COSC 101 Homework 5: Spring 2025

Introduction

In this assignment you will re-create a version Wordle using the following core concepts in Python.

1. Accumulator patterns
2. Functions
3. Loops
4. Conditionals

Important Tips

- Remember, computer science is a science. Always write code with a prediction in mind. While you can sparingly code to try and see output, you should focus on thinking through what you want your code to do, what you expect as a result, and compare what your code actually does to your result.
- Before you type, trace the execution of the starter code already provided
- Sometimes sub-tasks go together, like computation followed by printing. You should consider them simultaneously if that helps you follow the correctness of your work.
- Use extra steps if needed for your thought process, printing results to test your code. Just remember to remove these extra prints once you know your code is correct.
- Match your output to the output given. Precision is important in computer science.
- Don't change file names. You're generally given `.py` files to work in. Don't change the filenames of those files.

Your assignment

Your task is to complete following steps:

1. Download the `hw5.zip` file and open it. You will see three files: `hw5_wordle.py`, `test_words.txt`, and `sgb-words.txt` in the unzipped folder.
2. Complete all of the functions in `hw5_wordle.py`. Make sure to build on your bottom-up programming skills and test the fucntions as you go!
3. Submit your completed program.

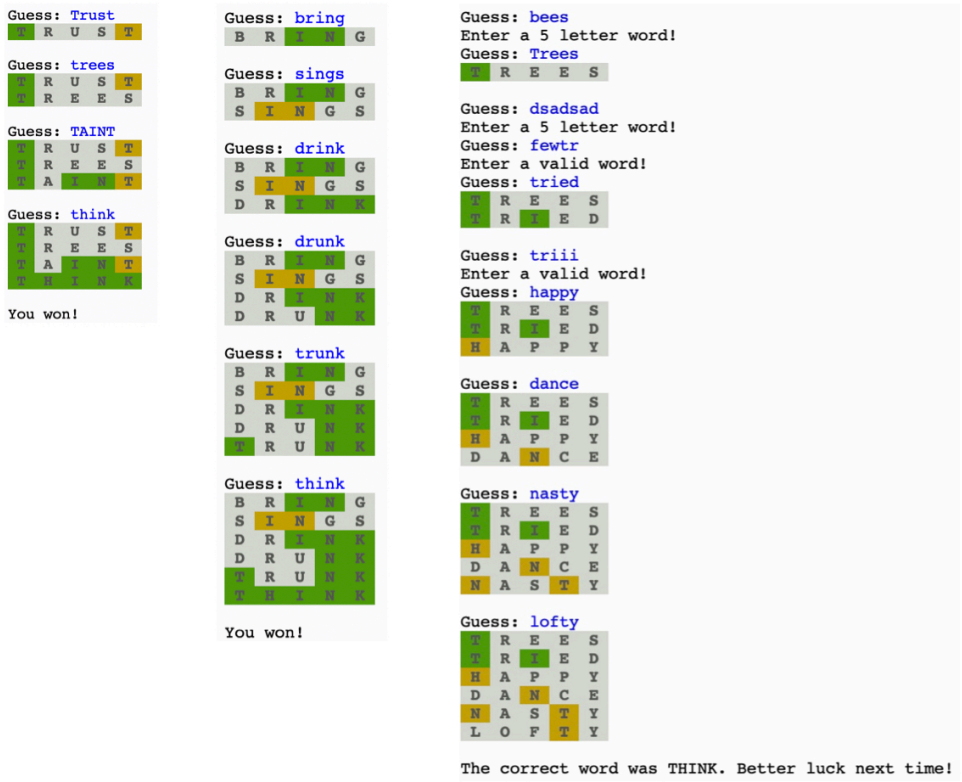
Required functionality

Here are the basic guidelines for how your program should behave:

1. The file `sgb-words.txt` (included with this homework) consists of 5757 lines, where each line consists of a single five letter word. Select a random five letter word from this file.
2. Ask the user for a word. Make sure that the guess is a valid five letter word from the list of words in the `sgb-words.txt` file. *Note: this part should be case-insensitive*. Note that `hw5_wordle.py` includes functions that makes words into uppercase or lower case.
3. Compare the `guess` to the selected word character by character. Each character in `guess` can have only one of three states: it is either fully correct, partially correct or incorrect. For example, consider a case where the original word is `THINK` and the guess is `STONE`. The letter `N` is fully correct because it occurs in the correct position. The letter `T` is partially correct because it is present in both the guess and selected word but in a different position. The letters `S`, `O` and `E` are incorrect.
4. Print the guess where the background of each letter indidates its state: green for fully correct, yellow for partially correct and white for incorrect. Note that the file `hw5_wordle.py` includes a function that takes in a character and a state (correct, partially correct or incorrect) and returns a formatted string based on the state. When you print this returned string, it will print the character with the correct background color.
5. Repeat steps 2, 3 and 4 until either the user guesses the word correctly or the user has had **six** guesses. If the user guesses the word correctly, you should print "You won!" and if the user has failed to guess the words after six tries, you should print "Better luck next time!".

Note the program should end after one game of Wordle has been played – just like the original game, the user gets only one word at a time!

Example outputs



If you are a Wordle aficionado you might notice that the expected behavior isn't exactly the same as real Wordle. Specifically, look at the first example when the guess "trust" is entered when the correct word is "think". In this case, the second "t" is marked as being partially correct even though there is only one "t" in the final word. We are going with this simplification because coding up the actual behavior can be tricky! If you are curious though, feel free to include a reflection about how you might implement the true Wordle behavior :)

Your tasks

Task A: Read existing functions

`hw5_wordle.py` has the following pre-defined functions

- `get_vocab`
- `format_char`
- `make_lowercase`
- `make_uppercase`

In this homework, you should understand *what the functions do* and *how to use* the functions, even if you do not fully understand how they are implemented.

Add one line at the top of the docstrings for each of these functions that describes what these functions do

Task B: Implement the core components

Using bottom up programming, implement the following core components of the wordle program.

- `get_guess`
- `check_guess`
- `format_guess`
- `play_game`
- `select_word`

Make sure to test each of these functions as you write them!

Task C: Compose the core components

Put the core components you implemented together in the `main()` function.

Submission Instructions

Submit `hw5_wordle.py` and `sgb-words.txt` to the platform indicated in your class section. Remember to complete the questions at the top of the file before submitting!