

# What is Wordle?



Wordle is a daily online word game created by Josh Wardle and later sold to The New York Times Company. Every 24 hours, there is a new word of the day, and we have to figure it out in 6 tries. Since its launch, the game has become very popular. According to The New Times, over 300,000 people play the game daily.

## How to play?

Guess the **WORDLE** in six tries.

Each guess must be a valid five-letter word. Hit the enter button to submit.

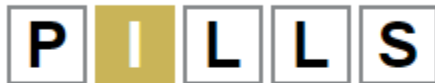
After each guess, the color of the tiles will change to show how close your guess was to the word.

---

### Examples



The letter **W** is in the word and in the correct spot.



The letter **I** is in the word but in the wrong spot.



The letter **U** is not in the word in any spot.

Image source: <https://www.nytimes.com/games/wordle/index.htm>

# Problem statement:

Find a strategy to guess the right word in Wordle in the least number of guesses.

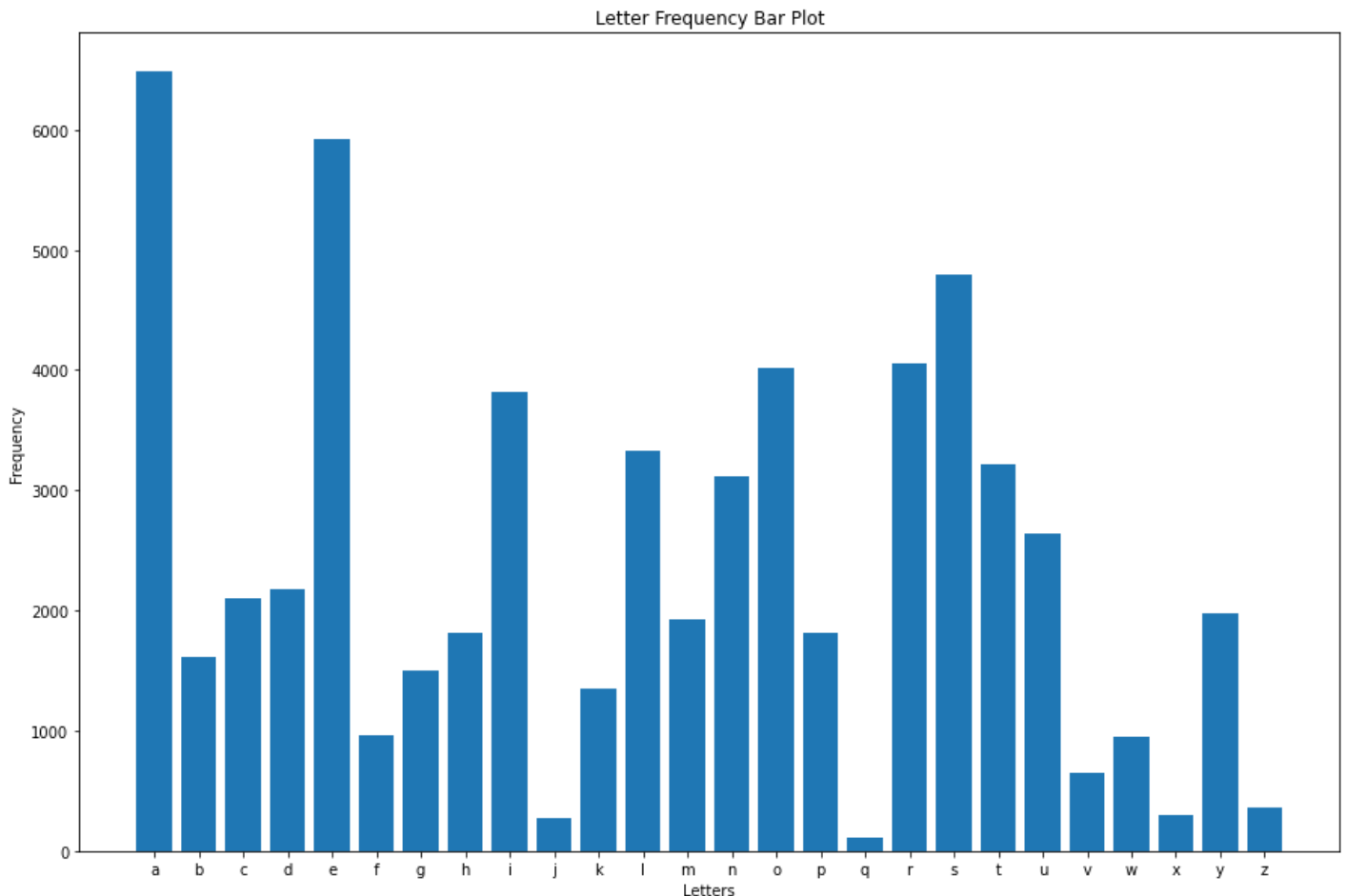
## Methodology:

