# WORDLE

**Group Members and Roles:**
Ananna Biswas - *Test Engineer*
Karthik Garimella - *Programmer*
Riya Mole - *Documentation and Reviewer*
Sandeep Alfred - *Algorithm Designer*

**Objective: Convene as a team to discuss the foundational requirements your program must fulfill to solve the WORDLE problem effectively. Write down your answers to the following questions:**

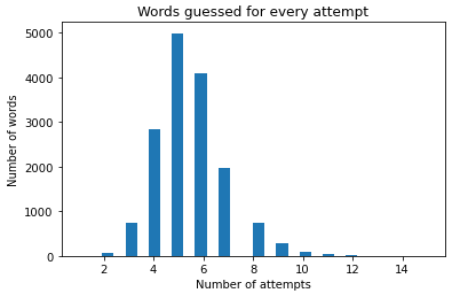
**What are the basic functionalities that a program must have to play WORDLE?**
Ans: The basic functionalities that a WORDLE game must have is an input where the user can make the 5-letter guess, the Green, Yellow and Grey highlight functionality in each of the boxes to determine if the guessed letter is correct and in the right position, exists in the word or does not exist at all. Another functionality would be to have a limit of 6 guesses to the player.

**How should the program represent the state of the game, especially the feedback after each guess?**
Ans: The program should re-adjust the possibilities of the word being the correct guess after each iteration/guess. The feedback helps the program in understanding which word is required, exists in the word or just does not exist at all. Each guess lets the program know which letters to omit from the words to be guessed in the next iteration.

**What type of decision-making algorithm or heuristic should be used by the automated agent?**
Ans: The decision-making algorithm that could be used by the automated agents could be calculating weights for each letter in each position and assigning an absolute weight value to each letter in the 5 lettered word. The first guess would always be the same with the highest weight and after the feedback given by the algorithm, it makes the 2nd guess with the highest weight. This continues until the correct word is guessed by the algorithm.

**Assessment and Challenges**
**Evaluate the performance of your agent by noting the number of attempts it typically requires to guess the correct word.**

- The code logic was tested for all the values of the dataset. The agent was able to guess the words in 6 attempts 80% of the time. The range of guesses lies in the interval 1 to 15.
- Following is the analysis of the test data.



**Question 1: Function Implementation: Describe the role of the guess_word function in the code. How did you implement the logic to provide color-coded feedback?**

Ans:

- Based on the feedback received after each guess the *guess_word(feedback, previousGuess, gs_ys)* function updates the *DF_GUESS* dataframe with the new possible words.
- *feedback* argument is a list of letters with either G, R, or Y for each position of the *previousGuess*. Example: [‘G’, ‘G’, ‘R’, ‘Y’, ‘R’]
- We iterate through the feedback received for each letter of the previous guess, *focus_letter* denotes that letter.
- If the *focus_letter* color is green, the dataframe is updated with all the words having the *focus_letter* in the current position of the iteration.
- If the *focus_letter* color is yellow, the dataframe is updated with all the words containing the *focus_letter*, and all the words containing the focus letter in the current position are removed from the dataframe.
- If the *focus_letter* color is red and the letter is also marked green or yellow, all the words containing the *focus_letter* only in the current position are deleted. Otherwise, if the *focus_letter* is red and is not marked green or yellow, all the words containing the *focus_letter* are deleted.
- Weights are again calculated for the newly updated dataframe which is based on the previous guess.

**Question 2: File Handling: Explain the steps involved in reading the word list from the text file and selecting a random word. Were there any challenges you encountered in file operations?**

Ans:

- Using the built-in python function open(), the file is opened up and each word in the text file is read using the read() function.
- The splitlines() function helps in splitting and considering each line as a new input. The words which are of length 5 are only considered to be in the words list. To select a random word from the words list, random module was imported to select a random word from the word list.
- The challenge we faced was on how to split the words as they were in different lines but using splitlines(), we could complete the process of breaking the text into isolated words.

