Dungeon 2.0

For more practice with functions and validation, we will be doing a second version of the Dungeon Crawl game. You may still add monsters moving randomly if you would like.

Program Requirements

You need to implement the following features. I realize that this might cause you to have to change some of your design features, but this is to give you the specific practice that I am looking for. You should validate that all numbers are positive integers on input and trap any attempt to enter a non-numeric value.

constant.py	# this is a separate module used for declaring constants. # These include default parameter values, object symbols, etc.
displayInstructions()	# this function will display the instructions for playing this game
userDefinedSize()	# this function should return true if the user wants to define # the dungeon size
getSize()	# return a tuple for the user requested dungeon size
createMap(width, height)	# this function will create the 2D list for the map and return it to # main after it has been created . This can be called with default # values for the default size, or with custom values from getSize
validYesNo()	# should be used any place the user is required to select yes or no # should return true for yes and false for no.
placeTrap(map, numTraps) placeTreasure(map, numTreasure) placePlayer(map)	# The place functions should be called from within createMap # and should take the map and number of each thing being # placed into the map and place the appropriate number of # objects randomly into the map. Default parameters should # be defined in constant.py and specified in function definition
findEmpty(map)	# this method will be used in createMap. When called, it will # find a random location on the map that is empty and return # a tuple (row, column) for that location
findPlayer(map)	# this function will find the player location on the map and return # a tuple (row, column) of where it is
getMove(map)	# this function should get a move from the player and return a # tuple containing the new location of the player. # This function should only return a move after validating that it # is within the array. The player should also be allowed # choose the letter q to quit the game and return (-1, -1) to main
checkQuit(move)	# if the player has chosen to quit the game, this should be used to # terminate the game loop **without the use of break**

checkBounds(map, move)	# this function should be called inside of getMove and validate # whether or not the player has attempted to move outside of the # bounds of the list.

checkWin(map, move)
checkLose(map, move)
	# these functions should accept the map and move and test
	# whether the chosen move will cause the player to win or lose
	# the game. Should return true or false. Should not update the
	# the map. The result of these should be used to terminate the
	# game loop **without the use of break**

updateMap(map, move)	# this function should accept the map and move from main and # use them to update the Map. You should not update the map # unless not win and not lose.
playAgain()	# after the game is over, this function should see if the player # wishes to start again. Should return true or false. Should use # validYesNo. If the user wants to play again, start over with a # new dungeon