1. Downloading a database of 5-letter words.
2. Analyze these words
3. Propose a strategy based on the analysis
4. Testing our strategy.
  - a. Write a program to guess the correct word using our strategy.
  - b. Apply the program for every word in the database and take the average number of guesses.
  - c. Develop a program to apply an algorithm on the Wordle website to guess the word.

## Letter Frequency Analysis

Used Python to analyze a database of 5-letter words. Then, this data was used to plot graphs, which helped further understand the data.

Here, the frequency of each letter occurring in all words.



# Strategy

Based on the analysis done on our database, I developed strategies to guess the right word in the least number of guesses and within 6 guesses. First, pass fixed words as initial guesses and then the code will find the correct position, wrong positions and letters not present in the word. Based on those details, the code will decide the next guess.

## Method -1:

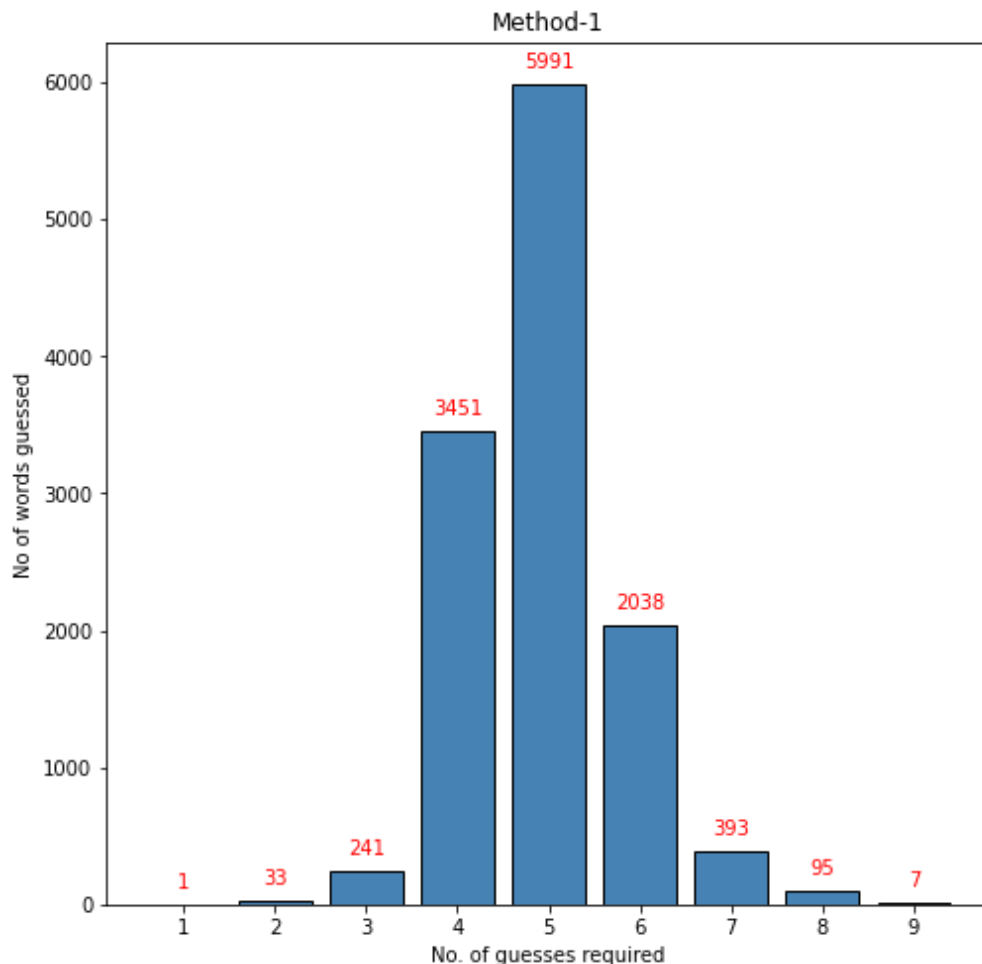
Let's choose the first 5 most occurring letters:

a e s r o

Make one word from it: {**AROSE**}

So, the first initial guess will be the word 'AROSE' each time, and then the code will provide further guesses.

