Program Title: AI Wordle Solver

Wordle is a web-based word game created and developed by Welsh software engineer Josh Wardle. Each day, players have six attempts to guess a five-letter word. After each guess, the game provides feedback in the form of colored tiles: green for letters in the correct position, yellow for correct letters in the wrong position, and gray for letters not in the word at all. Wordle has a single daily solution, with all players attempting to guess the same word.

Here is a link to the actual game: https://www.nytimes.com/games/wordle/index.html

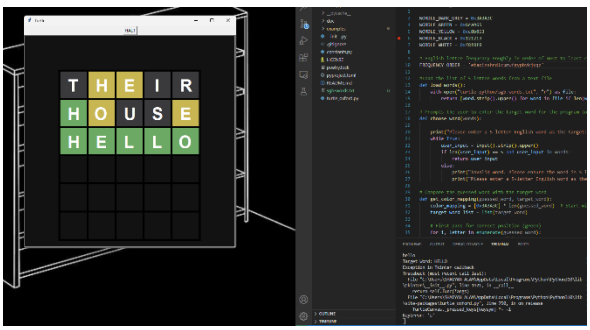
Interface of Wordle

The AI wordle solver streamlines the puzzle-solving process by applying algorithmic logic to make educated guesses. The significance of the program lies in its ability to demonstrate how artificial intelligence can replicate, and potentially enhance, human problem-solving skills. By leveraging classical AI techniques such as heuristic search and iterative learning, this program offers a glimpse into the practical applications of AI in our daily entertainment and beyond.

1. High-level implementation and setting up:

Upon launch, the program initializes by loading its dictionary from a text file, consisting of 5000+ five-letter English words. The repository of words is taken from Stanford's GraphBase. To ensure proper functioning, the text file, sgb-words.txt, should be placed in the **Code** directory, while the main python file is placed within the **examples** directory.

The program is understood best when the IDE window is minimized towards the right, as the canvas appears on the left. This allows the user to view each guess of the program properly.



Minimizing VSC to the right as the canvas pops under the VSC window.

The user is then prompted to input the target Wordle word into the terminal, which sets the stage for the AI to engage in a logical elimination process, narrowing down potential candidates with each guess based on the color-coded feedback provided by the Wordle game. The input word must be inside the text file for it to be validated.

```
Please enter a 5-letter English word as the target:
abcde
Invalid word. Please ensure the word is 5 letters long and is a valid English word.
Please enter a 5-letter English word as the target:
house
Target Word: HOUSE
```

Once the target word is fixed, the canvas will display the program's attempt at solving the wordle game.

The strategy involves selecting an initial guess from the list of five-letter words based on the frequency of each letter appearing in English words . After each guess, the program interprets feedback through a color-coded system—green for correctly placed letters, yellow for correct letters in incorrect positions, and dark grey for letters not in the target word.

The program continually narrows down the word list by preserving the order of green-marked letters and reshuffling the yellow-marked ones, avoiding repeats of grey-marked letters entirely.

2. Determining the Initial guess

The initial guess is based on an analysis of letter frequency. Using the FREQUENCY_ORDER constant, which reflects the commonality of letters in English, the AI selects a word from the word text file. It prioritizes words with frequently occurring letters to increase the likelihood of matching the target.

```
# English letter frequency roughly in order of most to least common
FREQUENCY_ORDER = 'etaoinshrdlcumwfgypbvkjxqz'
```

The letter frequency order is used to assign a score based on the presence of these letters, with the aim of selecting a word that has a high probability of containing correct letters. The scoring function, `score_initial_guess`, iterates through each word in the text file and tallies a score for each unique letter it contains, according to its position in the frequency order—the more common the letter, the higher the score it contributes.

```
frequency_scores = {letter: len(FREQUENCY_ORDER) - idx for idx, letter in enumerate(FREQUENCY_ORDER)}
```

If a letter is repeated in the word, the function penalizes this by subtracting a fixed amount from the total score, discouraging the selection of words with duplicate letters. This scoring system ensures

that the chosen word is likely to have a diverse and frequently occurring set of letters. After scoring all words, the one with the highest score is chosen as the initial guess.

Since there are no other constraints for the initial guess, this scoring system consistently selects "THEIR" as the initial guess.

3. Interpreting feedback and computing next guess

3.1 Feedback Interpretation:

After each guess, the `get_color_mapping` function interprets the game's color-coded feedback. This function assigns values to letters based on whether they are correct and in the right position (green), correct but in the wrong position (yellow) , or incorrect (grey). These findings are used to adjust future guesses.

