TDQC

Dungeon Dudes

Jack Spence

2 November 2018

1. Write-Up

1.1 Requirements

Requirements were to create a simple RPG style game in Python3. The game is to support player versus monster combat and use simulated dice rolling for attacks and for initiative. The program is also meant to support a -d option to print the dice rolls that happen during combat. Another requirement was for there to be a chance of loot dropping from the enemy creatures upon being defeated.

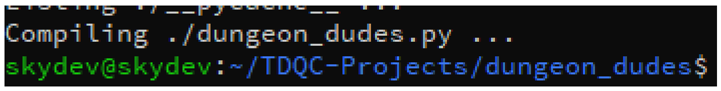
1.2 Suggested Features

Suggested feature attempted in my submission is to have a combat potion be a potential loot drop that allows the player to consume it at will and give them a temporary bonus die on their next combat roll.

1.3 Syntax

Compile:

python3 -m compileall . - compiles with the following output



```
Compiling ./dungeon_dudes.py ...
skydev@skydev:~/TDQC-Projects/dungeon_dudes$
```

Usage: ./dungeon_dudes [-h | --help] [-d]

-h or –help: Prints usage statement

-d: Prints the dice that are rolled during combat

2. Project Design Plans

2.1 Initial Design Plans

The big focus on this program was to try and stick to the inheritance portion of Python objects and utilize that the best I could. Having the parent class of Character and the two subclasses handle all of my character interactions was the key design feature of this program.

2.2 What didn't work

Brushing off the cobwebs of Python was a bit tricky at first. Programming in C for so long made me over think a lot of things, and I had just glazed over some of the options that were now available with Python. One example being with the room creations. I had forgotten about the choice function in random and was instead using itertools and permutations to generate a huge list of all available room options. I believe using the 3 lists and adding random combinations to a set is a more Pythonic solution with less overhead because unused rooms will never be loaded.

2.3 What went well

Implementing the classes went exceptionally well. Getting them to inherit things properly as well as making each class and subclass have all the relevant class and instance variables was quite easy.

2.4 Conclusion

As stated above, I believe Python inheritance is really the bread and butter of this project. Being able to save time and confusion with inheritance is definitely something I enjoyed with Python in comparison to C.

3. Test Procedures

3.1 Testing

I typically like to create test scripts with these projects but this one is rather straight forward and has a limited amount of test cases I could come up with. I think I could have benefited on the testing by using TDD and unittest, but simply didn’t get the time to write up tests for each function.

3.2 Testing Procedure

I used the brute force method of unit testing and just running the program as I went along. At first I wanted to make sure at any time the user is getting input that I handle the EOF or the CTL+C exceptions, I chose to just print a “Good bye” message and exit the program.

I made the conscious decision to not do a lot of checking on the name itself because I felt like it would be appropriate to allow the player’s name to be anything. I tried using a couple of the ANSI escape sequences to try and clear the screen but Python knew to treat it as a string literal and did not have any adverse affects.

To test the ending of the game, in the unlikely scenario the player defeats the gauntlet of enemies, I increased the players health to 10,000 and played the game. When the final enemy was defeated the game ended properly and displayed the end game information.

Lastly I checked to make sure that any unavailable menu options just re-prompted the player with the menu, which worked as it should.