Team 7

Members: Bar▮▮▮▮▮▮▮▮▮, Lim Si Hui Brenda

Description:

The game is a text-based Role Playing Game (RPG) with ASCII Art inspired graphical output and is modelled after graphical elements of Maplestory and gameplay from the battle sequences of the Pokemon games.

The gameplay is split into two parts: a story exploration game and a battle sequence. The story exploration allows the player to find items that will increase their stat. The objective of the game is to defeat the monster by reducing its health points (HP) to zero, by means of attacking the monster.

If the player attacks continuously, the player will eventually win and the game will end; if the player continuously selects to taunt the monster, check the inventory or he input zero as his attack value and continues attacking, the game will not end; if the player chooses to run, the monster will attack, the player loses and the game ends.

Documentation:

No additional libraries are required to run the game.

#String Format. This list of formats controls the justification for the text display of the game. Collating them at one place also allows for future addition of a feature to control the display resolution of the game.

#Dictionaries. Dictionary for the player, player inventory and the monster to contain the respective stat values and items. Monster dictionary may be expanded to contain other monsters.

#Frame Listings. These pack each line of the display into a list. There is potential to adjust the vertical display resolution through here.

##Combined Graphics List. These sets of lists combine two lists which represent two halves of the screen using the zip() function and displays them as a pair of tuples on each line. As the lists get updated with new values from the updated dictionaries, the tuples have to be repackaged prior to running the display_graphic() function.

display_grapahic(graphics, player, monster): This function takes in the frame lists, and player-monster dictionaries as parameters. When run the function prints each element in the list as one line. This function is executed inside most of all other functions in the game.

main_menu(player, monster, inventory). This function takes in the player, monster and inventory dictionaries as its parameters. This is the function that runs the game and all other functions are nested within the while-elif loop of this function.

The functions below are nested in this function:
- display_graphic(graphics, player, monster)
- game(player, monster, inventory)
- option()
- exit_frame()

option(). This function displays the option screen. There are no options and the player is returned to the main menu on input.

exit_frame(). This function displays the exit confirmation screen and either exits the program or returns the player to the main menu.

stat_input(stat). This function takes in the name of the stat as a parameter. The function takes in the stat name as a key, appends the value input by the user and updates the player dictionary with the key-value pair.

weapon_selection(player, monster, inventory). This function takes in the player, monster and inventory dictionaries as its parameters. The function displays the weapon selection screen and prompts the user for a string. The string is then set as a value to the weapon key and the player dictionary is updated to the key-value pair.

game(player, monster, inventory). This function starts the gameplay sequence of the game. It takes in the player, monster and inventory dictionaries as its parameters. The player is asked for their name and the stat_input(stat) function is used for stat input.

The function then runs the following functions:
- weapon_selection(player, monster, inventory)
- battle_action(player, monster, inventory)

battle_action(player, monster, inventory). This function takes in the player, monster and inventory dictionaries as its parameters. The function displays the monster graphic, player and monster stat info and prompts the user for an action.

- Attack. Runs the attack_sequence function and calculates the resultant health point of the monster and updates the respective monster key-value pair. Once the health point of the monster reaches zero, the function returns the player to the main menu.

- Taunt. Increases the attack value while decreasing the defence value of the monster. The monster dictionary is updated accordingly.

- Inventory. The inventory screen is shown and tells the player what he possesses. There is a choice to use the 'red solution' if he had obtained it in the story exploration game. The player is then returned to the action selection screen upon input.

- Run. Confirms the player's intention to run away from the monster. On running away, the player's health points is reduced to zero, the game ends and the function returns the player to the main menu.

attack_sequence(player, monster, inventory). This function takes in the player, monster and inventory dictionaries as its parameters. This function simply shows the monster being attacked and prompts the player to continue.

Function Type A: Story Display Functions - Takes in the player, monster and inventory dictionaries as its parameters. Used in the following functions:

a. **story()** - displays text sequentially on player input, leads to three_object_choice().
b. **before_battle()** - displays text sequentially on player input, leads to set_stats(player, monster, inventory).
c. **three_objects_cleared()** - displays text to tell the player that he has completed the exploration.

