Merck Coding Challenge

Instructions: Please do all questions. I recognize that this problem set will require significant effort, so I promise to read and give feedback on all reasonably complete submissions. I also welcome thoughtful questions. Please email me if they come up (eugene.kwan@merck.com) so we can have a dialog. I can also provide solutions when you are finished. ~~The challenge is closing! The deadline to submit a challenge is July 26th, 11:59 pm EST.~~ The challenge is CLOSED! Thanks to all who submitted solutions. I should have provided feedback by now – please open a GitHub issue in your repo if I missed you. Thanks!

How to Submit:

- Please create a private GitHub repository.
- Please share the repository with **ekwan** (and not eugene.kwan@merck.com)
- Please place your resume in the repository (so I know who you are).

Grading: Documentation is critical! Please comment, describe, and explain everything you can. I reward correctness, clear documentation, and good questions/reasoning. If you're able to complete both questions correctly, I'll ask you to participate in a 1:1 interview with me.

---

1. PyPlate is a Python package for designing high-throughput chemistry and biology experiments. Suppose that you need to screen conditions for 12 cross-coupling reactions of the form:

$$A_i + B_i \rightarrow C_i$$

where A_i and B_i are starting materials, C_i is a product, and i runs from 1…12. For each reaction, let A_i be the limiting reagent (0.1 mmol), add 1.1 equivalents of B_i, 10 mol% $Pd(OAc)_2$, and 15 mol% of ligand (see below). (Equivalents are relative to limiting reagent.)

(a) We'd like to screen a common set of 2 temperatures (60 °C and 80 °C), 4 solvents (toluene, glyme, TBME, and dichloroethane), and 3 ligands (XPhos, SPhos, and dppf). Please write a PyPlate Recipe that implements the above experimental design. Use a total reaction volume of 200 uL and 96 well plates with a maximum volume of 500 uL. Use a random number generator with a fixed seed to set the molecular weights of A_i and B_i (set between 100 and 500 g/mol). Use the real molecular weights of everything else.

Of course, multiple experimental designs are possible here. Can you design a Recipe that is not only easy to code but also practical to carry out in the lab? What sort of practical considerations are there? Please provide your answer as a clearly documented Jupyter notebook. You should graphically illustrate your design. These diagrams should not show the details of precisely what is in each well, but rather *explain* the concept behind the design.

Note that PyPlate does not currently have a feature to specify temperatures. You will have to keep track of that manually. As you go through this exercise, please read through the PyPlate documentation. If you notice anything that could be improved (and there a lot of possibilities), please email me. I will award significant extra credit for thoughtful, chemically sensible and computationally reasonable proposals along these lines.

(b) As you can see, PyPlate currently allows *absolute* quantities like 1 mmol to be specified, but not *relative* quantities like 1.1 equivalents of B_i. In fact, one could imagine that it would be ideal to specify something like "10 mol% ligand."

Without writing any actual code, please explain how you would modify PyPlate to incorporate this feature. Which parts of the API would have to change? How would you ensure that the relative quantities are logically and chemically reasonable? Can you write docstrings for the new or modified functions?

2. Chromatography is frequently used to determine the outcome of experiments. However, most chromatography instrument manufacturers provide data in proprietary data formats. We've developed the Rainbow package to unlock these files and we want to know whether you can extend Rainbow.

Here are three folders with artificially generated and encoded chromatography data:

(a) **pear** challenge (easy): time vs. intensity data
(b) **scale** challenge (intermediate): time vs. wavelength vs. absorbance data
(c) **sixtysix** (hard): time vs. mass vs. intensity data

In each folder, you will find a `sample/` subfolder and `problemX` subfolders (where X=1,2,3). The `sample` subfolder contains a matched binary/csv pair. You should examine this pair with a hex editor (or any other tool of your choice) to determine its binary organization. The rest of the folders contain only binaries. Your decoding script should run on these files. I will check that the csv output matches what is expected.

Note: your answers should *not* include any hard-coded magic numbers (other than the lengths of headers, chunks, footers, etc.)

Please provide a concise and clear explanation for each file structure in markdown format. What is the format of the header, data, and footer? I suggest writing a couple paragraphs to accompany a table like this:

Location	Length (bytes)	Endianess	format	Value
0x180	4	big	uint	time[0] (ms)
0x184	4	little	uint	intensity[0]
…				

Please document your code clearly with comments and docstrings. Please provide your answer as one `.py` file per problem (so, one for pear, one for scale, and one for sixtysix). Please provide the decoded `.csv` files so I can check them against the expected results. Place one decoded csv file per problem directory like this: `pear/problem1/pear.csv`.