Jessica Zoltai, Lauren Blaire, Bryce Watson, Andrew Schwarz, Ryan Eves

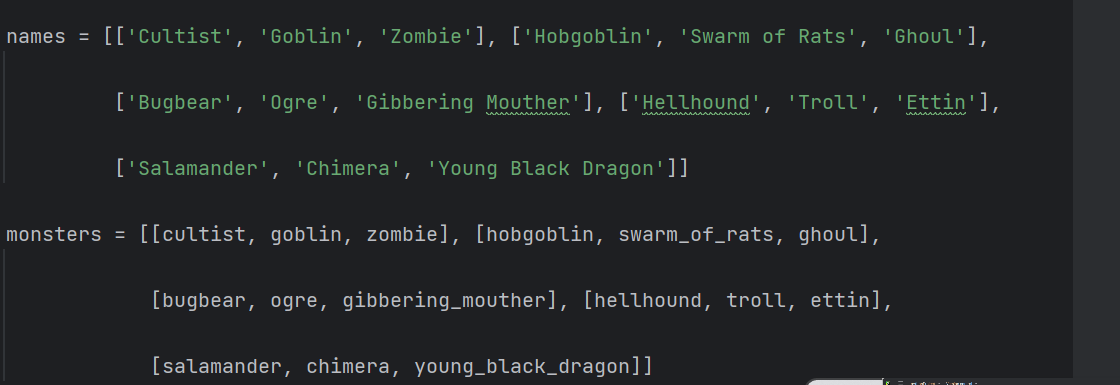
Pamela Thomas

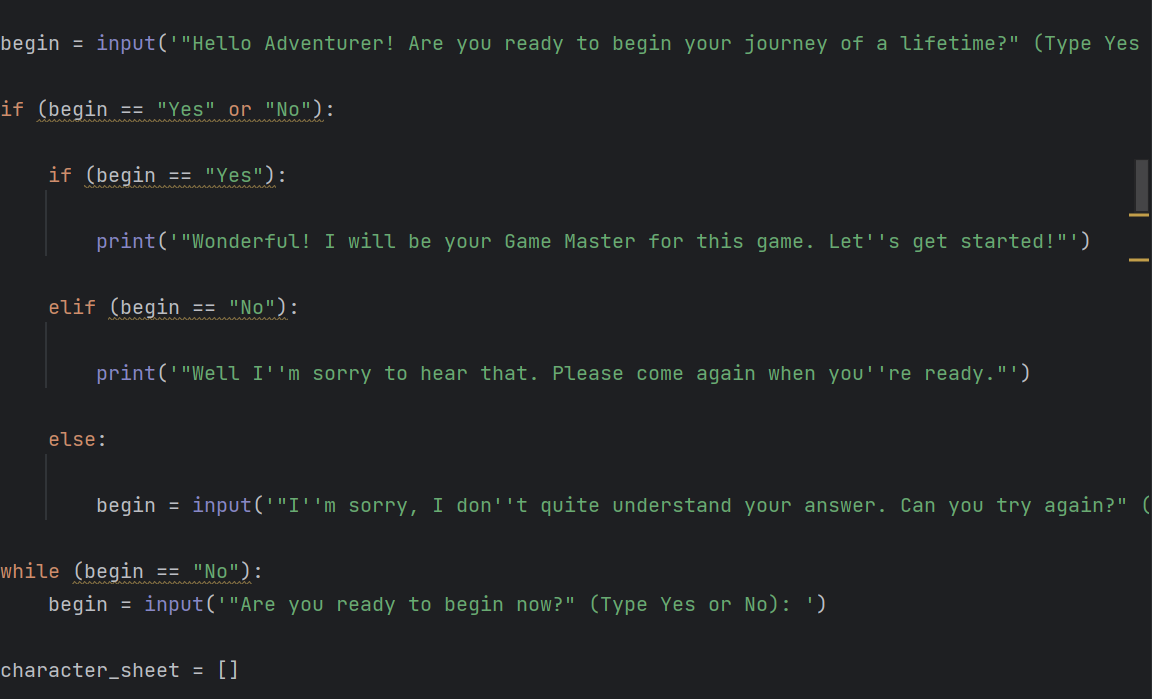
CSE 120

7/26/23

This paper is an overview of an assigned group project for CSE 120. For our group project, we created a text-based adventure game based off a popular game called DnD. The game asks for user input which will guide players through their own adventure through a series of choices. Our code for the game uses many skills that we have learned in this class including loops, arrays, if statements, importing packages, class/objects, and lists. Bryce Watson, Andrew Schwarz, and Ryan Eves wrote the code for the project. Jessica Zoltai and Lauren Blair wrote the paper and presentation. While Jessica Zoltai and Andrew Schwarz worked on debugging the code. For the presentation, we all divided the slides up and each person recorded a separate video explaining the code and what skills we used. Jessica then played the game to give a run through of exactly how the code worked. We then compiled all of these into one video to complete our final presentation.

The first part of the code assigns variables to the different stats of all the monsters. It also assigns stats to location visits. The monster stats are randomized using the randint function within a preset range. The