MOL518 Spring 2026

Homework 1

Due: Friday February 6, 2026

For this assignment, you will generate and then submit a Jupyter notebook on Canvas. You may use generative AI tools as per the syllabus unless otherwise stated in the problem.

First, make a new Jupyter notebook named `MOL518_HW1_<YourLastName>.ipynb`*. All problems will be answered in this notebook. For each problem, start with a Markdown cell using Problem XX as the title, i.e. "# Problem 1".*

Problem 1: Practice with Markdown

Figure 1 is the preface page for Lubert Stryer's classic textbook *Biochemistry*. Recreate this page in a Markdown cell in your notebook. The image at the bottom of the page is provided as `StryerPreface_liverketone.png`. *Do not use AI for this problem.*

PREFACE

The more we learn, the more we discover connections threading through our biochemical world. In writing the sixth edition, we have made every effort to present these connections in a way that will help first-time students of biochemistry understand the subject and how very relevant it is to their lives.

Emphasis on Physiological Relevance

Biochemistry is returning to its roots to renew the study of its role in physiology, with the tools of molecular biology and the information gained from gene sequencing in hand. In the sixth edition, we emphasize that an understanding of biochemical pathways is the underpinning for an understanding of physiological systems. Biochemical pathways make more sense to students when they understand how these pathways relate to the physiology of familiar activities such as digestion, respiration, and exercise. In this edition, particularly in the chapters on metabolism, we have taken several steps to ensure that students have a view of the bigger picture:

- Discussions of metabolic regulation emphasize the **everyday conditions** that determine regulation: exercise versus rest; fed versus fasting.
- New **pathway-integration figures** show how multiple pathways work together under a specific condition, such as during a fast.
- More **physiologically relevant examples** have been added throughout the book.

This physiological perspective is also evident in the new chapter on drug development. The use of a foreign compound to inhibit a specific enzyme sometimes has surprising physiological consequences that reveal new physiological principles.

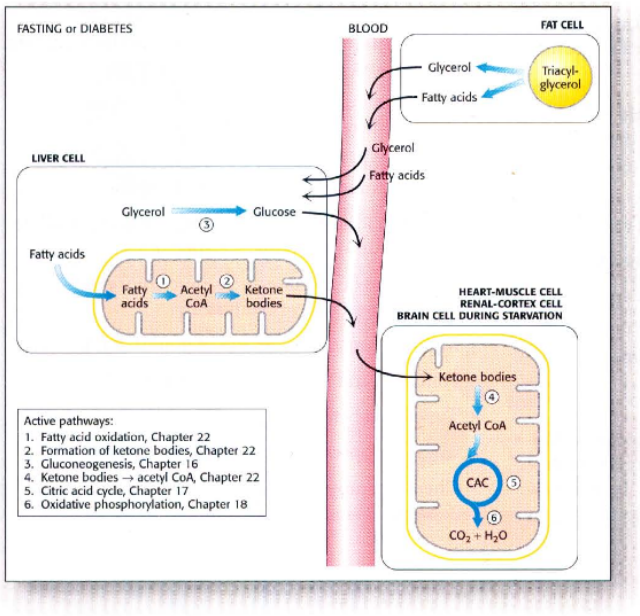

Figure 1: Stryer preface page.

Problem 2: Wet-Lab Arithmetic

You are planning a simple infection in a single well of a plate. You have a tube of virus stock at a known titer, and you want to prepare an inoculum at a target MOI in a specified final volume.

In your notebook, write your work as if you are preparing a mini "protocol note" for yourself: define variables clearly, keep track of units, and print a short summary at the end.

Do not use `for` loops or `if` statements in this problem.

a) In a single code cell, define the following variables (with these exact names) in Python:

- `cells` = 250000 (cells)
- `moi` = 0.2 (dimensionless)
- `titer` = 2.0e8 (infectious units per mL)
- `inoc_vol_uL` = 500 (μL total inoculation volume)

b) Compute the number of infectious units needed for this infection (call it `iu_needed`). Briefly explain (in a Markdown sentence) what MOI means in words.

c) Convert the titer to infectious units per μL, then compute the volume of virus stock to add (in μL) to deliver `iu_needed`.

d) Compute the remaining volume (in μL) that must be filled with media so the final inoculum volume is exactly `inoc_vol_uL`.

e) Print a short, readable summary that includes: `cells`, `moi`, `iu_needed`, virus volume (μL), media volume (μL).

