The University of Texas
Computer Science

313E Programming Assignment 02
Wordle

Setup

This assignment may be completed individually or with a pair programming partner. Be sure to include your and your partner's name and EID in the Python file header. If you are working alone, you may delete one of them.

Complete these steps to prepare for the Assignment.

Check	Description
▢	Download wordle.py, secret_words.txt, and valid_guesses.txt. You will be working on the wordle.py file.
▢	Place all three files in the same folder/directory.
▢	You may not change the file names. Otherwise, the grading script will not work.

This programming assignment uses three files: wordle.py, secret_words.txt, and valid_guesses.txt. wordle.py contains predefined functions that you will be modifying. wordle.py will be able to read from the two text files, secret_words.txt and valid_guesses.txt. ***secret_words.txt*** contains a list of words that will be used to assign the secret word. ***valid_guesses.txt*** will be used to create a list of words that you will use to check for valid guesses. wordle.py will handle the entire logic of the game, which includes setting up the game, choosing the secret word, checking for valid guesses, and providing feedback for each guess.

A format called ANSI is used in this project to color the letters. ANSI escape codes are characters recognized by the terminal and interpreted as ways to color any characters that are surrounded by them.

Some starter code, such as opening files, may be unfamiliar to you as we have added them to comply with PEP8. We have included descriptions of what is happening, and you should use the same syntax provided for opening additional files. If you have questions about them, feel free to ask about them during office hours.

The colors in the VSCode terminal may be hard to see because of the font size. You can increase the terminal font size by going to the Settings editor by navigating to File > Preferences > Settings. Then, search for **terminal.integrated.font** in the search bar. Set the font size to your desired size.

If you would like to turn on autosave in VSCode, click on File -> Autosave.

Problem Description

Wordle

For this programming assignment, you will be creating the command-line version of the game Wordle. Wordle is a web-based word game in which players have 6 attempts to guess a 5 letter word. For each attempt, the player is given feedback in the form of colored tiles. A green tile indicates that the player has guessed a correct letter in the correct position, a yellow tile indicates a correct letter in the incorrect position, and a gray tile indicates an incorrectly guessed letter. If the player can correctly guess the word in 6 attempts or less, they win. If they do not guess the secret word after 6 attempts, they lose the game, and the correct word is revealed. In the official game, there is only one game of wordle per day. For this assignment, the game can be played as many times as the player wishes. You can find the official game here. If you have not played Wordle before, we recommend giving it a try to understand what you will accomplish in this programming assignment. Additionally, you can use this website to create your own Wordle for testing.

Requirements

Write a program that allows you to play Wordle in your terminal. You will modify each function to return or print something according to each function's purpose.

1. Your program should be able to initialize a game of wordle with the option to manually choose a secret word, utilize a seed, and/or read guesses from a text file. On user input error, you must print out the string in the constant INVALID_INPUT.
2. Your program uses the text file **valid_guesses.txt** provided to create your set of valid guesses, and you will create the functionality to randomly choose a secret word from the file **secret_words.txt**.
3. Your program should **only** accept and display lowercase input, **except** when the program is accepting a seed to set random. On user input error, you must print out the string in the constant INVALID_INPUT.
4. **You may not change the names of the functions listed. They must have the functionality as given in the specifications. You can always add more functions than those listed.**
5. **You may not import any additional external libraries in your solution except random and sys.**

You will modify the following functions to implement the functionality below.

prepare_game()

This function will set up the game by choosing the secret word and creating a **set** of valid guesses. The set of valid guesses is already created for you from the text file **valid_guesses.txt**; any word not in **valid_guesses.txt** is considered an invalid guess**.** The secret word will either be chosen from **secret_words.txt *OR*** it will be manually chosen, depending on the command that is entered. There can only be one argument provided to Wordle upon startup. Only a lowercase word **or** a number for the seed should be accepted. All other variations of arguments given to wordle.py are considered invalid, and the function should return **None** instead of a tuple of the secret word and valid guesses on user input error. Examples of this can be found in the **Input** and **Output** section.

get_feedback(secret_word, guessed_word)

This function will generate the proper feedback by coloring the guessed letters that were given. wordle.py contains 3 constant colors: `CORRECT_COLOR`, `WRONG_SPOT_COLOR`, `NOT_IN_WORD_COLOR`. It will always color a correct letter in the correct position `CORRECT_COLOR`, color the remaining correct letter(s) in the incorrect position(s) `WRONG_SPOT_COLOR` from left to right, and color an incorrectly guessed letter `NOT_IN_WORD_COLOR`. It will return the correct feedback based on these criteria.