Using the feedback, the `update_info function` records the confirmed location of correct letters, notes the presence of correct but misplaced letters, and discards any letters marked as incorrect.

3.2 Filtering words and computing guess

Afterwards, the `is_fits_criteria` function filters out any words from the words list that do not match the feedback. It removes all words that have incorrect letters in positions where correct letters (green) have been identified, words that include letters known to be absent (dark grey), and words that lack letters that have been indicated as present in the target word (yellow).

The resulting list contains only the words that meet all the criteria based on the feedback received so far.

The words in the filtered list are then taken and re-scored using the `score_filtered_words` function.

It again assigns points to each word based on the frequency of its letters. Then, it doubles the score for letters known to be in the word but not yet placed correctly (yellow). This process ensures that words with letters that are more likely to be in the target word are given priority.

A diversity bonus is also calculated for each word to favor guesses with a more unique mix of letters. This method allows the program to avoid duplicates, potentially revealing more information in the next round of feedback.

```
diversity_bonus = len(letter_set) / len(word)
word_scores[word] = score * diversity_bonus
```

Using the re-scored words, the `compute_guess` method selects the word with the highest score for its next guess.

This guessing process repeats, with the program continuously refining its filtered list of words words based on the game's feedback. This runs until it successfully guesses the word or runs out of attempts.

4. Conclusion and limitations

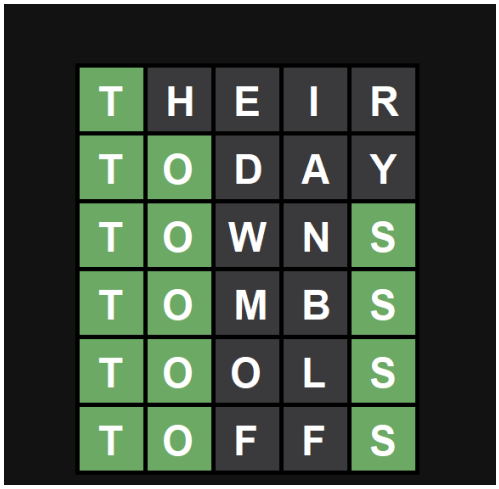
4.1 Limitations

The program encounters limitations, particularly in scenarios where only one or two letters of the target word remain unidentified, and the filtered list is still considerably big. In such cases, the AI tends to proceed through the remaining possibilities in a sequential manner (order in the list). This happens because its current scoring system (implemented in `score_filtered_words`) does not sufficiently differentiate between words that are almost identical except for one or two letters. The scoring heavily relies on letter frequency and the presence of letters, without a more nuanced strategy that could prioritize the most likely variations based on the specific context of the remaining unknowns.

Furthermore, while the `update_info` function effectively records which letters are correct, in the correct place, or absent, it does not dynamically adjust its strategy based on the closeness to the solution. For instance, when it becomes clear that only one or two letters need to be determined, the algorithm doesn't shift to a more aggressive or experimental guessing pattern which could explore different combinations more rapidly.

This results in the program being unable to guess a certain word within six attempts.

Attempt to solve for target word: DADDY



Attempt to solve for target word: TOUTS

4.2 Conclusion

The AI wordle solver utilizes statistical information about letter frequencies to improve the probability of correctly identifying letters in the target word and leverages patterns identified from the feedback to refine its guesses. The program adapts its strategy based on the outcomes of previous guesses, applying general rules, and learned strategies to optimize future responses. This decision-making process, grounded in heuristic principles, allows the program to make increasingly accurate guesses. This approach underscores the program's ability to use available data to systematically narrow down the solution space and converge on the correct answer, demonstrating a practical application of heuristic reasoning in problem-solving tasks.

Appendix

1. The use of different color constant to the ones already defined in the turtle package was done to emulate the actual colors of the wordle web game.

```
WORDLE_DARK_GREY = 0x3A3A3C
WORDLE_GREEN = 0x6ca965
WORDLE_YELLOW = 0xc8b653
WORDLE_BLACK = 0x121213
WORDLE_WHITE = 0xf8f8f8
```

2. The words text file taken from Stanford's GraphBase:
   https://www-cs-faculty.stanford.edu/~knuth/sgb-words.txt

   The file can be downloaded from this github repository:
   https://github.com/charlesreid1/five-letter-words/tree/master

3. The frequency order for all 26 letters across English words was taken from here:
   https://www3.nd.edu/~busiforc/handouts/cryptography/Letter%20Frequencies.html