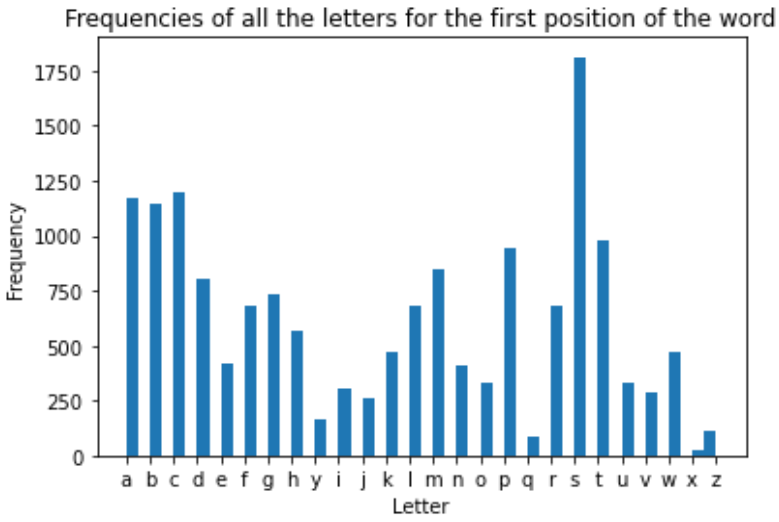
**Question 3: Loops and Conditionals: How did you implement the loop to limit the number of guessing attempts to six? What conditional statements did you use to break out of the loop?**

Ans: The loop was limited to 6 attempts by looping attempt as the iterator till range(6) i.e. 0 - 5. The break statement was used if the guessed word matched the randomly selected word, it says "Congrats! You got the correct word" or if the 6th attempt was complete and the program still did not find the correct sequence of letters, it says "End of Game! Boo!".

**Question 5: Heuristic Design: Describe the heuristic or algorithm your team decided to use in the agent function for making guesses. What was the rationale behind this choice?**

Ans:

- The algorithm choice was based on the probability of each letter in each position of the word.
- We analyzed the words dataset *words_alpha.txt*. Our findings saw specific letters to occur more frequently in certain positions.
    - Example: The following graph shows the frequencies of all the letters for the first position of the word.




- We decided to calculate the total frequency of each word and sort the dataframe with the word that has the highest cumulative frequency.
- This is so that the word that has the highest total frequency for each letter in each position is our first guess.

**Question 6: State Representation: How did you represent the state of the game, especially the feedback received after each guess, within your program?**

Ans:

- The state of the game is represented by the dataframe *DF_GUESS*.
- The total frequency of each word is calculated and the dataframe is sorted with the word which has the highest cumulative frequency on the top.
    - The dataframe with the calculated weights is shown below.

| | words | letters | letter_1 | letter_2 | letter_3 | letter_4 | letter_5 | letter_1_freq | letter_3_freq | letter_4_freq | letter_2_freq | letter_5_freq | CumiWeight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | sanes | [s, a, n, e, s] | s | a | n | e | s | 1813 | 1238 | 2510 | 2871 | 3148 | 11580 |
| **1** | sales | [s, a, l, e, s] | s | a | l | e | s | 1813 | 1061 | 2510 | 2871 | 3148 | 11403 |
| **2** | sores | [s, o, r, e, s] | s | o | r | e | s | 1813 | 1545 | 2510 | 2281 | 3148 | 11297 |
| **3** | cares | [c, a, r, e, s] | c | a | r | e | s | 1196 | 1545 | 2510 | 2871 | 3148 | 11270 |
| **4** | bares | [b, a, r, e, s] | b | a | r | e | s | 1141 | 1545 | 2510 | 2871 | 3148 | 11215 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **15915** | ewhow | [e, w, h, o, w] | e | w | h | o | w | 421 | 208 | 903 | 174 | 94 | 1800 |
| **15916** | iddhi | [i, d, d, h, i] | i | d | d | h | i | 301 | 514 | 288 | 136 | 509 | 1748 |
| **15917** | zmudz | [z, m, u, d, z] | z | m | u | d | z | 112 | 787 | 545 | 233 | 54 | 1731 |
| **15918** | enzym | [e, n, z, y, m] | e | n | z | y | m | 421 | 143 | 161 | 557 | 297 | 1579 |
| **15919** | oghuz | [o, g, h, u, z] | o | g | h | u | z | 334 | 208 | 686 | 102 | 54 | 1384 |