It is important to note that if the player guesses a word containing a repeating letter, the colored letters in the feedback will only reflect the amount of letters contained in the secret word. Below are some examples to illustrate this.

In this example, the secret word is **"aback"**, and the player guessed "abaca".

```
Enter your guess: abaca
                  abaca
```

In this example, the secret word is **"shell"**, and the player guessed "hello".

```
Enter your guess: hello
                  hello
```

In this example, the secret word is **"aargh"**, and the player guessed "abaca".

```
Enter your guess: abaca
                  abaca
```

If you are green-yellow colorblind or prefer the high-contrast version, here is the alternative coloring that is available on the project. The top of the file wordle.py has instructions for how to enable it.

```
Enter your guess: abaca
                  abaca
```

In this example, the secret word is **"shell"**, and the player guessed "hello".

```
Enter your guess: hello
                  hello
```

In this example, the secret word is **"lever"**, and the player guessed "level".

```
Enter your guess: abaca
                  abaca
```

main()

This function will call *prepare_game()* to set up the game of Wordle. If setup fails and *prepare_game()* returns None, print out the INVALID_INPUT message and return instead of continuing with the game. It will loop for 6 **valid** guesses, display feedback for each valid guess, and it will stop once the player wins the game or runs out of guesses. On invalid guesses, the loop will not count it as an attempt and should print out the error message stored in the variable INVALID_INPUT ("Bad input detected. Please try again.").

Input

This project will use the command line to initialize a game of Wordle. **len(sys.argv)** checks the length of the command line arguments. Your program will start a game of Wordle with the given arguments in the command line as instructions. Your project should be able to initialize a game with each of the following commands:

```
python3 wordle.py
```
This example will initialize a default game of wordle. A secret word will be chosen randomly when launching wordle.py. The player will be able to manually guess words for 6 attempts.

```
python3 wordle.py 123
```
In this example, 123 is the number that will be set to random.seed() to choose the secret word. This means that your program will support having the random seed set (an integer) when launching wordle.py. The player will be able to manually guess words for 6 attempts, and the random word chosen should be the same every time with the same seed.

```
python3 wordle.py hello
```
In this example, we are manually choosing the secret word to be “hello”. The player will be able to manually guess words for 6 attempts.

```
python3 wordle.py 123 < guesses.txt
```
In this example, we are choosing the random seed and accepting an input file containing guesses. Each attempt to guess the secret word will be contained in the input file, meaning the user will not type guesses into the command line. It is guaranteed that the input file will have exactly six words. The input redirection will be handled by the shell (terminal); you will not have to implement this in your code. Alternatively, if you are using a Windows computer and are having trouble with input redirection <, you can run:

```
cat guesses.txt | python3 wordle.py
```

which will do functionally the same behavior.

```
python3 wordle.py hello < input.txt
```
In this example, the secret word is manually chosen and it accepts an input file of guesses. Each of the six attempts will be contained in the input file, meaning the user will not type guesses into the command line. It is guaranteed that the input file will have six words exactly. Alternatively, if you are using a windows computer and are having trouble with the redirection operator <, you can run:

```
cat input.txt | python3 wordle.py hello
```

which will do functionally the same behavior.

Output

The following commands are examples of what will be used in the test cases. The first output will be the standard coloring, and the second output will be the high-contrast coloring.

```
python3 wordle.py graph
```

Your output will look like the following:

Standard Coloring:

```
Welcome to Command Line Wordle!

How To Play
Guess the secret word in 6 tries.
Each guess must be a valid 5-letter word.
The color of the letters will change to show
how close your guess was.

Examples:
weary
w is in the word and in the correct spot.
pills
i is in the word but in the wrong spot.
vague
u is not in the word in any spot.

Enter your 1st guess: civic
                      civic
Enter your 2nd guess: 123ad
Bad input detected. Please try again.
Enter your 2nd guess: mamba
                      mamba
Enter your 3rd guess: oddly
                      oddly
Enter your 4th guess: those
                      those
Enter your 5th guess: snout
                      snout
Enter your 6th guess: sport
                      sport
Congratulations! You guessed the word 'sport' correctly.
```

