# Dungeon Dave by Razal Minhas

## Reflection Document (10 points)

When submitting your final code, include a ~one page reflection document containing:

- Instructions about how we should go about testing and using your project
- What you completed and anything that you didn't complete in the project that you would finish later
- Discuss challenges you faced and how you overcame them
- Please make this document a pdf before uploading to github!

### System Requirements

- Use Windows Command Prompt (do not use a Linux shell or emulator)
- Install the numpy package

### Start the Game

- python dungeon.py

### Help:

- python dungeon.py help

### What is Complete

**Game Engine**

The game engine is complete and has the following steps:

1. Render screen
2. Present options to user (e.g. fight or run)
3. Accept user input
4. Render screen
5. Update game state

**Key features**:

- Ability to enhance the dungeon map and add more rooms by updating the list of rooms and tuples
- Random generation of monsters and treasures in rooms so that no two games experiences are alike
- Random assignment of vital statistics to the player, monsters and treasures

### What is not Complete

- Compatibility with Linux shells
- There is room to make the user experience more pretty
- I need to make the “agility” statistic impactful and have bigger influence on the player vs. monster fights.

## Challenges

- Writing the game engine loop and making updates in a sequence that flows. A lot of experimentation and reading other game loops on the internet was needed.

- Added the same information in different variables and spent time cleaning up the code, e.g.:
  - Added too many methods to navigate the rooms in the dungeon and things got confusing. Spent time cleaning it up and consolidating the functionality into the Dungeon class.
  - Added too many variables that held the current location of the player in the dungeon and had difficulty following my own code. Spent time cleaning this up.

- Once the game engine was complete, the UI was not attractive at all. I had to figure out how to create UTF8 files with Unicode characters that can show game screens. Then I identified UTF-8 font numbers (0 to 10) that can be used to render stats on the screen.

- Had encoding issues in the files that I had to fix. Used a hex editor to identify where lines end since text was not lining up on the screen. UltraEdit is was very useful in this regard.

- Debugging a game loop with multiple state variables is hard. Had to figure out how to use the debugger in Visual Studio Code. Success.

- Scrolling the code and following it is hard in a single 700+ line file. Had to split the file into multiple screens so I can work on the code in different places in the file.