier choice changed, you peeked.
- Learning the past (overfitting). Any rule can be tuned until it fits five years of history. The rehearsal (Task 5) scores it on months it was not tuned on. Run it once and write the numbers down.
- Forgetting costs. Every change of mind costs 0.1%. `trades_at_least` makes sure your rule changes its mind at least four times; Task 2 shows what too many changes do. `position_is_valid` simply says every day got a 1, 0 or -1.

7 How you are scored

The position you hold at the close of one day earns the next day's move. Each change of position costs 0.1% of what moved. The five-year board ranks by the Sharpe ratio after costs: return per unit of bumpiness, roughly, where above 1 is good and above 3 is suspicious. The hidden four weeks rank by plain return after costs. Buy-and-hold is an entrant on both. In the notebook, `compare_to_baseline` says whether you beat it on the five years; a NOTE there costs nothing. An entry that fails a check is not sent; fix it and run Submit again.

8 The write-up (Task 6, two answers)

- What your rule does and when it loses, in words a friend outside the course would understand:

- One mistake Gemini made, how you caught it, what you changed:

9 Read before you run

The code below was written by an AI assistant. It runs without an error and it is wrong. An assistant was asked for a rule that holds SPY on days when the price is rising and goes to cash otherwise. It wrote the code below, and the backtest over five years showed a Sharpe ratio above 6.

```
def peeking(prices):
    return (prices["Close"].shift(-1) > prices["Close"]).astype(int)
print("Sharpe ratio:", round(backtest.run(prices,
    peeking(prices)).sharpe, 2))
check.reset(); check.no_lookahead(peeking, prices); check.report()
```

AI for Real-World Data and Automated Decisions

Exercise

- Question 1. Do not run it. Which day's close is each row comparing with which? Could a trader at the close of day t have known it?

- Question 2. Predict the Sharpe ratio it prints (above or below 3?) and whether `no_lookahead` passes. At step 3 you run it and compare. It uses its own name, `peeking`, so your own `strategy` and `entry` are untouched.

- Copy the cell (click inside it, Ctrl+A, Ctrl+C), paste it into a new Gemini chat and add: Explain this code in plain words. Did Gemini's explanation mention the fault? Write yes or no, and the sentence that shows it.

- Delete the "# " at the start of every line below the first line (the first line stays), run the cell, and write down whether you were right.

10 Quiz yourself before you leave

The checks say whether the code is right. They do not say whether you know what it did. Open Gemini, click the three dots at the top right of the Gemini panel, then New chat, then Yes, paste this and answer in your own words, without looking back until it has marked you. Five minutes.

“Rules: Ask one short question at a time, in plain words. Take the examples from the code and outputs in this notebook. Wait for my answer. Then say in one sentence whether I am right, and why. Do not give the next question until I answer. Task: Quiz me on this lab: five questions on these ideas: the five steps of the loop (plan, prompt, generate, verify, refine) and the step most people skip; why 'it ran without an error' is weak evidence, and what the three checks add; what buy-and-hold is and why it is the rule to beat; what lookahead is, in plain words, and how the cut-the-end-off check catches it; why a rule that trades often loses to costs. After the fifth, give my score out of 5 and the one idea I should revise, in one line.”

AI for Real-World Data and Automated Decisions

- Score out of 5, and the one idea to revise in your own words. Your tutor asks for these in the plenary.

11 After your first sitting

- Your name is on the scoreboard.
- What buy-and-hold caught that my rule missed (Task 3):

- Rehearsal numbers, later (Task 5): tuned-on Sharpe, scored-on Sharpe, last month's return:

- Mistake caught: what I expected · what was there · how I found it

- Three numbers written down: the strangest number in the data, buy-and-hold's Sharpe ratio, what costs ate from the jumpy crossover.

12 Questions

- Your rule's Sharpe ratio is 2.4 on the five years and 0.3 on the six-month rehearsal. What does that most likely mean, and which number should you trust?

- Two rules have the same return over five years; one changed its mind 11 times, the other 71. Which has the higher Sharpe ratio after costs, and why?

AI for Real-World Data and Automated Decisions

--