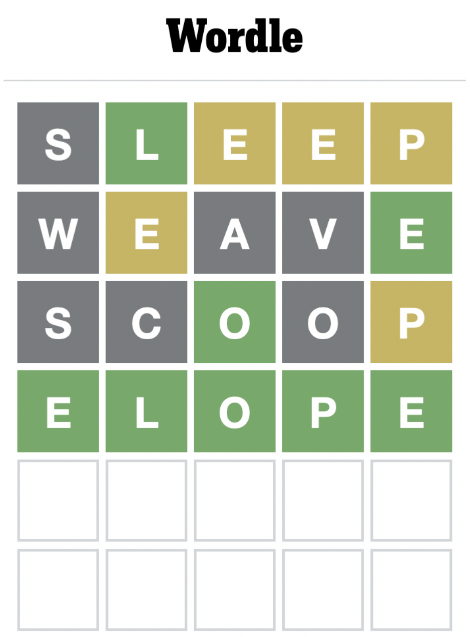
Homework 1: Wordish Front End

Due date: September 6th, 2022 at 11:59 pm

This homework is the first part of a three-part homework sequence in which you will build a simplified Wordle game, called "Wordish". For this first assignment, you will build only the front-end user interface for the game panel. In the later assignments, you will make the game work. For the second homework, you will make it playable in the browser, using JavaScript. For the third homework, you will implement the game and run it by sending web requests to a Django application running server-side.



The learning goals for this assignment are to:

- Use Git and GitHub to correctly submit the homework.
- Gain experience with a modern version control system (Git) and practice documented incremental development with Git commit (and commit messages).
- Create a front end for a small-scale application with the correct usage of HTML and CSS.

About the Wordle Game

In the actual Wordle game, a player tries to guess a five-letter "target" word. The New York Times newspaper selects a new target word every day. The player has up to six guesses to find the target word. The fewer guesses the better.

Not knowing what the target word is, a player will make a guess and the game will provide feedback on the correctness of the guess to help the user figure out the target word. Each guessed letter:

- in the target word and in the correct position will be displayed with a green background,
- in the target word but not in the correct position will be displayed with a yellow background,
- not in the target word will be displayed with a gray background.

A player wins the game if the target word is found in six or fewer guesses. (The player loses after making six incorrect guesses.)

About our Wordish Game

Our game will differ from the actual Wordle game in the following ways:

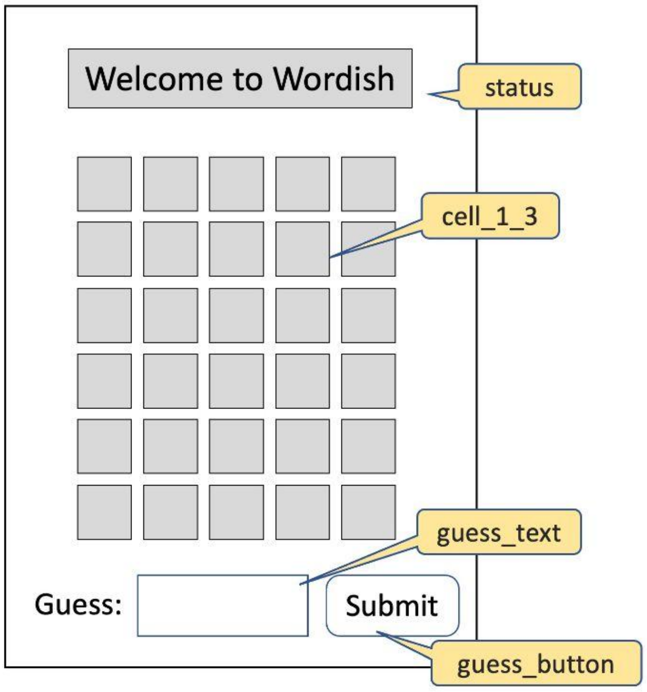
1. In the real game, the New York Times selects a new target word every day. When playing our

game, the user can select the target word at the start of the game before guessing, and our game can be played multiple times a day. This will simplify debugging and testing. (You may provide a feature for your game to choose a random target word, but we won't test this.)
2. In the real game, all guesses must be English words. In our game, the player may guess any five-letter string. (our game may reject guesses that are not English words, if you wish.)
3. The interface for our game will be different from the real game, but similar.
4. You will be required to use certain concepts presented in the lecture to implement our game.

