**Project One Documentation :**

**Michael Nguyen**

This document includes some output screenshots of the different parts of my project. Included are output screenshots of the Pokemon Project, the statistics library, and the MonteCarlo Simulations. In addition to that, both the formula sheet and github essay are here. This is a bit redundant because they’re also on the github link but I figured I would put them here as well to be all encompassing.

**Formula Sheet :**

**Sample variance of a Data set**

**Standard deviation**

**Multiplicative law of probability**

P(

=

If A and B independent then

**Additive law of probability**

If A and B mutually exclusive

**Independence :**

Otherwise they are dependent.

Conditional Probability of event A, given B

**Combination :**

**Permutation :**

**Bayes’ Theorem :**

**Permutations for Discrete Random Variables**

**Variance of random Variable Y**

**Standard Deviation of Y**

**Probability Mass Function for Binomial Distribution**

E(Y) and V(Y) for Binomial Distribution

Expected

Variance

**Geometric Probability Distribution**

E(Y) / V(Y) for Geometric Distribution

=

**Hypergeometric Distribution**

**Negative Binomial Distribution**

**GitHub Essay :**

Git is like a giant undo button for building coding projects. You can use it on your own projects or in collaboration with other people. It helps people work together without messing things up. If something breaks, you can go back to an earlier version and fix it. To use Git, you need to understand a few basic things: repositories, workflows, pushing, pulling, merging, and fixing conflicts. Understanding these makes things much easier when collaborating on a project because it allows everyone to make changes without fear of ruining the entire project.

A repository (or repo) is just a folder where all your project files are, along with a history of changes. It can be on your computer or online on sites like GitHub or GitLab, where others can access it too. Repositories help keep track of every change made to a project so nothing gets lost. Different teams follow different workflows, which are just plans for how they work together using Git. Atlassian’s "Comparing Workflows" page explains some common ways teams organize their work. Picking the right workflow makes teamwork easier and helps prevent confusion.

When you push, you're sending your changes from your computer to the online repo so others can see them. This is like saving your work and sharing it with your team. When you pull, you're grabbing the latest changes from the repo so your work stays updated. This helps everyone work with the newest version and avoid problems like accidentally overwriting someone else’s work.

Merging is when different people work on the same project and then combine their changes. If two people worked on different parts of a project, Git can merge them easily. But sometimes, Git gets confused if two people changed the same thing. That’s a merge conflict, and it means you have to decide which changes to keep—like if two people edited the same sentence in a shared document. Merge conflicts require manual fixing, but once resolved, Git can continue combining everyone’s work smoothly.

There are different ways teams use Git to stay organized. The Gitflow method is good for teams that like to work on features separately before combining them. It creates a clear structure and prevents unfinished work from affecting the main project. The Forking method is common in open-source projects, where people copy a project, make changes, and then suggest updates without affecting the original version. This is great for people contributing to projects they don’t officially own.

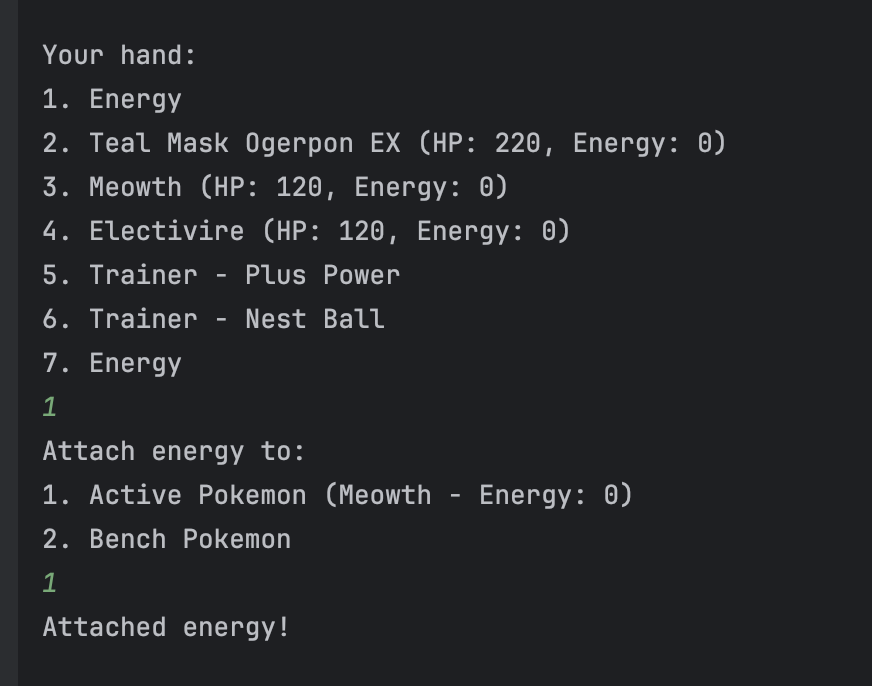
In short, Git helps teams stay organized and avoid big headaches. Knowing how to push, pull, merge, and fix conflicts makes working together a lot smoother. It also ensures that work is never lost and that everyone is always on the same page. For example, currently in my Software Engineering class we have to use Github for work incorporating the front end and backend. Without it, it would be extremely scary to push changes on the project because if things go wrong there’s not an undo button like this to revert changes for us. Git is very beneficial for both large and small scale projects. It is good for working by yourself (such as how we are with our project one) or in a group. Using it is imperative to a well structured and organized project.

