Coding Exercise #2

OBJECTIVES

After completing this week's exercises, you will know how to:

- Use functions
- Use branching (if statements)
- Add packages to the Eclipse library
- Import packages into modules

Tasks

1. Open Eclipse and find the **sample-code-solutions** repository under the Project Package Explorer window and **pull** from this repository. This should make the file "background_reading_2_2022.py" appear. Use this as you go through the Background Reading pdf, found on D2L under the Coding Exercise 2 folder, to review the basics that are relevant to this Coding Exercise.

2. Locate your **last name-first name** repository in Eclipse and **pull**. It is good practice to periodically pull from both the **sample-code-solutions** and **last name-first name** repos**,** especially before each commit, to avoid merge conflict errors. I.e., if the instructor or TA have modified one of your submitted files and you have not pulled from the repo, once you do try to commit, you will be sending a version of files to GitHub, (e.g., a previous Coding Exercise), which will not match with our annotated version that we also pushed to GH. When in doubt, pull first.

3. Create a new .py file (the guides from Coding Exercise 1 may be helpful if you've forgotten how to do this) **in your last name-first name repository**. Name your file **coding_exercise_2.** The .py extension will automatically appear.

4. Review the Background Reading documents found in **Coding Exercises > Code Exercise 2** on D2L. The text provided in the Background Reading is meant to gently guide you through problems similar to what your deliverables are for this exercise. There are no deliverables from the reading itself.

5. Write a code that has the following features:

    - Select any five of your favourite isotopes (these must be real isotopes, not just random combinations of protons and neutrons). Use the **periodictable** package to create unique variables for each of your isotopes (see Background Reading #2). Make a list of these variables. These isotope objects will already have stored desirable properties such as .number (atomic number) and .isotope (atomic mass number).

- Create a function that takes the input of the atomic mass number (A) and the number of protons (Z) in a nucleus then calculates the number of neutrons (N) and the ratio of neutrons to protons, N/Z, rounded to 2 decimal places. This function should return N/Z.

- Create a function that uses an isotope's Z and the N/Z ratio to determine whether the isotope is likely stable or likely unstable. Use the following criteria (hint: if-else statements are useful here) in your code to predict this:

  If $Z \leq 20$ and $N/Z \leq 1$: likely stable
  If $Z \leq 20$ and $N/Z > 1$: likely unstable,

  If $20 < Z < 80$ and $N/Z < 1.4$: likely stable
  If $20 < Z < 80$ and $N/Z \geq 1.4$: likely unstable

  If $Z \geq 80$: likely unstable

  The output should be a string, which states either: 'likely stable' or 'likely unstable'.

- Use function docstrings to explain the basic purpose of your function and to provide some details on the input arguments, what is being returned, and if there are any side effects (see the Background Reading #2 pdf document for details on this).

6. Create a print statement with string headings 'Isotope' 'N/Z', 'Stability'.
   Print this information for all your isotopes from Step 5, e.g., the following is a sample console output:

```
Isotope  N/Z  Stability

238-U  1.59  likely unstable
4-He  1.0  likely stable
24-Na  1.18  likely unstable
186-Au  1.35  likely stable
79-Zr  0.97  likely stable
```

To complete this Coding Exercise, you can structure your functions however you wish (one inside another, using variables and lists to assign to output from one function for use in another function, making additional functions, etc.).

Try to keep each function as simple as possible, with only a few main tasks happening in each one. Avoid calling functions five separate times (or re-writing any of the same lines of code over and over) for each of the isotopes. Try using lists and iteration as much as possible.