Function Type B: Story Decision Functions - Uses if-else loops for players to make a choice. Takes in the player, monster and inventory dictionaries as its parameters. Used in the following functions:

a. **three_objects_choice()** - the 'main page' of the story exploration game which provides the player with three objects to choose from to explore.
b. **three_objects_proceed_or_not()** - executed when the player has interacted with the side table and cupboard, whether or not they were cleared, so that the player can choose to proceed on with the game.
c. **side_table()** - provides the player with an opportunity to interact with the side table.
d. **side_table_interaction()** - a mini choice game
e. **open_side_table()** - executed after the player chooses to <Try again>. Player's 'Attack_Modifier' increases and he gains a bottle of red solution which may be useful in the battle afterwards.
f. **window()** - provides the player with an opportunity to interact with the window.
g. **look_under_window()** - gives a clue to unlock the safe in the cupboard.
h. **look_outside_window()** - gives a brief description of the player's surroundings.
i. **cupboard()** - provides the player with an opportunity to interact with the cupboard.
j. **look_inside_cupboard()** - a mini passcode game
k. **open_safe()** - executed when the player enters the correct 4-digit code. Player gains a set of armour which may be useful in the battle afterwards.

Changelog:

19 Nov: (Lawrence Wong visit)
1. Started coding the game.

25 Nov:
1. Re-wrote the game from if-else loops into functions.

26 Nov:
1. Added dictionaries.
2. Finished the battle sequence part of the game.

3 Dec:
1. Bug fixes:
   a. Fixed stat returning to default when attacking monster
   b. Fixed crash when inputting stat larger than 100

4 Dec:
1. Added graphical sequence for failing to input correct value for player stats.
2. Added graphical sequence to the attack sequence function and to prompt the player to continue.

5 Dec (v1.00 - v1.02):
1. Added story and exploration prologue to the game.
2. Removed some dead code.
3. Bug fixes:
   a. Fixed game not ending when monster is defeated and looping into exploration..
      i. Added about 20-30 lines of 'return' to the end of if-else statements.

6 Dec (v1.02 - v1.03):
1. Limited the number of characters for name input to 10
2. Completed inventory choice to use 'red solution'
3. Bug fixes:
   a. Fixed graphical bug when monster is attacked (number of backslash characters)

7 Dec (v1.02 - v1.03):
1. Added "enter valid 4-digit code" elif for players who attempted to unlock the safe with an invalid input (i.e. entered < 4 or > 4 digits)
2. Removed some redundant 'continue' in while loops
3. Fixed game reset: Shifted dictionaries reset and counters reset to the start of the while loop in main_menu(player, monster, inventory).

8 Dec (v1.03 - v1.04)
1. Added dictionaries (player, monster, inventory) as parameters to almost every function to solve the resetting of inventory.
2. Added another condition under 'else:' to yes/no option when the player chose the <run> away option so that they will not go back to the main menu when entering anything but yes/no.

References:

The following references were used to clarify conceptual misunderstanding on Python's built-in functions and for 'bug' fixes. No user defined functions were used from these references.

1. Menus - Pygame Tutorial by DaFluffyPotato:
https://www.youtube.com/watch?v=0RryiSjpJn0&t=115s

2. Understanding the asterisk(*) of Python:
https://medium.com/understand-the-python/understanding-the-asterisk-of-python-8b9da aa4a558

3. Python methods, documentations and bug fixes (tuples, map(), format(), exit(), quit(), *, etc): docs.python.org, geeksforgeeks.org, stackoverflow.com, thispointer.com, treyhunner.com, w3schools.com, programiz.com

4. Text to ASCII Art Generator (TAAG):
http://patorjk.com/software/taag/#p=display&f=Big&t=Mush!

5. Ascii Art Creator (jpg to ascii): https://www.ascii-art-generator.org/

Appendix:

1. Function Tree Structure (for v1.01 bug fix):