**Result:** After applying this algorithm for all words, the output is as follows.



### Method -2:

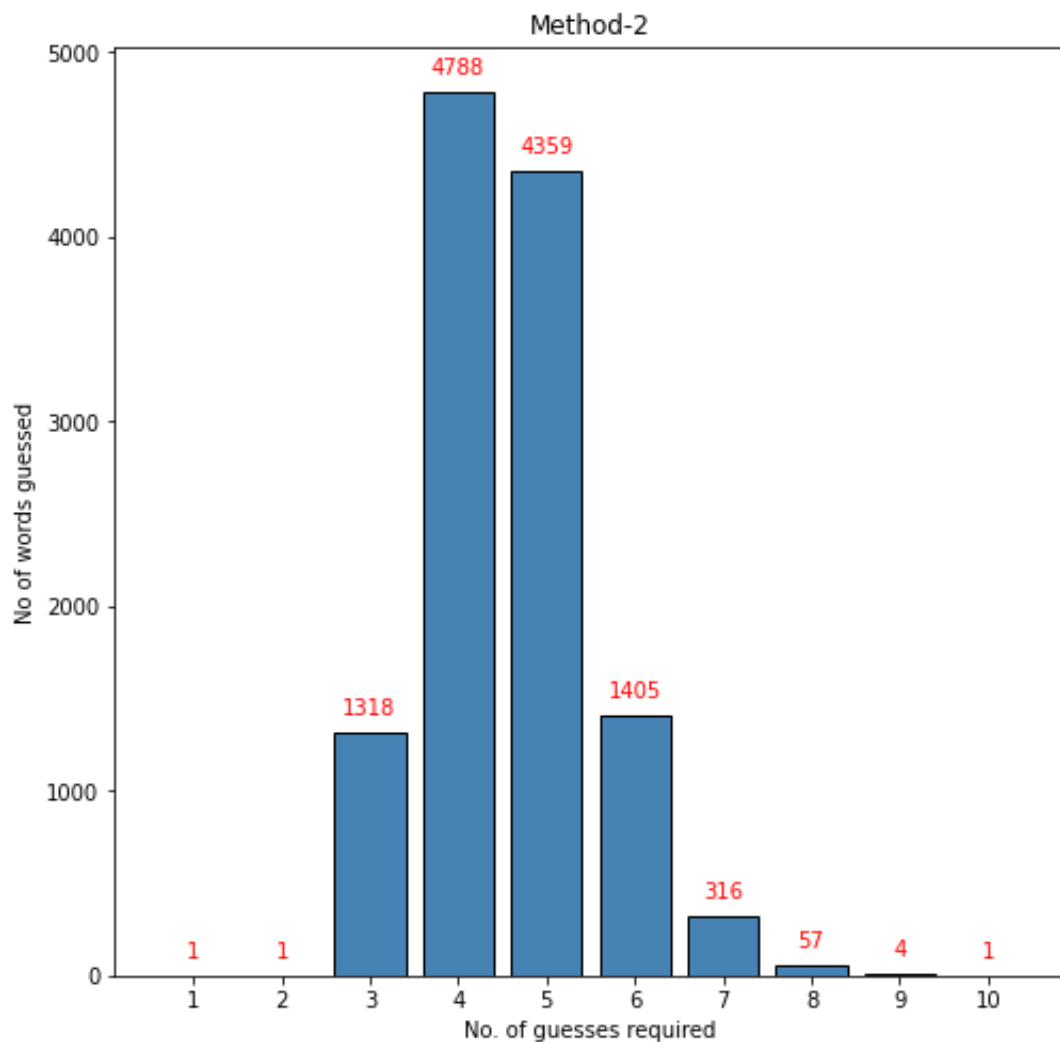
Let's choose the first 10 most occurring letters:

a e s r o i l t n u

Make two words from it: {**SAINT**, **LOURE**}

So, the first initial guess will be the word 'SAINT', and the second guess will be the word 'LOURE' each time, and then the code will provide further guesses.