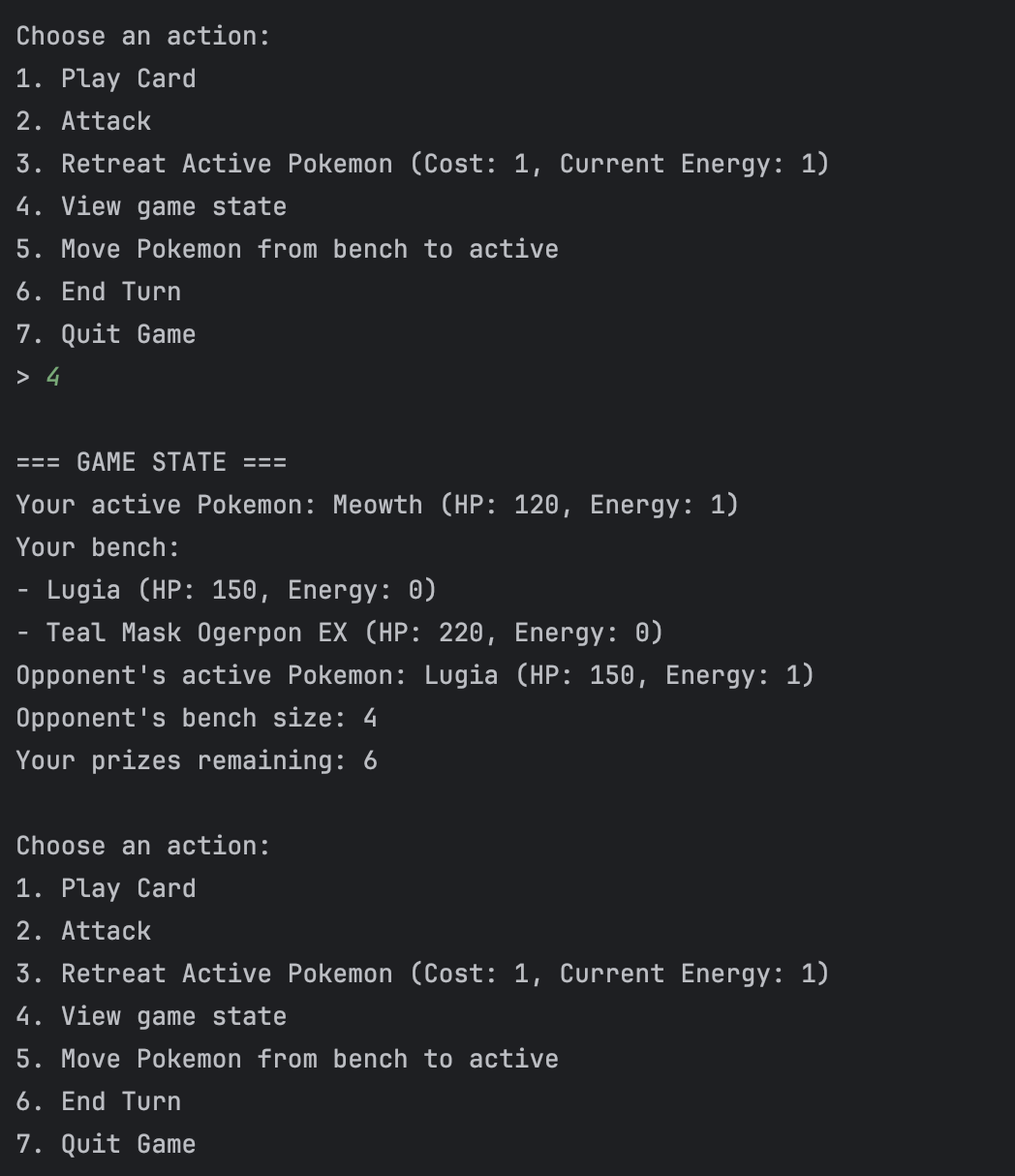
Pokemon Project output screenshots from one game my friend (pokemon TCG pro) and I played against each other

