Computational Carpentry Project

Deadline 26 September 2025

As a project for this module, we ask you to create a Jupyter notebook and answer the following questions:

Part A – Data structures and functions (30 points)

1. Load the “periodic_table.csv” data file using pandas (0 points)
2. Create a python dictionary where the keys are elements symbol and the values are atomic masses. (5 points)
3. Write a function that takes chemical formulas like “H_2O” and “$C_6H_{12}O_6$” and return their molecular mass. (10 points)
4. Extend the molecular mass calculator to handle parentheses in formulas (e.g, “$Ca(OH)_2$”) (10 points). (If you use recursive programming you will have 5 bonus points)
5. Extend the molecular mass calculator so it can handle coordinated (hydration) water (e.g. “$CuSO_4.5H_2O$”) (5 points)

Part B - Stoichiometry and reaction balancing (30 points)

1. **Reaction Balancer Function (20 points)**

- Write a function `balance_reaction(reactants, products)` that:
  - Takes as input:
    - **`reactants`**: list of chemical formulas for reactants (e.g., [“H_2”, “O_2”])
    - **`products`**: list of chemical formulas for products (e.g., [“H_2O”])
  - Returns: a **list of stoichiometric coefficients** that balances the reaction.
  - You don’t need to consider parenthesis or “.” in the formula for this problem
  - **Hint:** You can represent the reaction as a system of linear equations based on the number of atoms of each element. Look up the term Singular Value Decomposition, you can use this link.
  - Test it for 3 reactions of your choice.

2. **Mass Conservation Check (10 points)**

- Using the molecular mass calculator from **Part A**, compute the total mass of reactants and products with the coefficients found. Achieve this by writing a function which takes a dictionary of molecules and their coefficient and return a True or False.
- Verify that the total mass is conserved. Print a message indicating whether the reaction is balanced.

Part C - Simulation / Modeling (40 points)

1. **Monte Carlo π estimation (intro exercise)**

- Write a function `estimate_pi(N)` that: (10 points)

Shoots N random points uniformly inside a square of side 2 (x and y from -1 to 1).
Counts how many points fall inside the unit circle (distance ≤ 1 from origin).
Using that, calculate π.
Hint: What is the probability of a randomly placed point being in the circle?

- Plot the convergence of your estimate versus N to visualize accuracy improvement**.** (5 points)

2. **Chemistry-inspired Monte Carlo**

- Simulate M random molecular collisions in a box.
- Assign random energies to each collision (e.g., uniform or normal distribution).
- Count how many collisions exceed a given threshold energy.
- Interpret the fraction as an estimate of **reaction probability**.

Notes

In your notebook, use the markdown cells to describe the steps you perform and what you learned from your analysis. This is known as literate programming.

Be sure to upload using the correct naming convention, where # is the letter of your group: (Group#_Block1_project.ipynb) or (Group#_Block1_project.zip)
For this project you can upload a single Jupyter notebook containing all the explanation of your code. If you want to submit more than one file, make a zip file and upload that. Be sure to include **ALL** your teammate names in the Jupyter Notebook.
Do not forget to fill in the questionnaire of contribution. (If NOT filled, we assume equal contribution).

For those who want to benefit form the project:

These days it is not so important to know syntax but it is more than ever critical to develop and shape **how** you think. Therefore it is not discouraged and even **recommended to use** AI for this project. If you use ChatGPT or any other AI tool for this project use it **wisely**. Try to understand the concepts, and first try one or two things yourself. You even get more credit (up to 5 points bonus) to include your failed experiments in the notebook and argue why they didn't work.
For example don't copy and paste the full question in the prompt and get the answer, a good prompt can be: "Having two lists A and B, how can I combine them element-wise: A = [1, 2, 3] and B = ['a', 'b', 'c'], the output should be [(1, 'a'), (2, 'b'), (3, 'c')]. Knowing the syntax to do this (it is zip(A, B) by the way) is less relevant than knowing you have to do this in part A question 1.
Your evaluation is not taking a hit by not being 100% correct, if you are showing us how much you are developing the way you think.