**Result:** After applying this algorithm for all words, the output is as follows.



### Method -3:

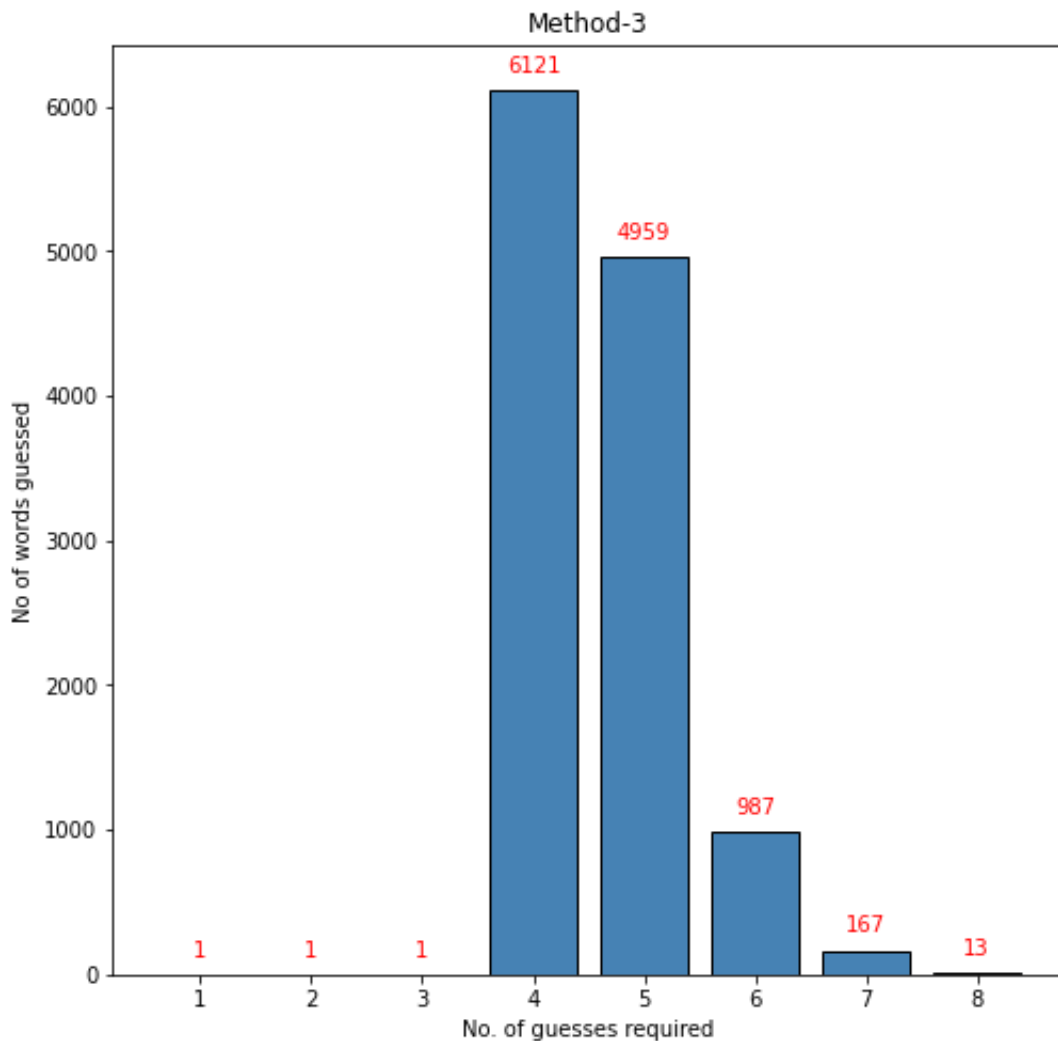
Let's choose the first 15 most occurring letters:

a e s r o i l t n u d c y m p

Make three words from it: {**CANDY**, **PLUME**, **RIOTS**}

So, the first initial guess will be the word 'CANDY', the second guess will be the word 'PLUME', and the third guess will be the word 'RIOTS' each time, and then the code will provide further guesses.