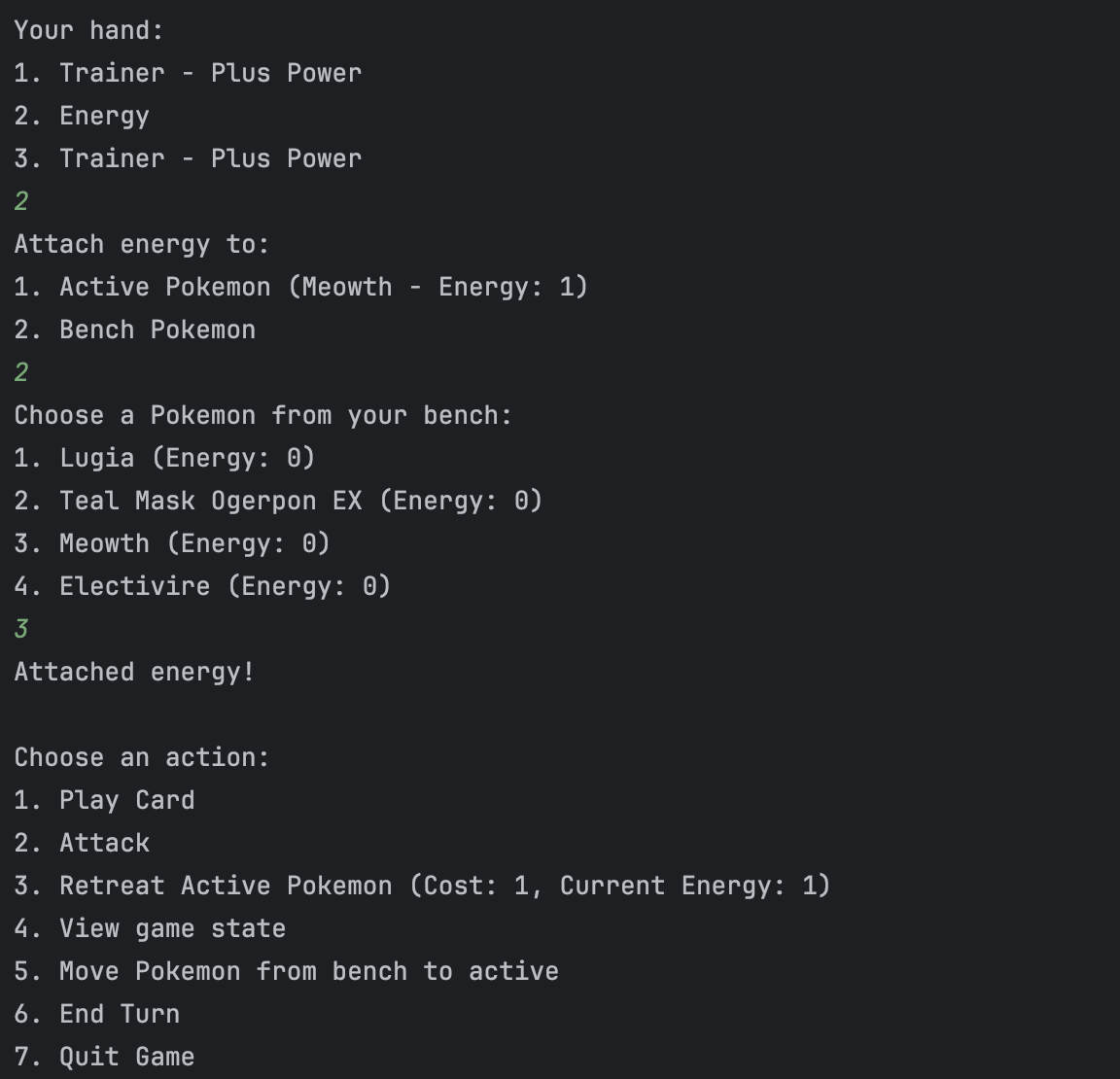
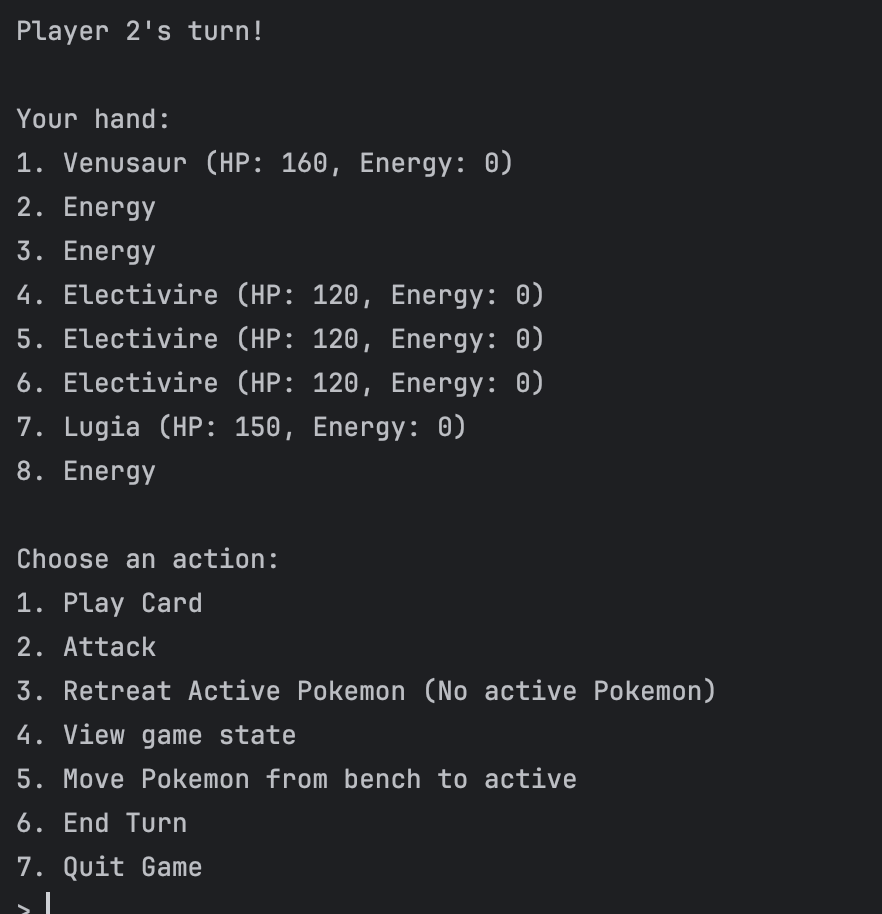
**Pokemon Project :**