Specification

In Homework #1, you will implement only the interface for your Wordish game. You do not need to worry about coloring different cells, we just want an initial layout of the interface with all cells empty. It must have the following features:

- A input field for the user to enter the guessed "word"
- A 6x5 matrix of blank cells to display the guesses
  - Cells must have a border
  - All cells must have the same background color (not white for homework #1)
- A labeled text field for entering guesses
- A button that will be clicked to submit a new guess
- A place to show game status (welcome/error messages, etc)
  - For this homework, the status area must display a "welcome" message (it must contain the word "welcome")



Requirements

Your submission must also follow these requirements:

- We will run automated tests on your code. To facilitate this, you must provide the following "id" attributes on your HTML elements:
  - Each display cell must have an `id="cell_`x`_`y`"`, where x is the row number (0 to 5) and y is the column number (0 to 4), as shown in the sketch above.
  - The input text field and button for submitting new guesses must have `id="guess_text"` and `id="guess_button"`, as shown in the sketch on the next page.
  - The area to show status messages must have `id="status"`.
- You must have some design and general theme used throughout your page using CSS and HTML. (I.e., your UI should look consistent in terms of layout, colors, etc.)

- The entire game must be visible on the screen without scrolling. (The AutoGrader's screen is 683 px high and 1366 px wide. AutoGrader will let you know if your game is too tall or wide.)
- You must put your files for this assignment in your repo in the `hw1` directory.
- You must submit one HTML file called `wordish.html`
- The CSS must be separated out into one CSS file called `wordish.css`.
- You may not use images in your game (either external files or embedded)
- You must write the CSS and HTML yourself. You may not include packages from other sources (such as Bootstrap, for example). You may not use JavaScript for this homework assignment.
- Your submission should look reasonably similar to the sketch above.
- Cite all external resources used and any additional notes you would like to convey to your grader in the `README.md` file.

The README.md file for this homework must have the following lines in it:

```
1) Andrew ID:
2) Name:
3) Homework: #1
4) Andrew IDs of students that I consulted on this homework:
5) Andrew IDs of course staff that I consulted on this homework:
6) Names of other people I consulted on this homework:
7) List of external sources I consulted when solving this homework:
8) Integrity Statement: I, <insert your name>, did not electronically copy any
   source code, in whole or in part, from any source, other than the course
   examples provided to me this semester, when preparing my solution for this
   homework assignment.
```

When answering the questions 4, 5, 6, and 7, above, you may indicate "None" if appropriate.

Committing your work

As you work you should demonstrate good use of Git, our version control system. By "good" we mean that you should commit small, incremental changes and use meaningful commit messages.

Incremental changes mean that each commit should focus on a single issue, feature, or addition to your solution. For example, if you implement five new features then it is poor to commit all your changes in a single large commit. Instead, you should commit five times, with each commit containing all the changes for a single feature.

Commit messages should consist of a brief one-line summary and an optional, more detailed explanation separated by an empty line. For example, one commit message might be:

```
Added CSS for rounded corners on the buttons.

TA's suggestion. TA is HCI major. :-)
```

The first line of the commit message is similar to an email subject, and the optional explanation is like the body of an email message. For more guidance on how to write good commit messages, see https://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html).

HTML & CSS Validation

While we are not strictly grading for style, we are grading for correct usage of the introduced technologies.

- You must run your solution through the W3 HTML validator (https://validator.w3.org/) and the CSS validator (https://jigsaw.w3.org/css-validator/) and have no errors.
- It is extremely bad practice to use inline CSS throughout a front end. This is why we require that you put your CSS into a separate file.
- You must use semantically correct tags. For example, use the `<button>` tag for buttons, not the `<div>` or the `<img>` tag. The area for status messages must use the `<div>` tag.

Turning in your work

Your submission should be turned in via Git and should consist of an HTML file called `wordish.html` and associated assets (CSS). Here is an example directory structure (some directories/files omitted) of a submission.

```
[YOUR-ANDREW-ID]/hw1/
  |-- wordish.html
  |-- wordish.css
  |-- README.md
```

Grading

You must run the grading script to get credit for this assignment. The grading script is available at:

https://grader.cmu-webapps.org

You may correct, resubmit, and regrade your homework, up to the deadline, to get all the points. You may submit it after the deadline, as per the homework late policy, in the syllabus.