location visits were originally set to zero. This number would update as a player visits each location. This enables different text to be printed when a player enters a location for the second time. It also enables different options when a player clears a dungeon. The code then creates a 2d array for the monster's names and types of monsters.

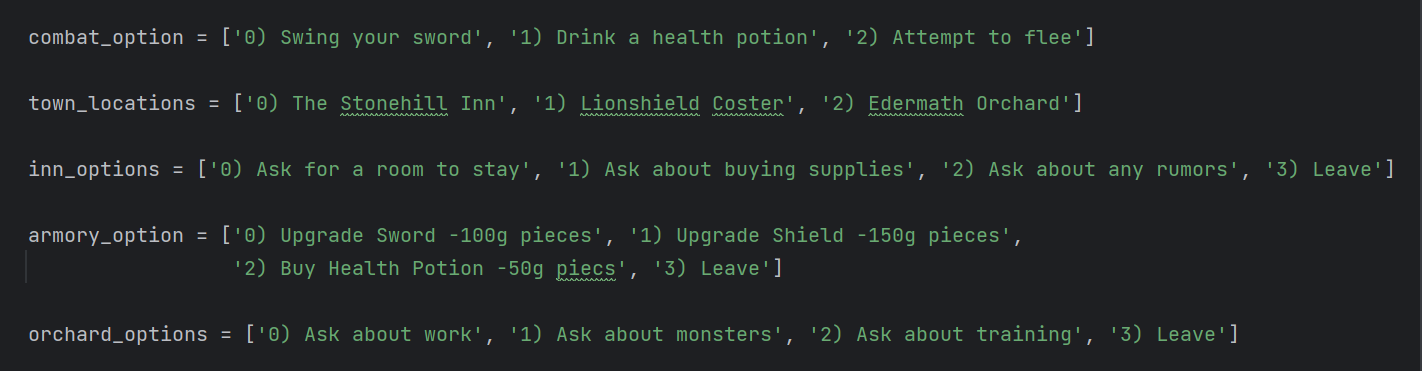


The first input asks the player if they are ready to start. As you can see in the picture to the left, the first user input states. ”Hello Adventurer! Are you ready to begin your journey of a lifetime?” The next line then uses an if statement depending on if the user answers yes or no. If the user enters yes, the code will print a statement motioning the start of the game. An elif statement is used if the user answers no and tells the player to come again when they are ready. The last statement used in this block of code is an else statement which is used when a player does not answer yes or no. The program does not understand what this player wants to do, so they will ask the question again to receive a yes or no answer. The final statement in this picture is a while loop which is used after a player originally answers no. The while statement will ask the player if they are ready to begin now. The program will also expect this answer to be yes or no.

