Merck Coding Challenge

Please submit your answers by creating a private GitHub repository and adding me (Eugene Kwan, ekwan) as a collaborator. Please be sure to put your name in the README and include your resume in the repository.

I recognize that these questions will take some significant effort. I pay a lot of attention to your documentation. I also give credit for asking good questions (eugene.kwan@merck.com) – please do reach out. I'm including some optional questions so that you can have more context.

There are three mandatory sections (and two optional ones) that you should spend no more than one day on in total. I'll read every submission, give feedback, and provide solutions for the decoding questions.

---

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).

(a) **[*optional*]** 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. 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 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 B_i. 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 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 – ***optional***, for bonus points): time vs. mass vs. intensity data

In each folder, there are some binary files, along with a .csv file containing the decoded data. Your job is to:

(i) examine the raw data
(ii) determine how the data are stored in binary form
(iii) write a Python program that converts the binary data into csv form (to parallel the provided csv)

Please provide a concise and clear explanation for (ii) in markdown format. What is the format of the header, data, and footer? For example, "the header is made of one little endian 64 bit integer denoting the number of data points (n), followed by n big endian floats, followed by…"

Please document your code for (iii) clearly with comments and docstrings. Please provide your answer for (iii) as one .py file per dataset. When these files are run, they should decode the data and generate the appropriate .csv file.