Hangman Testing

Game feature / function / input / process to be tested	Steps or notes on how testing was done to confirm fail/pass	Failed? Why? What changes made?	Passed?
Generate random *mystery_word* from *potential_words* list	When *potential_words* list is in the *choose_mystery_word* function, the *mystery_word* is correctly chosen from *“hello”, “bye”* or *“lamp”* and always one of these three options		Yes!
	When *potential_words* list is in the *words.py* file and we are using our small list, the *mystery_word* is correctly chosen from *“hello”, “bye”* or *“lamp”* and always one of these three options		Yes!
	Based on the above, when *potential_words* list is 5,000 words strong in the *words.py* file, it too seems to be working perfectly		Yes!
Notes on above:	This was through a STACK of trial and error, and by utilising only three words that are 3, 4, and 5 letters long, I could quickly ensure the *mystery_word* either was or wasn’t *“hello”, “bye”* or *“lamp”* and go from there		
Letters for the *mystery_word* and player *guess* are only alphabetical characters	I chose to define the alphabet as a variable called *letters*, shown as a set: *(‘ABCDEFGHIJKLMNOPQRSTUVWXYZ’)* This is threefold: • Converts our *mystery_word* into capital letters • I could use an *if/else statement* for if the player’s input wasn’t in our *letters* set, they would be prompted to try again - this therefore automatically covered anything that isn’t a single letter (numbers, symbols, multiple characters etc) • By applying the *.upper()* method to our player input, any letters are converted and then matched correctly	Tried the *string* package, but I couldn’t remove letters from that to see if they’ve been guessed or not (at least simply!), so went with typing out the alphabet instead	Yes!
Does the *mystery_word* get updated when a player guesses a correct letter?	This too was a lot of trial and error, generally with the trusty *“hello”, “bye”* or *“lamp”* words so I could be sure of what the *mystery_word* would be.		
	I would purposely type in either incorrect letters and expect that the underscores for each letter would stay as underscores, or correct letters and ensure that those underscores changed to beautiful big capital letters		Yes!
Ensuring the *chances* variable was correctly counting down	As above, when using my trusty *“hello”, “bye”* or *“lamp”* words, I could quickly type in correct or incorrect guesses purposely and know if my *chances* should be staying stable or ticking down		
	If I purposely entered a wrong letter, would the *chances* minus 1? To confirm this, the picture should change to reflect how many chances are left. That is, the corresponding ‘picture’ value to the current key in the *hanging* dictionary in the *hanging_man.py* file		Yes!
	If I entered a correct letter (or multiple in a row!) would my *chances* remain stable? To confirm this, I would enter a correct letter (or multiple), expecting no hanging man picture to print, and for the prompt to take another guess. I would then enter another incorrect letter, expecting the picture now printed to be one ‘level’ closer to ‘dying’		Yes!
Continue to play Python Hangman if player wants to, or exits program otherwise	By calling *intro()* first, and then having all of my game functions in the *main()* function, I could ensure that the game makes sense! *intro()* should only run the once when program is first opened, and *main()* continuing to run so long as the player says they want to.		Yes!
	I needed to have all the different game functions *choose_mystery_word() hangman_game(), did_he_die(),* and *outcome()* in the *main()* function, so that they would all run through correctly and in order every time the player selected they wanted to play		Yes!
	First question a player sees after giving their name is: *Do you want to play Python Hangman and save this fella? (y/n)* To get this to run correctly, I’ve used *.upper()[0]* on their input. As above, the *.upper()* method forces any letters into a capital. The *[0]* takes only the first character that is given.		Yes!
	This test was through - you know it! - trial and error: • If the player types ANYTHING that starts with an ’n’ (no, nope, nincompoop etc), the program will read that as a ‘no’, and give a nice goodbye and exit • If the player types ANYTHING that starts with a ’y’ (yes, yep, yippee, yeaaahhhhh buddy etc), the program will read that as a ‘yes’, and run through our functions to make the game work: *choose_mystery_word() hangman_game(), did_he_die(),* and *outcome()* • If the player types anything that doesn’t start with an ’n’ or a ‘y’ (a number, a symbol, any other letter), then they’ll just get a cheeky error message and asked for the yes/no again		Yes!
Notes on above:	I know that typing anything (nincompoop or yeaaahhhhh buddy in particular) might not be an actual ‘yes’ or ‘no’, but I felt it was more congruent with my fun/casual language in the program. If a player realises that anything starting with those letters will work, then they can feel like they won an extra mystery game 😉		
Does the game run in order/ make sense?	As mentioned above, I needed to call *intro()* first, and then having all of my game functions in the *main()* function, I could ensure that the game makes sense! If I had that as part of the *main()* function, then every time the player was asked if they want to play again, they would be asked for their name - it’s not overly friendly or smooth feel. You should only have to introduce yourself once!	Moved *intro()* out of *main()* function to ensure correct order of events	Yes!
With style code Pep8 and the long strings, do output printed lines make sense/look good?	That is, is the indentation all correct as viewed by the player? I had to just play with it a whole heap. Weirdly, you’re supposed to have indents to keep congruency with the first line, but can’t do it for multi-line strings or they will print out, which looks funny. Hence, this ended up being a lot of trial and error to get it to look good - but done!		Yes!
Is the *color* package installed and working correctly, and look right?	Do all the colours for the hanging man’s precarious state flow into one another and make sense? I played with the shades of red and orange a bit by purposefully ‘losing’ the game, until I settled on a nice gradient that felt like danger was escalating as chances ticked down	Had to ensure that box looked correct - ended up with spaces on either side and it looks better that way	Yes!
Is the *emoji* package installed and working correctly, and look right?	I tried to put *emoji* faces on the hanging man, but they never aligned correctly! They would always look weird and the body was off centre, so I opted instead to scatter them through the comments and prompts instead. In hindsight, glad because I think colours AND emojis would’ve been tacky for the hanging man picture.	emoji face didn’t look right for the man, so used them in comments instead	Yes!
Scorecards - getting this to work was a nightmare. An actual absolute nightmare.	Well, I tested this with trial and error and many tears. My three words helped speed this process up but it still took aaaaages to try and work through all options and make sure everything was covered. Basically, I checked things off as I went, made the changes needed (when I finally figured out how to do so), continued with testing, made changes again, etc etc:	Failed too many times to count. However, eventually passed everything. I am a genius (eventually).	
	In the except block, does a new player get added correctly?		Yes!
	Do their wins and losses accumulate correctly?		Yes!
	If we exit and reopen, we’ll be in the try block. If we introduce a new player, are they added correctly?		Yes!
	Do the original player’s stats still show?		Yes!
	Does the newest player’s wins and losses accumulate correctly?		Yes!
	If we exit and reopen with our original player, is there no extra player (ie. no duplicate)?		Yes!
	If so, do their new wins and losses accumulate correctly in addition to their previous scores?		Yes!
	And last but not least, does the csv file open in numbers (just to be sure?)		Yes!
	When the above is all done, reminder to self to breathe. Then cry with relief.		Yes!