The code will then ask a series of questions to collect character data. The code will ask for a variety of character information including name, class, and race. To collect this data, a character sheet empty array is created. Once the user is ready to begin, they are asked to enter a name. The user is then asked to enter their character's race. The three options are human, dwarf, or elf. The character class is always fighter. The strength of the character is then randomly selected using the random function. This is all appended to the character sheet array in order to keep track of all the players information. Next, an armor class is also randomly selected using the random function. The hit point is then set. The number of swords and the level of experience is set to zero. The character level is set to 1. A random amount of gold is then given in increments of 25 using the random function. The amount of health potions is also set to zero. If statements are then used to give bonus armor class, strength, or hit points based on the characters race. The stats names are then assigned to a dictionary.

The program then prints a statement that tells the user a spell is going to be cast on them. The player is then told to press enter to begin the tale. Several other variables are then set equal to zero, including the number of rumors, inn visits, armory visits, orchard visits, shrine visits, cave visits and times slept. The turn count has been set to 1.

The actions, options in combat, town locations, inn options, armory options, and orchard options have been put into separate arrays. The screenshot below shows how these arrays are used in our code.



The player can be asked where they would like to go. The locations they are given to choose from include The Stonehill Inn, Lionshield Coster, or Edermath Orchard. For options in combat, the user can either choose to swing their sword, drink a health potion, or attempt to flee. The player can also be provided with options to choose from at the inn. The player can choose to ask for a room to stay, ask about buying supplies, ask about any rumors, or leave. The armory options players are asked to choose from include upgrading their sword, upgrading their shield, buying health potions, or leaving. The orchard options are to either ask about work, ask about monsters, ask about training, or leave. Several statements are then printed that describe the setting of the game. A while loop and if statement is then used that allows the player to choose where they would like to go. If statements are then used to print the correct print statement depending on if the player has been to the different places in town before or not; which is known by updating the values for the various options visits variables when you enter them.

We also created a combat function that simulates a simple turn-based combat between a player character and an enemy. Within this function, three other functions are used. These three functions are attack\_dmg(), roll\_d20(), and potion\_heal(). The function takes the parameters player\_health, player\_ac, dmg\_mod, enemy\_health, enemy\_ac, enemy\_dmg\_roll, and used\_potion (boolean). Players have the choice to attack or use a healing potion. A d20 function is used to simulate an attack to hit which is compared against the defender’s ac.

* Another important aspect of our code was the inventory class we used. The inventory class is meant to represent a bag or container. The inventory class has parameters name, contents and optional parameters for carrying capacity and droppable items. The “def\_\_str\_\_” statement customizes the string representation of the object when printed. The “def\_\_add\_\_” statement add items to the bag along with their weight. The “def\_\_remove\_\_” statement call an item in the bag and removes it.

The players' input will now take them on their adventure through the game and they will have to make decisions that determine their journey. The story progresses through the rumor function at the inn. By selecting the Listen to rumor option while inside the inn, you will add options to your map which are dungeons. The options at the orchard or the Lionshield allow the player to strengthen themselves and their weapons by appending things to your stats dictionary. While resting at the inn allows the player to regain all their lost hp to their hp maximum. The code will run until your hp is reduced to zero. If your hp is reduced to zero while fighting the last boss, a series of print statements will run detailing the end of the world and then you will be cycled back to the beginning of the loop to start the game over again. If your hp is reduced to zero while inside the last dungeon, but not fighting the final boss, you will be asked if you would like to restart the final dungeon and the loop for the dungeon will restart. If you defeat the last boss, a series of print statements detailing your accomplishment and a final THE END, is the end of the loop.

Overall, our group worked together to successfully create an adventure game. There were some difficulties we encountered while writing our code. For the combat function, we had issues with getting the health of a character to correctly update after each round. For direction 4, we had trouble getting the while loop that we had used to allow the player to go back to the beginning of the location if their heath reached to function properly. The error that kept occurring is the program would just skip to the next part of the code. Another major challenge we encountered was organization and keeping track of indentations, specifically when creating 2d loops and if statements. We also struggled with redefining variables in the while loop. When the player decided to restart from the beginning of the location after their health reached 0, the variables from their previous run would stay the same. This caused the code not to function the same the second time around. Despite these challenges, we used what we had learned in this class and each other to help solve the difficulties and finish our code.