Name: Daniel Ems
Date: 03/04/2017
Current Module: Object Oriented Programming
Project Name: dungeon_dudes

Project Goals:

The goal of the project was to take create a simple game of characters, rooms, treasure and battles. The game requires the hero move from room to room battling monsters and collecting loot. However, the true test of the project is to organize your game such that it utilizes object oriented programming.

Objectives:
1. Create a hero that has health, strength, and a loot bag
2. Initialize a quest with a hero
3. Provide the user with the ability select from a menu
4. Create a monster for the user to battle
5. Implement a battle function dependent on dice rolls that updates results
6. Implement the random dropping of loot
7. Implement a method for moving through rooms
8. Provide the user with constant access to the menu

Considerations:

- Determine what elements of the game should be considered an object.
- Determine what that object 'HAS' versus what it 'IS
- Determine if there are objects that can inherit traits from the same parent
- Make sure to follow the DRY principle
- Identify the best way to organize the objects logically
- Identify the best implementation of your objects as a collection
- Error handling user input
- Error handling menu edge cases.
- Identifying what is not an object, and rather is a function

Initial Design:

The initial design for this project revolved around two important ideas; the concept of a quest, and the concept of objects in a quest. Logically, a quest is an adventure that has a hero, monsters, locations or scenes, and some form of action. In this instance, that action is battle. So, this quest has a hero, monsters, scenes, and battles. This served to conceptually break down what to look at as "objects", and what not to. This train of logic forces one to ask questions such as; "what is a hero?", "what is a monster?". They are both characters. This mental evaluation of a programs requirements helps to create a clean and logical design early in your development process.

Data Flow:

1. Allow the user to pick there name and error check
2. Initiate a quest which in turn initiates a hero, monsters, a map, and diceRolls

3. Provide the ability to check loot, next room, hero health, monster health, and battle
4. Error handle edge cases to user options
5. Error handle user input for selections
6. Determine a winner and loser in battle
7. Modify the health of the loser appropriately.
8. Notify the user if he has won or died

Potential Pitfalls

The largest pitfall identified from this project is in organizing classes that work together. Accessing class attributes and modules, or passing them through out your program, can make it difficult to keep track of. This creates a very unique problem while you write you are in the development stage of your program. However, this potential pitfall is one that I believe decreases with time and experience in doing such. Like wise, the benefits you receive far our weigh the cons.

In addition to the original pitfall, being inexperience in developing classes and organizing them together can make it perhaps inefficient overall. While there are standards to follow, and methods to practice, the only one can truly maximize object oriented programmings effects is through trial and error through implementations

Test Plan

User Test:
1. Identify a potential class and what it has
2. Create class and then see if I can access their attributes, and call their modules.
3. Try to break the classes by passing unexpected behavior.
4. Repeat this for every class created.
5. Begin to organize classes in relation to one another.
6. Rpeat step 3
7. On user Input, try to pass unexpected values.
8. In game play, attempt to break the game disrupting the logical flow.

Test Cases:
1. Once an element of the game was created, it was immediately tested for flaws
2. If flaws were found then they were patched and step 1 was repeated.
3. As new elements of the game were built upon others, continue to try and break the code from the beginning through the end.
4. Repeat step 2

Conclusion:

The most effective element of this project was the instruction in class. The instructors did a great job of introducing classes and objects. I felt confident in implementing objects and classes. While I may have not done so %100 correctly, I do feel I did so well and that I will be able to learn as we move forward.

The most hindering element of the project was the time lapse in programming in python. We had a very successful week in brushing up on what is and is not pythonic. However, with the additional

instruction we were given, there was a fair amount of time simply going back to see how something simple should be done in python. This is at no fault of the instructors or the course.