Eli Lustig

Q1

My initial goals were to create a pokemon battle simulator using PokeAPI and Google Images API. I planned to gather data about 100 different Pokemon, sort them by their type, assign each pokemon with their proper stats, and then ask the user to enter two pokemon they would like to see fight. Based on the calculations my program made the image of the victorious pokemon would appear onscreen.

Q2

Of the goals that I set out for myself, I only achieved about half. I was able to gather data about 100 different Pokemon and sort them by their type. Additionally, I was able to find the strongest Pokemon within each type. However, I was not able to make a full ‘simulator’. I also never ended up using Google Images API.

Q3

The biggest problem I faced was what metrics I should use to determine the strength of a Pokemon. PokeAPI provided me with lots of data but very little context about the data. The was a big factor in why I wasn’t able to complete the “simulation” part of my project. Determining which stats had more weight and how that would effect the outcome of a battle proved to be too difficult given the amount of time I had. I also face problems with presenting images after I had gathered data from Google API. I could retrieve the image URL from Google, but there was no way to displaying it using a python module if I only had a URL. I also had problems with the visualization even though I eventually figured it out. Matplotlib is very particular in how its whats the input. This forced me the find workarounds so that I could get my scatterplot looking how I wanted with having to compromise the data I was trying to present.

Q4

```
This is a list of 100 Pokemon and their stats which include type, name, attack, and health.


[('poison', 'bulbasaur', 49, 45), ('poison', 'ivysaur', 62, 60), ('poison', 'venusaur', 82, 80), ('fire', 'charmander', 52, 39), ('fire', 'charmeleon', 64, 58),
('flying', 'charizard', 84, 78), ('water', 'squirtle', 48, 44), ('water', 'wartortle', 63, 59), ('water', 'blastoise', 83, 79), ('bug', 'caterpie', 30, 45),
('bug', 'metapod', 20, 50), ('flying', 'butterfree', 45, 60), ('poison', 'weedle', 35, 40), ('poison', 'kakuna', 25, 45), ('poison', 'beedrill', 90, 65),
('flying', 'pidgey', 45, 40), ('flying', 'pidgeotto', 60, 63), ('flying', 'pidgeot', 80, 83), ('normal', 'rattata', 56, 30), ('normal', 'raticate', 81, 55),
('flying', 'spearow', 60, 40), ('flying', 'fearow', 90, 65), ('poison', 'ekans', 60, 35), ('poison', 'arbok', 95, 60), ('electric', 'pikachu', 55, 35),
('electric', 'raichu', 90, 60), ('ground', 'sandshrew', 75, 50), ('ground', 'sandslash', 100, 75), ('poison', 'nidoran-f', 47, 55), ('poison', 'nidorina', 62,
70), ('ground', 'nidoqueen', 92, 90), ('poison', 'nidoran-m', 57, 46), ('poison', 'nidorino', 72, 61), ('ground', 'nidoking', 102, 81), ('fairy', 'clefairy', 45,
70), ('fairy', 'clefable', 70, 95), ('fire', 'vulpix', 41, 38), ('fire', 'ninetales', 76, 73), ('fairy', 'jigglypuff', 45, 115), ('fairy', 'wigglytuff', 70,
140), ('flying', 'zubat', 45, 40), ('flying', 'golbat', 80, 75), ('poison', 'oddish', 50, 45), ('poison', 'gloom', 65, 60), ('poison', 'vileplume', 80, 75),
('grass', 'paras', 70, 35), ('grass', 'parasect', 95, 60), ('poison', 'venonat', 55, 60), ('poison', 'venomoth', 65, 70), ('ground', 'diglett', 55, 10),
('ground', 'dugtrio', 100, 35), ('normal', 'meowth', 45, 40), ('normal', 'persian', 70, 65), ('water', 'psyduck', 52, 50), ('water', 'golduck', 82, 80),
('fighting', 'mankey', 80, 40), ('fighting', 'primeape', 105, 65), ('fire', 'growlithe', 70, 55), ('fire', 'arcanine', 110, 90), ('water', 'poliwag', 50, 40),
('water', 'poliwhirl', 65, 65), ('fighting', 'poliwrath', 95, 90), ('psychic', 'abra', 20, 25), ('psychic', 'kadabra', 35, 40), ('psychic', 'alakazam', 50, 55),
('fighting', 'machop', 80, 70), ('fighting', 'machoke', 100, 80), ('fighting', 'machamp', 130, 90), ('poison', 'bellsprout', 75, 50), ('poison', 'weepinbell',
90, 65), ('poison', 'victreebel', 105, 80), ('poison', 'tentacool', 40, 40), ('poison', 'tentacruel', 70, 80), ('ground', 'geodude', 80, 40), ('ground',
'graveler', 95, 55), ('ground', 'golem', 120, 80), ('fire', 'ponyta', 85, 50), ('fire', 'rapidash', 100, 65), ('psychic', 'slowpoke', 65, 90), ('psychic',
'slowbro', 75, 95), ('steel', 'magnemite', 35, 25), ('steel', 'magneton', 60, 50), ('flying', 'farfetchd', 90, 52), ('flying', 'doduo', 85, 35), ('flying',
'dodrio', 110, 60), ('water', 'seel', 45, 65), ('ice', 'dewgong', 70, 90), ('poison', 'grimer', 80, 80), ('poison', 'muk', 105, 105), ('water', 'shellder', 65,
30), ('ice', 'cloyster', 95, 50), ('poison', 'gastly', 35, 30), ('poison', 'haunter', 50, 45), ('poison', 'gengar', 65, 60), ('ground', 'onix', 45, 35),
('psychic', 'drowzee', 48, 60), ('psychic', 'hypno', 73, 85), ('water', 'krabby', 105, 30), ('water', 'kingler', 130, 55), ('electric', 'voltorb', 30, 40)]
----------------------
This dictionary is a calcualtion of the strongest Pokemon of each type. This calcualtion was done by sorting the Pokemon
by their type and finding which pokemon had the highest attack score.


{'poison': [('victreebel', 105, 80)], 'fire': [('arcanine', 110, 90)], 'flying': [('dodrio', 110, 60)], 'water': [('kingler', 130, 55)], 'bug': [('caterpie', 30,
45)], 'normal': [('raticate', 81, 55)], 'electric': [('raichu', 90, 60)], 'ground': [('golem', 120, 80)], 'fairy': [('clefable', 70, 95)], 'grass': [('parasect',
95, 60)], 'fighting': [('machamp', 130, 90)], 'psychic': [('slowbro', 75, 95)], 'steel': [('magneton', 60, 50)], 'ice': [('cloyster', 95, 50)]}
```

