SOFE 3980U
Software Quality
Winter 2022

Assignment-1: Choosing the right software process to build good-quality software, and ensuring quality using test automation (12.5 points)

Owais Quadri
100697281

Software Quality Assignment 1

Program

The program that I developed was a spinoff of the new viral internet game called Wordle. The name of my game is “Wordify”. The user must use 6 tries to guess a word that contains 6 letters

Development Process

The development process that was used was the incremental process. This entails the initial stage of planning, where I defined the requirements for the software as well as designed the outline. Then, the implementation process is where the development happens and then the development progress is tested for functionality and any failures. if there are any missing functionalities, then the process loops back to the planning process or else the program can go to deployment.

Planning

The planning process began by identifying the game that I would like to recreate. I decided for Wordle because I wanted to understand the complexity/simplicity of the program that got so popular for a simple idea. I extended the functionality by making the word length of guesses to be six letters instead of five. I also needed a word list to pick from so I used a list of 450 words and the word was randomized on each attempt instead of only selecting a certain word for a whole day. This encourages the user to make more than one attempt in a day, and enhances the replayability of the game.

Requirements

List of functional requirements:

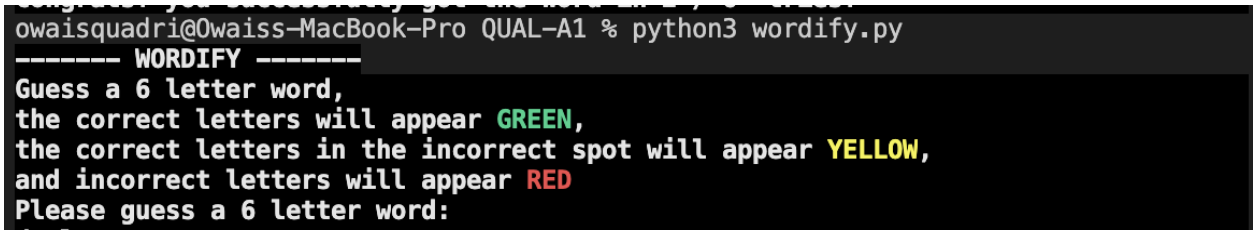
- R1: The user must be able to guess any six letter word but only a six letter word.
- R2: If the user guesses the correct word, a congratulations message will print and the program will exit.
- R3: The letters that are correctly guessed are made known to the user.
- R4: The letters that are in the incorrect position are made known to the user.
- R5: The letters that are not in the word are made known to the user.
- R6: The program will exit as an unsuccessful attempt when the user incorrectly guesses the word 6 times.

List of non-functional requirements:

- NR1: The status of the guessed letters have a visual cue (red for incorrect, green for correct, and yellow for incorrect position).
- NR2: On exit after an unsuccessful round, the program will reveal the correct word.

Implementation

I implemented the game in python by using the command line as an interface. The user is informed of the rules of the game when the program starts:



This was the first round of implementation, and basic completion of the user interface. Then, the next part of planning consisted of collecting a random word to be used for the correct word. This was implemented by using the 'random' library and selecting a random word from the list of words in 'wordlist.txt'. Once this was implemented, it was tested by printing the word on each run to make sure the words are different and randomized each time.

The next implementation is the implementation of the guess verification. First verification is whether the guess contains six letters. If the guess doesn't contain six letters, the attempt is invalidated and the user is able to guess again without spending a guess. Then, we iterate through the guessed word to check on the status of each letter. If the letter is correct, the letter is rendered green in the output. If the letter is in the incorrect position, the letter is rendered yellow. If the letter is incorrect, the letter is rendered as red.

In the end, if there is a correct guess within 6 guesses, a congratulations message is displayed or else the program exits and reveals the correct answer.

Testing

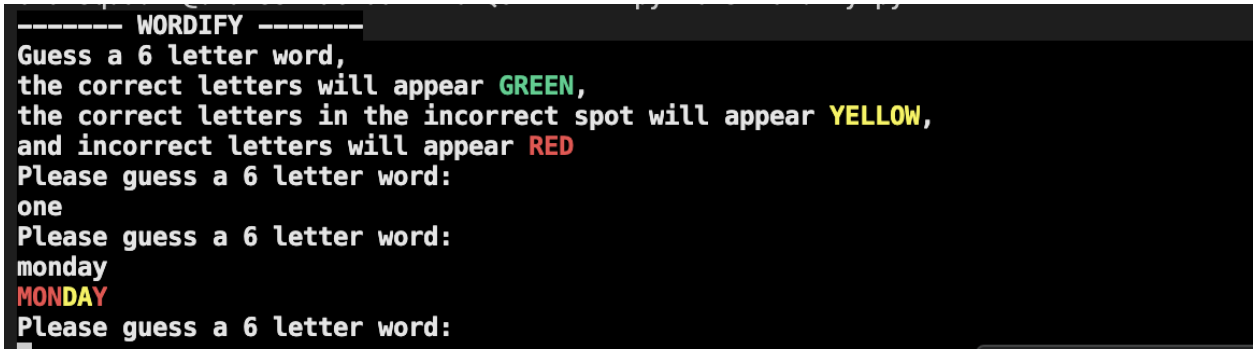
The testing was done by running the file to check for functionalities but I also implemented assert statements in my code as I wrote it:

Test Cases (screenshots will follow):

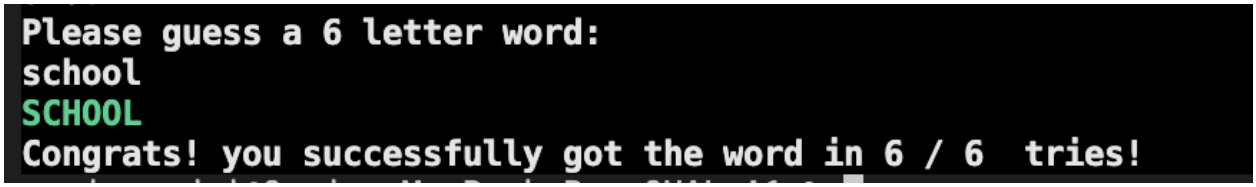
Test Case	Description	Corresponding Requirements	Result
T1	Ensure only 6 letters can be entered in each guess.	R1	PASS
T2	Ensure a congratulations message is displayed after successful completion.	R2	PASS
T3	ensure that the user knows of the correctly guessed letters	R3	PASS
T4	ensure that the user knows of the correctly guessed letters in the wrong position	R4	PASS
T5	ensure that the user knows of the incorrectly guessed letters	R5	PASS
T6	Ensure that the program will exit as an unsuccessful attempt when the user incorrectly guesses the word 6 times.	R6	PASS
T7	Verify that the status of the guessed letters have a visual cue	NR1	PASS
T8	ensure the program will reveal the correct word after an incorrect attempt.	NR2	PASS

Screenshots:

T1:

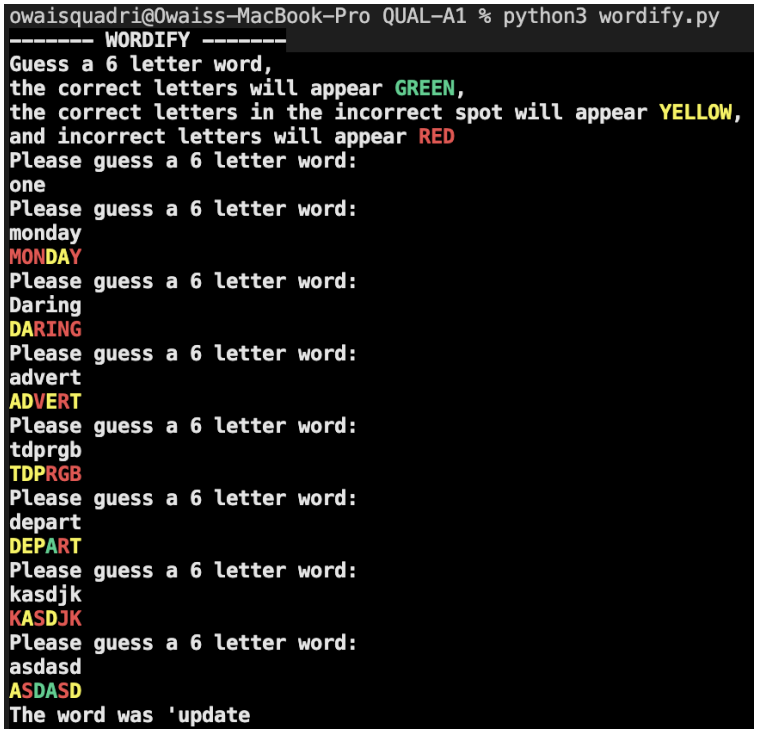


T2:



T3, T4, T5, T7:

T6, T8:



Challenges

One challenge that I faced during development was when I was moving from my desktop windows PC to a Mac. I ran into an issue where the runtime could not interpret inputs from the user. I realized that I needed to use the python3 command instead of just using the python command from windows.

```
owaisquadri@Owaiss-MacBook-Pro QUAL-A1 % python wordify.py
------- WORDIFY -------
Guess a 6 letter word,
the correct letters will appear as 'C',
the correct letters in the incorrect spot will appear as '/', and incorrect letters will appear as '_'
Please guess a 6 letter word: murder
Traceback (most recent call last):
  File "wordify.py", line 63, in <module>
    start()
  File "wordify.py", line 59, in start
    guess(0)
  File "wordify.py", line 41, in guess
    guessed_word = input("Please guess a 6 letter word: ").lower()
  File "<string>", line 1, in <module>
NameError: name 'murder' is not defined
```

Automated Tests

I integrated automated tests into my code using assert statements throughout the program. The tests automatically run as each unit is executed and will exit with errors if it fails.