**Screenshots are from a game I played against my friend who competes professionally for Pokemon TCG**

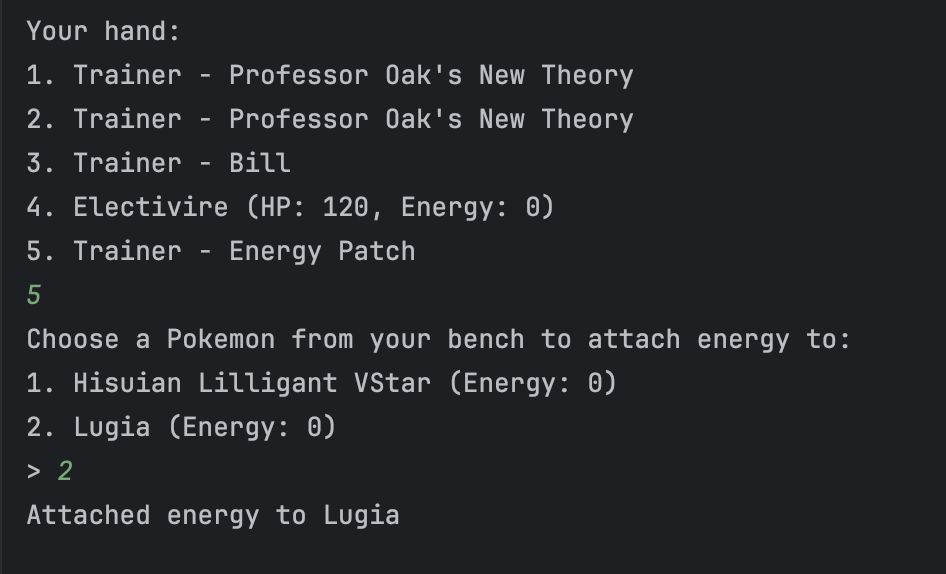
**Example 1st turn**



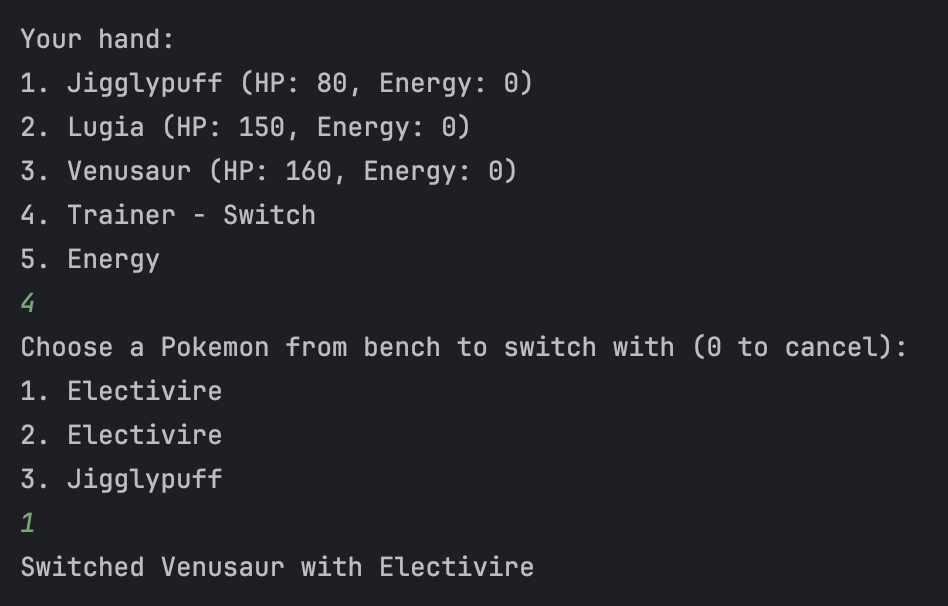