High Contrast Coloring:

```
Welcome to Command Line Wordle!

How To Play
Guess the secret word in 6 tries.
Each guess must be a valid 5-letter word.
The color of the letters will change to show
how close your guess was.

Examples:
weary
w is in the word and in the correct spot.
pills
i is in the word but in the wrong spot.
vague
u is not in the word in any spot.

Enter your 1st guess: civic
                      civic
Enter your 2nd guess: 123ad
Bad input detected. Please try again.
Enter your 2nd guess: mamba
                      mamba
Enter your 3rd guess: oddly
                      oddly
Enter your 4th guess: those
                      those
Enter your 5th guess: snout
                      snout
Enter your 6th guess: sport
                      sport
Congratulations! You guessed the word 'sport' correctly.
```

```
python3 wordle.py 123
```

Your output will look like the following:

Standard Coloring:

```
Welcome to Command Line Wordle!

How To Play
Guess the secret word in 6 tries.
Each guess must be a valid 5-letter word.
The color of the letters will change to show
how close your guess was.

Examples:
weary
w is in the word and in the correct spot.
pills
i is in the word but in the wrong spot.
vague
u is not in the word in any spot.

Enter your 1st guess: flood
                      flood
Enter your 2nd guess: puppy
                      puppy
Enter your 3rd guess: batch
                      batch
Enter your 4th guess: grate
                      grate
Enter your 5th guess: smart
                      smart
Enter your 6th guess: tiara
                      tiara
Sorry, you've run out of attempts. The correct word was
'bleed'.
```

High-Contrast Coloring:

```
Welcome to Command Line Wordle!

How To Play
Guess the secret word in 6 tries.
Each guess must be a valid 5-letter word.
The color of the letters will change to show
how close your guess was.

Examples:
weary
w is in the word and in the correct spot.
pills
i is in the word but in the wrong spot.
vague
u is not in the word in any spot.

Enter your 1st guess: flood
                      flood
Enter your 2nd guess: puppy
                      puppy
Enter your 3rd guess: batch
                      batch
Enter your 4th guess: grate
                      grate
Enter your 5th guess: smart
                      smart
Enter your 6th guess: tiara
                      tiara
Sorry, you've run out of attempts. The correct word was
'bleed'.
```

```
python3 wordle.py !!!!!
```

Your output will look like the following:

```
Bad input detected. Please try again.
```

Grading

Visible Test Cases - 75/100 points

The program output exactly matches the sample solution's output. To receive full credit, the program must produce the correct output based on all requirements in this document and pass all the test cases.

Hidden Test Cases - 15/100 points

The program output exactly matches the sample solution's output. To receive full credit, the program must produce the correct output based on all requirements in this document and pass all the test cases.

Style Grading - 10/100 points

The wordle.py file must comply with the PEP 8 Style Guide.

If you are confused about how to comply with the style guide, paste the error or the error ID (e.g. C0116 for missing function or method docstring) in the search bar here in the Pylint Documentation, and you should find

The TAs will complete a manual code review for each assignment to confirm that you have followed the requirements and directions on this document. Deductions will occur on each test case that fails to follow requirements.

Submission

Follow these steps for submission.

Check	Description
▢	Verify that the code does not have a compile or runtime error. A compile or runtime error will result in a 0 for the assignment as a whole.
▢	Verify that you have no debugging statements left in your code.
▢	Submit **only** wordle.py (the starter code file with your updates) to the given assignment in Gradescope. This will cause the grading scripts to run.
▢	If you have a partner, make sure you are submitting it for you **and** your partner on Gradescope. See this Gradescope documentation for more details.
▢	When the grading scripts are complete, check the results. If there are errors, evaluate the script feedback, work on the fix, test the fix, and then re-submit the file to Gradescope. You can submit as many times as necessary before the due date. NOTE: By default, only the most recent submission

Check	Description
	will be considered for grading. If you want to use a previous submission for your final grade, you must activate it from your submission history before the due date.

Academic Integrity

Please review the Academic Integrity section of the syllabus. We will be using plagiarism checkers, and we will cross-reference your solution with AI-generated solutions to check for similarities. Remember the goal in this class is not to write the perfect solution; we already have many of those! The goal is to learn how to problem solve, so:

- don't hesitate to ask for guidance from the instructional staff.
- discuss with peers about **only** bugs, their potential fixes, design decisions, and project clarifying questions. Do not look at or copy another student's code when doing so.

Attribution

Thanks to Dr. Carol Ramsey for the instruction template.