15920 rows × 13 columns

- The *initialise_dataframe(wordle_list)* function creates the dataframe and assigns it to the *DF_GUESS* global variable for easy access across the program.
- The *claculate_weights()* function calculates the weights of the newly generated dataframe each time changes are made to it.
- The dataframe keeps its state changing with every guess, it has lower rows as values get deleted that are not related to the word.

**Question 7: Agent Behavior: What are the strengths and weaknesses of your agent? Is it capable of solving the WORDLE problem within the stipulated six attempts consistently?**

Ans:

- One of the strengths of the agent is that it can guess 80% of the words in the given text file in or under 6 attempts.
- Another strength is it can handle double letter guesses and not present the 2nd instance of the letter as yellow even though that letter exists in the word already.
- Without this check in the code, a letter that can have another instance in the word can show up as yellow saying it exists in the word but not in the correct position although that specific letter is present in the word marked in green color.

- One of its weaknesses is not being able to guess the words which consist of higher and lower weights in each letter position simultaneously.
- This can cause issues while trying to narrow down the letters.

**Question 8: Decision-Making: Explain the decision-making process of your agent. How does it evaluate the feedback from previous guesses to make a new guess?**

Ans:

- The agent gets feedback on its initial guess, and based on this it deletes all the letters which are not related to the word and follows the guess_word function. The *guess_word(feedback, previousGuess, gs_ys)* function updates the *DF_GUESS* dataframe with the new possible words.
- *feedback* argument is a list of letters with either G, R, or Y for each position of the *previousGuess*. Example: [‘G’, ‘G’, ‘R’, ‘Y’, ‘R’]
- We iterate through the feedback received for each letter of the previous guess, *focus_letter* denotes that letter.
- If the *focus_letter* color is green, the dataframe is updated with all the words having the *focus_letter* in the current position of the iteration.
- If the *focus_letter* color is yellow, the dataframe is updated with all the words containing the *focus_letter*, and all the words containing the focus letter in the current position are removed from the dataframe.
- If the *focus_letter* color is red and the letter is also marked green or yellow, all the words containing the *focus_letter* only in the current position are deleted. Otherwise, if the *focus_letter* is red and is not marked green or yellow, all the words containing the *focus_letter* are deleted.
- Weights are again calculated for the newly updated dataframe which is based on the previous guess.

**Question 10: Collaboration: How did your team divide the tasks? Were there any challenges in collaboration and if so, how were they resolved?**

Ans:
The group roles:

- Ananna Biswas - *Test Engineer*
- Karthik Garimella - *Programmer*
- Riya Mole - *Documentation and Reviewer*
- Sandeep Alfred - *Algorithm Designer*

The roles were flexible and it did not lead to any significant challenges.

**Question 11: Learning Outcomes: What are the key takeaways from this assignment regarding Python programming, artificial intelligence, and teamwork?**

Ans:
The key takeaways from python programming are understanding how to open a file, split each word and manipulate which words we want. Using dataframes, we learnt how to plot density of the letters in each position, using functions to make the code cleaner and learning how to color the output with respect to the conditions it satisfies.
Regarding teamwork, the takeaways are that collaboration helps in getting teh best out of each other's output on how to solve a particular problem. Everyone's opinion gets molded into a collaborative algorithm. Another takeaway being how effective teamwork can be.