**Player 2’s first turn**



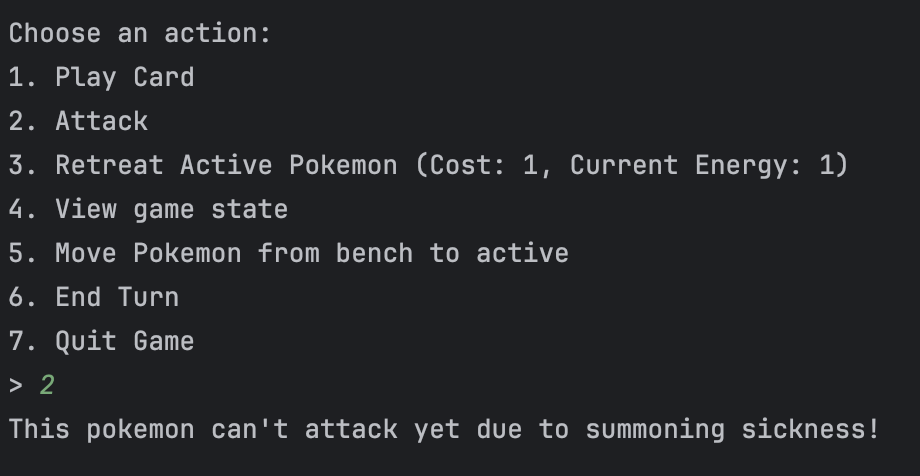
Energy Patch utilization

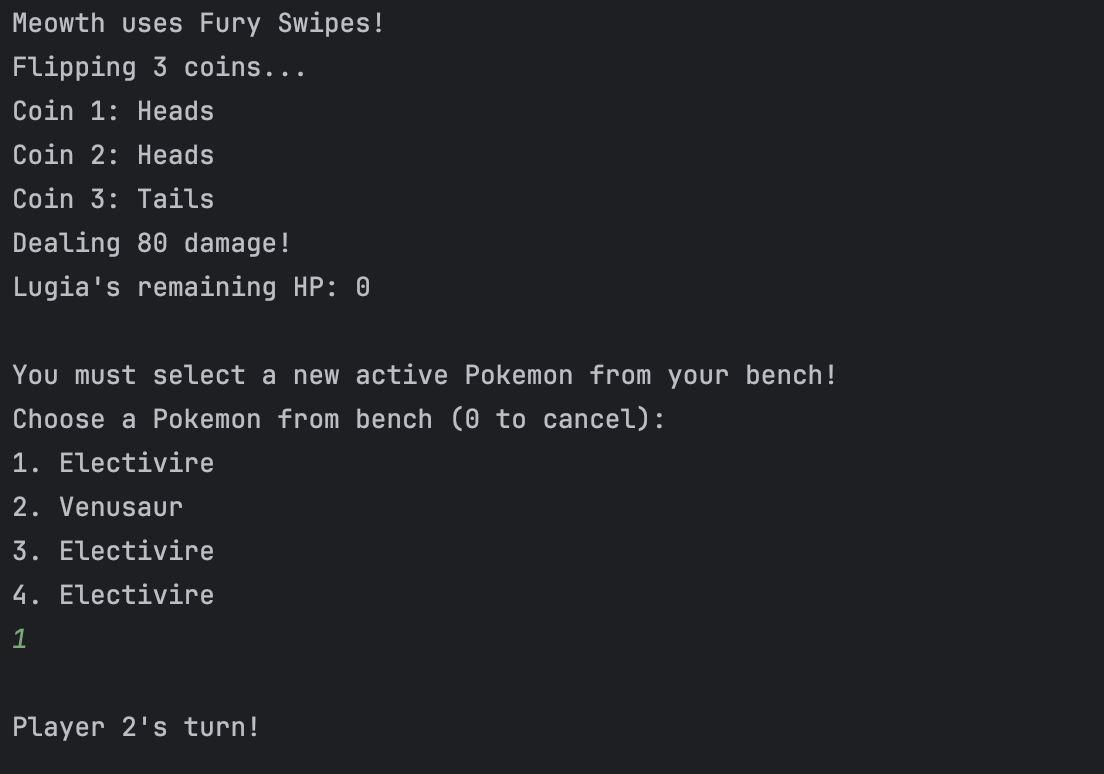