Problem 3: CSV "Sanity Check" Report (No Loops)

(15 points)

You ran a plate reader growth experiment and saved the results as a CSV file. Before doing any serious analysis, you want to do a fast "sanity check" to confirm the file loaded correctly, the sampling looks consistent, and the values are in a reasonable range.

Your goal is to produce a short, human-readable report in your notebook: show what you loaded, show a few key numbers, and make it easy for a reader to understand what the dataset looks like.

Do not use `for` loops or `if` statements in this problem.

a) Load `Lecture_2/data/growth_curve1.csv` as a 2D NumPy array called `data`. (Include the exact code you used.)

b) Extract `time` (seconds) and `od` (unitless) as 1D arrays. In one Markdown sentence, state what each variable represents.

c) Print the shape of `data`. Then print the first row and the last row, and briefly interpret what those rows mean.

d) Compute and print the sampling interval in seconds: `dt_sec = time[1] - time[0]`. In one sentence, explain what it would suggest if `dt_sec` changed over time.

e) Compute and print the total duration of the experiment in hours.

f) Compute and print the following summary quantities, and label each one clearly in the output:

- absolute OD change: `od[-1] - od[0]`
- fold change in OD: `od[-1] / od[0]`

Problem 4: Create, Transform, and Save a Synthetic Dataset (No Loops)

(15 points)

In real life, your first dataset might be messy: missing points, noise, odd formatting, or surprising values. In this problem, you will generate a tiny *synthetic* dataset where you control the structure and the meaning of each column.

Think of this as a practice run for the workflow: create arrays, combine them into a table, write to disk, reload, and confirm nothing got scrambled.

Do not use `for` loops or `if` statements in this problem.

a) Create a NumPy array called `time_hr` with these time points (hours): `[0, 1, 2, 4, 8, 16]`. Print it.

b) Define `od0 = 0.08` and `doubling_time_hr = 1.5`. In one Markdown sentence, explain (conceptually) what "doubling time" means.

c) Compute `od` using: `od = od0 * 2**(time_hr/doubling_time_hr)`. Print `od`.

d) Create `od_norm` by normalizing `od` to its first value. Print `od_norm`.

e) Use `np.column_stack` to make a 2D array (a mini table) with columns `time_hr`, `od`, `od_norm`. Print the resulting array and its shape.

f) Save the result to `Homework_1/simulated_growth.csv` using `np.savetxt(..., delimiter=",")`. (This path is relative to the repository root.)

g) Reload the file you saved into `data2`. Print `data2.shape` and the first 3 rows, and confirm (in a sentence) that the reloaded numbers match what you expect.

Problem 5: Lists — Sample Tracking and Lane Batching (No Loops)

(15 points)

You are organizing samples for a downstream assay (e.g., loading lanes on a gel, assigning sequencing lanes, or planning a plate layout). In practice, a lot of mistakes come from simple bookkeeping: missing a sample, duplicating a label, or mixing the order.

In this problem, you will treat a Python list as your "sample sheet" and practice simple, reliable operations: counting, appending, sorting, and slicing into batches.

Do not use `for` loops or `if` statements in this problem.

a) Create a Python list called `samples` with the following sample IDs (strings), in this exact order:

```
samples = [
    "WT_A", "WT_B", "WT_C", "KO1_A", "KO1_B", "KO1_C",
    "KO2_A", "KO2_B", "KO2_C", "Rescue_A", "Rescue_B", "Rescue_C"
]
```

b) Print the number of samples using `len(samples)` and briefly interpret what that number means in this context.

c) Append two new samples to the end of the list: `"Blank"` and `"PositiveCtrl"`. Print the updated list.

d) Create a new list called `samples_sorted` that is an alphabetically sorted version of `samples` (do not change the original list). Print both lists to demonstrate that only the new one was sorted.

e) Imagine you can only process 5 samples per "lane" (or batch). Create three new lists by slicing `samples_sorted`:

- `lane1` = first 5 samples
- `lane2` = next 5 samples
- `lane3` = remaining samples

f) Print `lane1`, `lane2`, and `lane3`. In a short Markdown note, explain why slicing is a safer approach than manually retyping sample IDs.