CSC 103 - HW 5

Complete each of the following exercises. To receive full credit please ensure the following:

- Submission structure follows the following:

```
HW4
    -> data
           -> poke.csv
           . . . .
    -> Q1
           -> q1.py
           . . . .
    -> Q2
           -> q2.py
           . . . .
    -> Q10
           -> q10.py
           . . . .
```

- Exercise artifacts are uploaded to your GitHub repository under a folder called `HW4` (i.e. https://github.com/<username>/carlow-me)
- Exercise artifacts are uploaded to Brightspace as a compressed `.zip` file called `<LastName>_HW5.zip`.

It is crucial that you are naming artifacts and storing files in the format mentioned above. All files that need to reference datasets should use a relative reference that refers to it's location in the `data` directory and not an absolute reference that would only work on your machine. Failure to meet any of these requirements will result in loss of points for that and any following problems.

Exercises

1 . Using the `pokemon_species` and `generations` datasets, create a single dataset. Write this new combined dataset to `q1.out`. Once you have a single dataset – get the following information and write the resulting dataframe in a csv format to `q1.csv`:

- Pokemon name
- Generation ID

2. Like we have in past weeks – build an application that allows a user to build a team of pokemon. This team can consist of a maximum of 6 pokemon. A user should be able to provide up to 6 pokemon names and they must exist within the `pokemon` dataset. Your menu should provide for the following options:

- Add pokemon
- Generate random team
- Delete Pokemon
- Exit

When a user decides to exit, draw a graph showing the `base_experience` for each member still existing in the team.

3. Using the appropriate datasets, draw a graph showing the number of pokemon within a region. The Y Axis should show the number of Pokemon within that region and the X axis should show the name (ex: Kanto). The name of the region should be capitalized appropriately. Ensure that appropriate labels are given to both the x and y axis and an appropriate title is applied to the graph as a whole. The labels should be formatted in a way that makes it easy for the user to know what the graph is visualizing.

4. Using the appropriate datasets, create a dataset that allows you to get a pokemons base experience and their primary color. Write this new dataset to `q4.out`. Grab a random selection of 10 of these pokemon and then draw a graph showing the name of the pokemon and their base_experience. The color for each bar should be set to what the primary color of that pokemon is. The x axis should be the name of the pokemon and the y axis should show the base_experience of the pokemon. Use appropriate labels and a title to allow the user to easily discover what your graph is visualizing.

5. Similar to what we just did, let's draw a graph showing the number of each primary type of pokemon that exist in a game. Create a new column called `type_color` that you use a lambda function to set. You can use whatever logic you would like to set the color based on available type names (i.e. you could create a function that returns a certain color based off of the names available in the `types` dataset). Once you have this new dataframe with the `type_color` applied, write this dataframe to `q5.out`. With this data in hand, create a new graph that shows the results (name of the type and its count) with each column being the color that was applied to that type.

6. Using the appropriate datasets, find what the most common secondary type of pokemon that exists is. Write the results out to `q6.out` showing the name and the number of that type.

7. Using the appropriate datasets, create a list of all mythical and legendary pokemon. Write this dataframe to a file called `q7.out`

8. Building from `q7`, create a dataset of all mythical and legendary pokemon. Create a new column for these pokemon called Strength that is based upon their base_experience multiplied by 5 added to the sum of their height and weight multiplied by 5. Return a dataframe with the strongest 5 of these creatures and write the dataframe to `q8.out`. Then draw a graph comparing these values.

9. We're back to dungeon crawling! Using a similar approach to the assignment in HW1, create a 4 x 4 dungeon. When the dungeon is generated, you should designate the spot that the user starts in and the spot that they must get to in order to exit. DO NOT SHOW THE USER WHICH SPACE IS THE EXIT.

Now, before allowing the user to begin, allow them to build a team of up to 6 members OR allow them to generate a team of 6 members. They will also need to select a location – this could be a specific location OR it could be an entire region. Now select a random selection of pokemon to put in each square of your dungeon. This random selection should ensure that only a single pokemon sits in each square.

Now as a user makes their way through their dungeon, if they land on a spot that is not an exit – the first pokemon that is available must battle the one that they encounter. If the base_experience of their pokemon is greater than the one they encounter – they can proceed! Otherwise their pokemon must be removed from their team. If a users team reaches 0, they are forced to exit and `failure` should be written to `status.out`. If a user decides to exit, you should write `failure` to `status.out`. If a user reaches the exit, then write `success` to `status.out` along with the number of members that are still in their team.