Switch Trainer card utilization ; bypasses energy required to retreat normally.



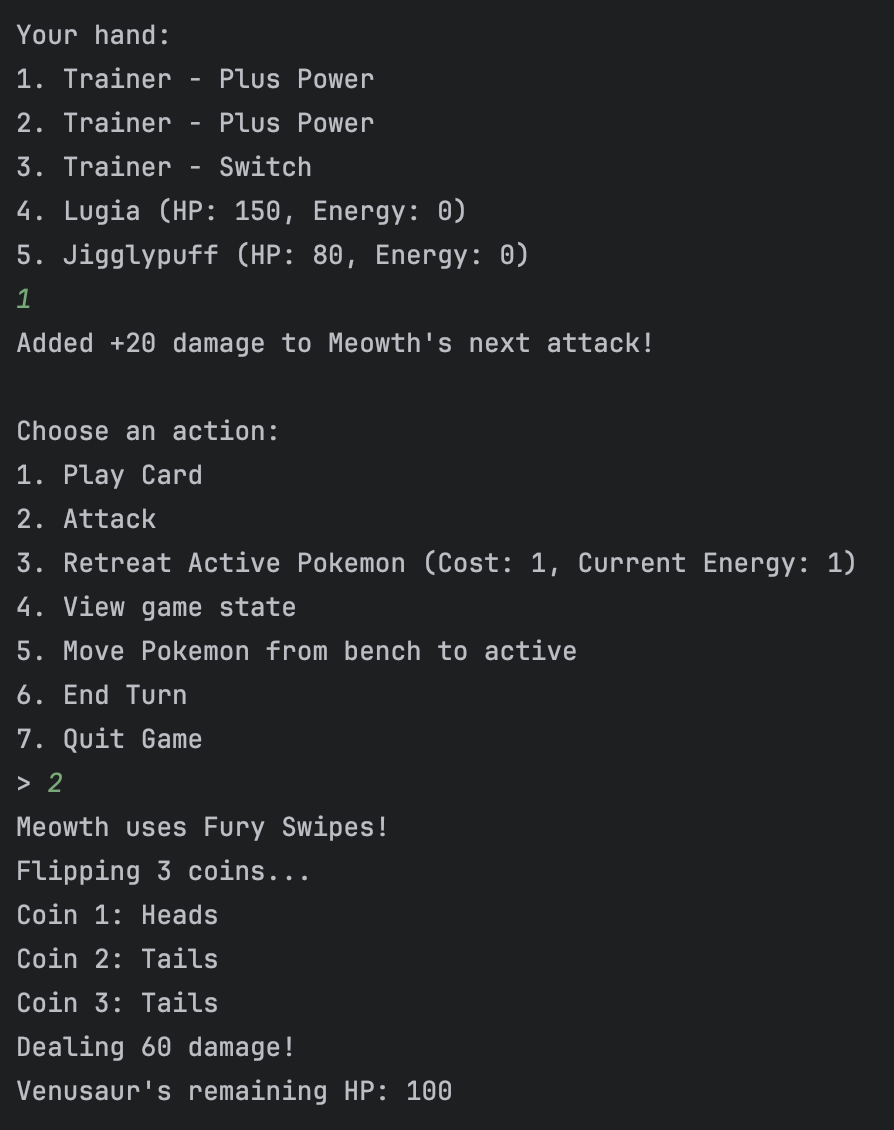
Summoning Sickness not letting the player attack with the pokemon they just played.

