Merck Coding Challenge

6/21/24: Challenge updated! It is ready to work on.

**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.

**How to Submit:**

* Please create a private GitHub repository.
* Please share the repository with **ekwan** (and not [eugene.kwan@merck.com](mailto: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.

1. PyPlate is a [Python package](https://pyplate-hte.readthedocs.io/en/working/index.html) for designing high-throughput chemistry and biology experiments. Suppose that you need to screen conditions for 12 cross-coupling reactions of the form:

Ai + Bi → Ci

where Ai and Bi are starting materials, Ci is a product, and i runs from 1…12. For each reaction, let Ai be the limiting reagent (0.1 mmol), add 1.1 equivalents of Bi, 10 mol% Pd(OAc)2, and 15 mol% of ligand (see below).

(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 Ai and Bi. Use the real molecular weights of everything else.

Your Recipe should consider how easy and simple it would be for an experimenter to carry out the Recipe. Please provide your answer as a clearly documented Jupyter notebook that contains clear documentation. Please use PyPlate [visualizations](https://pyplate-hte.readthedocs.io/en/working/users_guide/visualizations.html) to illustrate your designed layout (via the shaded DataFrame method).

(b) As you can see, PyPlate does not currently have a feature that allows the user to specify 1.1 equivalents of Bi. One remedy is to introduce the concept of *tags*. Within the context of a particular Recipe, each Substance could be attached to any number of string labels (like “A”, “B”, or “ligand”). Then, when transferring Substances to Containers or Plates, we might specify relative quantities like “1.1 \* A”.

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? Please copy and paste the appropriate docstrings into a markdown file and modify them to explain the new API behavior. How would you ensure that the quantities are physically reasonable? What other constraints might there be?

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](https://github.com/evanyeyeye/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:

1. **pear** challenge (easy): time vs. intensity data
2. **scale** challenge (intermediate): time vs. wavelength vs. absorbance data
3. **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.

**Please provide a concise and clear explanation for (ii) 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 for (iii) clearly with comments and docstrings. Please provide your answer for (iii) 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.