Q5

1. Make sure you have a connection to the internet.
2. Make sure you have sqlite downloaded on your computer.
3. The code will produce two files: pokemon.sqlite and Pokemon.csv. Make sure both of these files are in the same folder as the python script of else the program will not work.
4. Run the program and a matplotlib graph should appear.

Q6

Get_data
This function takes a URL as input and returns a JSON dictionary. Each dictionary returned by this function corresponds to one Pokemon.

Get_stats
This function takes a JSON dictionary and returns the speed(as an integer), special defense(as an integer), special attack(as an integer), defense(as an integer), attack(as an integer), and hp “health”(as an integer) of the Pokemon.

Get_Type
This function takes a JSON dictionary and returns the type of the Pokemon.

Get_name
This function takes a JSON dictionary and returns the name of the Pokemon.

Q6

12/3/2018
Figuring out how to use pokemon data.
https://pokeapi.co/
Yes, I figured out how to extract data from pokeapi.

12/8/2018
Figuring out how to properly write data into a sqllite database.
SI206 modules
Yes, I figured out how to get pokeapi data in a sqlite database.

12/14/2018
Figuring out how to use matplotlib to make a scatter plot.
https://matplotlib.org/gallery/lines_bars_and_markers/scatter_with_legend.html#sphx-glr-gallery-lines-bars-and-markers-scatter-with-legend-py
Yes, I figured out how to make a scatter plot with my data.

12/14/2018
Figuring out how to orient my legend in my plot.
https://stackoverflow.com/questions/30413789/matplotlib-automatic-legend-outside-plot/30413956
Yes, I now know how to properly place my legend.

12/15/2018
Deciding what colors I should use for the different Pokemon types.
https://matplotlib.org/examples/color/named_colors.html
Yes, I found pretty colors.