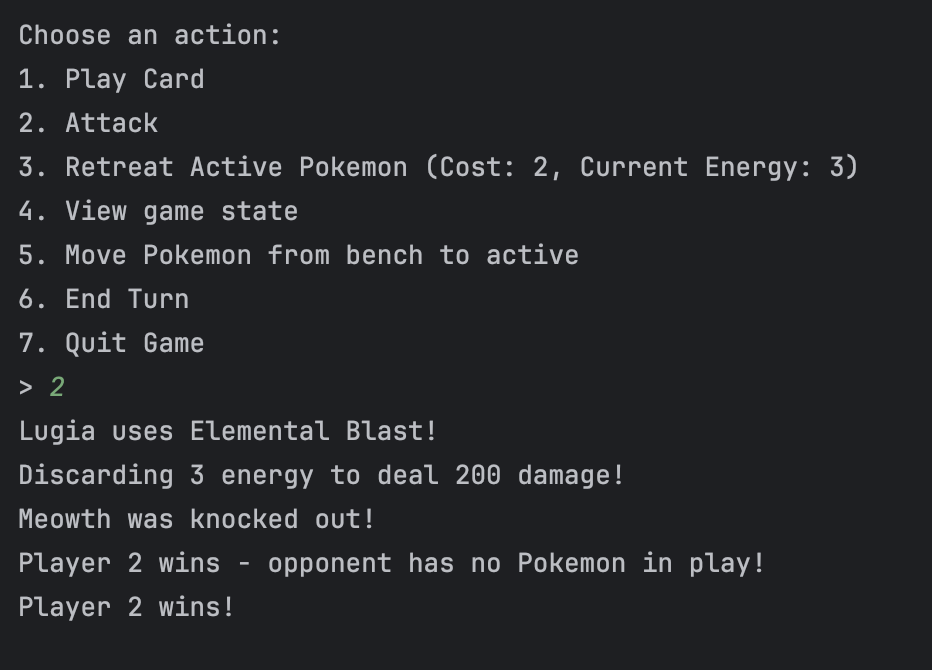




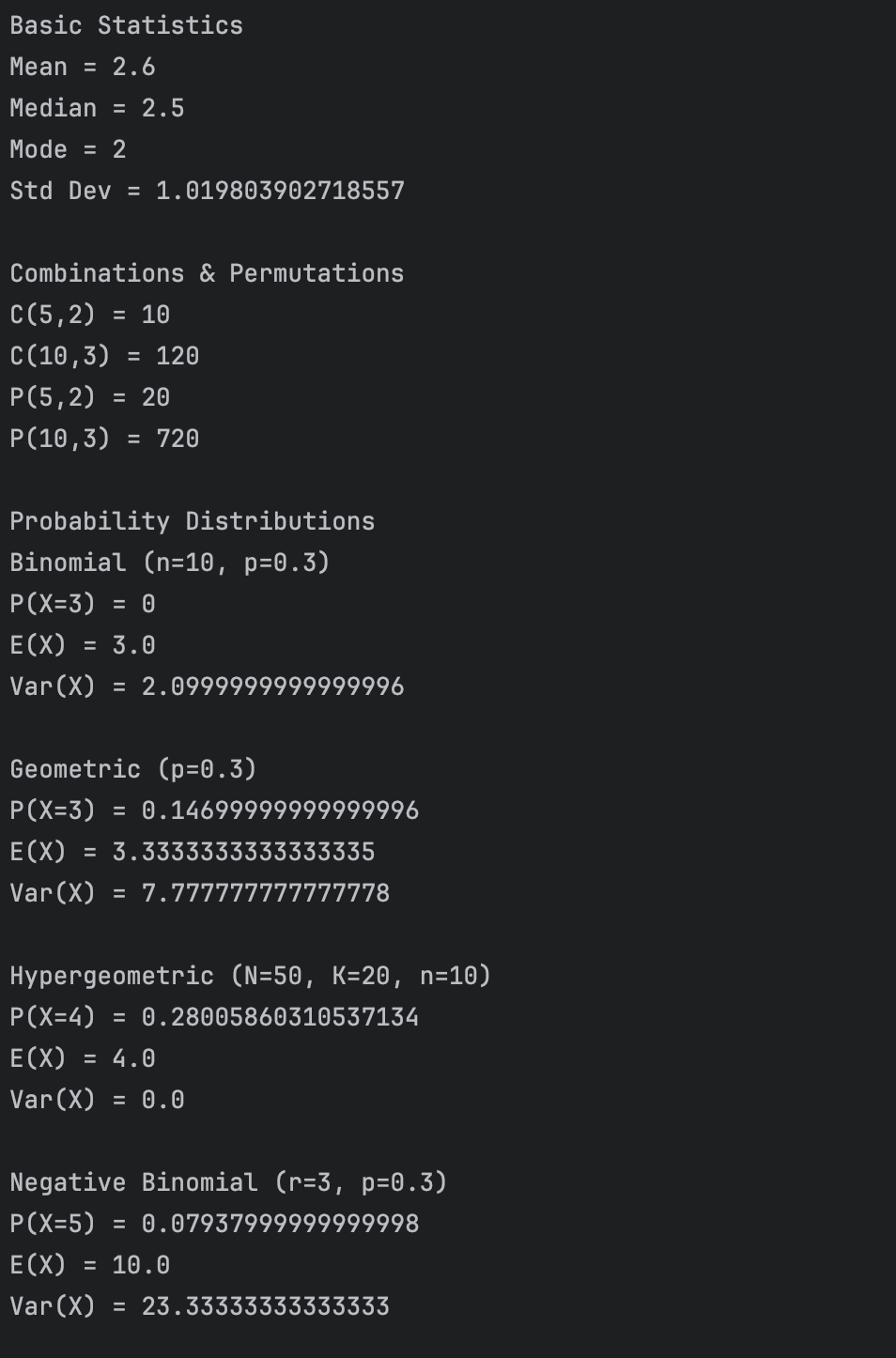
**Meowth Attack output + knock out**

**User chooses a pokemon from their bench to put as their active prior to continuing.**

Fury Swipes being used with Plus Power trainer card

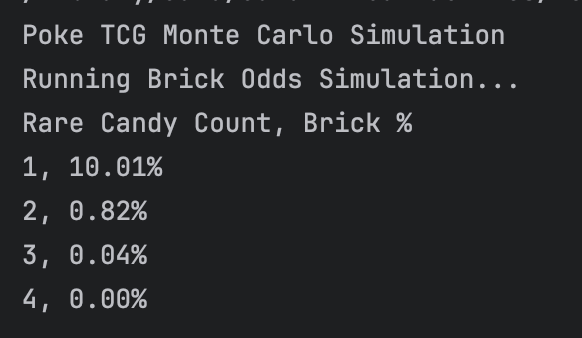


Player 2 winning screenshot

**Statistics Library :** 

**Monte Carlo :**

**Output screenshot for the rare candy brick simulation**



**Output for the Mulligan odds sim. It was far too long to take a picture.**

Running Mulligan Odds Simulation...

Pokemon Count, Mulligan %

1, 88.69%

2, 77.35%

3, 70.00%

4, 59.20%

5, 52.26%

6, 45.87%

7, 39.53%

8, 34.90%

9, 30.26%

10, 25.90%

11, 21.82%

12, 18.69%

13, 16.08%

14, 14.06%

15, 11.78%

16, 10.33%

17, 8.72%

18, 7.13%

19, 5.68%

20, 5.02%

21, 4.11%

22, 3.41%

23, 2.73%

24, 2.06%

25, 1.81%

26, 1.32%

27, 1.17%

28, 0.91%

29, 0.66%

30, 0.68%

31, 0.34%

32, 0.34%

33, 0.22%

34, 0.12%

35, 0.18%

36, 0.10%

37, 0.09%

38, 0.09%

39, 0.03%

40, 0.04%

41, 0.02%

42, 0.02%

43, 0.00%

44, 0.00%

45, 0.01%

46, 0.00%

47, 0.00%

48, 0.00%

49, 0.00%

50, 0.00%

51, 0.00%

52, 0.00%

53, 0.00%

54, 0.00%

55, 0.00%

56, 0.00%

57, 0.00%

58, 0.00%

59, 0.00%

**Output for the DoorGame monte carlo sim.**

