CO1102 CW2 (Simple Python Hangman) Report
By: Joey Groves and Sean Raisi

Function: is_secret_guessed(secret_word, letters_guessed):

This function is implemented correctly because as specified in Part 1, the function takes in 2 parameters: secret_word, and letters_guessed. The parameter secret_word is a string and letters_guessed is a list. This function first sorts the letters in the list 'letters_guessed' in alphabetical order using the sorted function. It then uses the lower function to change the secret_word into lowercase. An empty list is then initialised in order to get all the unique characters from secret_word (no duplicates). The following for loop then removes the duplicate characters from secret_word by checking each character in secret_word and appending it to the unique_secret_char list if it does not appear there. This is embedded within a try statement. The try statement, allows an appropriate error message to be displayed if there is an issue with the for loop.

Then the sorted function sorts the elements of the list into alphabetical order. The following nested if/else statements change the lists letters_guessed and unique_secret_char to sets and then compares the sets. So that if the set unique_secret_char is a subset of/is equivalent to letters_guessed then it will return Boolean True otherwise False is returned.

This function was tested by having print statements sporadically throughout the code in order to confirm that the code was running as expected. For example, after the for loop and if statement to check that the list unique_secret_char has no duplicates. It was also tested with the functions first_game and second_game. As these functions call the function is_secret_guessed in order to check whether the the secret word has been guessed by the user. In multiple tests the correct boolean values were returned meaning the function was acting correctly.

Test Plan for is_secret_guessed:

Test Data (secret_word, letters_guessed)	Expected Output	Actual Output	Status
("dog", ["h","d","o","g"])	Return True	Return True	
("dog", ["D","O","G"])	Return True	Return True	
("dog", ["d","a","y"])	Return False	Return False	

Function: first_game(secret_word)

The function first_game(secret_word) calls other functions such as secret_word() and is_secret_guessed() within the function itself. As well as, initialising the game itself, with the use of: while loops, inputs, string casting, try and except and if statements - the first_game(secret_word) function has been properly implemented to be able to get the user's input and output the result of the game according to the user's input guesses.

Moreover, the use of try and except statements, shown in line 94 and line 113, in the first_game() function have been properly implemented to handle invalid inputs and user input errors. For example, an assertion has been added to ensure that the 'user_guess_char' (which is an input variable) should be no longer than 1 character long. Otherwise, an exception is raised, which prints out an error message.

Similarly, an if statement is implemented in the function, which calls upon the function 'is_secret_guessed' (line 108) to check on whether or not the user has guessed the secret word. If true, then the while loop terminates, the win counter is incremented by 1 and prints out a message. In contrast, if the counter is equal to the value of the number_of_guesses variable, then a separate if statement is executed which prints out a message saying that the user has lost as well as incrementing the lose counter by 1 and setting the 'lose_Bool' variable to True.

Additionally, the testing of this function mainly consisted of using print statements within the code and a little bit of manual walkthrough, to see what values the functions were returning as well as seeing if the counters and boolean values outside of the while loop were updating.

Overall, the use of of try and except statements, if statements, and calling functions within the first_game(secret_word) function is properly implemented as these are the three main things that have been used to link other functions and code together, as well as initialising the game itself. All of which meets the criteria that was set out in 'Part 1' - in that the first_game function should take in one parameter (which it does) and start up an interactive game between the user and computer.

Test Plan for first_game(secret_word)

Test Data	Expected Output	Actual Output	Status
first_game("dog")	Secret_word = dog	Secret_word = dog	
Usr_guess_char = "cat"	Exception Raised: "String entered has a length greater than 1"	Exception Raised: "String entered has a length greater than 1"	
Count == number_of_guesses	lose_Bool = False and message printed: "You have ran out of guesses"	lose_Bool = False and message printed: "You have ran out of guesses"	