**Result:** After applying this algorithm for all words, the output is as follows.



#### Method -4:

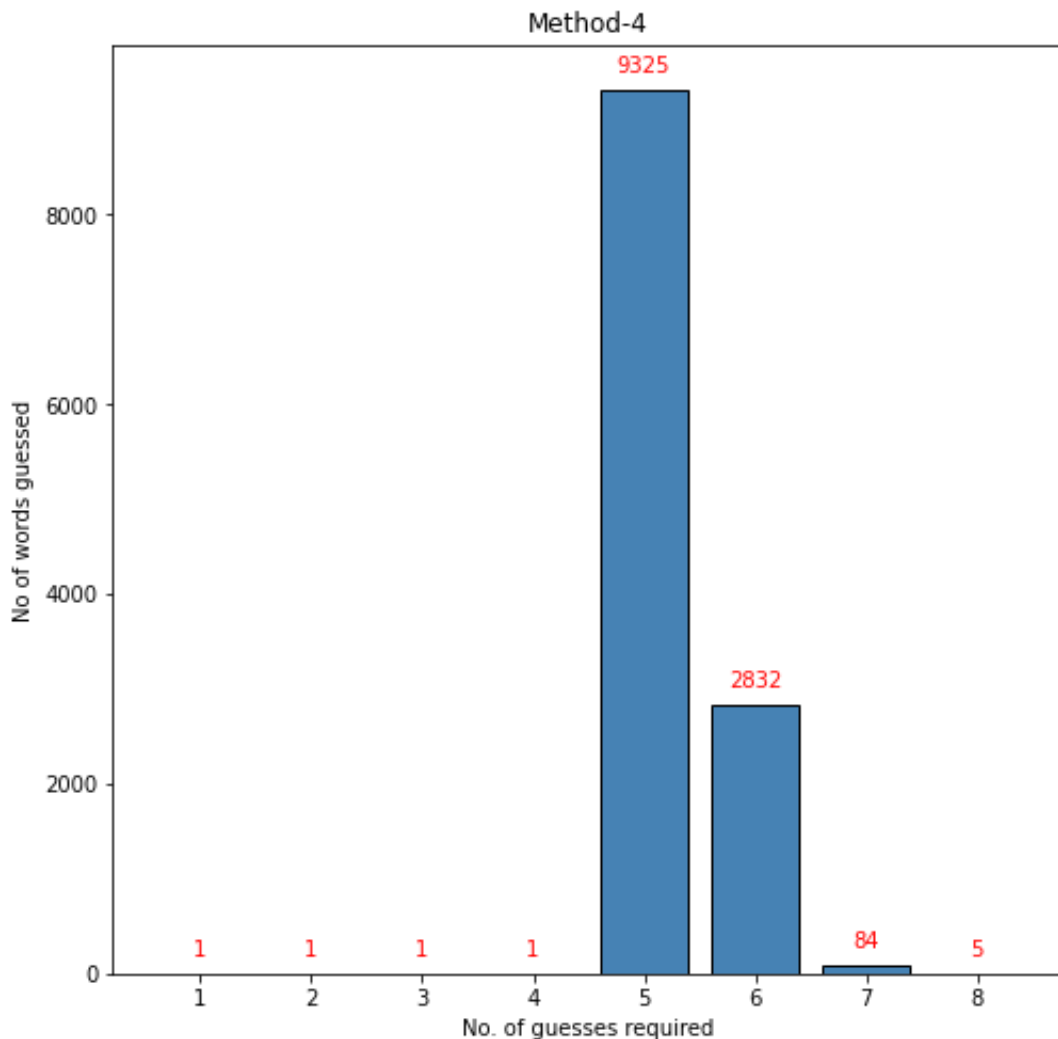
Let's choose the first 20 most occurring letters:

a e s r o i l t n u d c y m p h b g k f

Make four words from it: {**SHELF**, **DIRTY**, **BACON**, **GRUMP**}

So, the first initial guess will be the word 'SHELF', the second guess will be the word 'DIRTY', the third guess will be the word 'BACON', and the fourth guess will be the word 'GRUMP' each time, and then the code will provide further guesses.

**Result:** After applying this algorithm for all words, the output is as follows.



### Method -5:

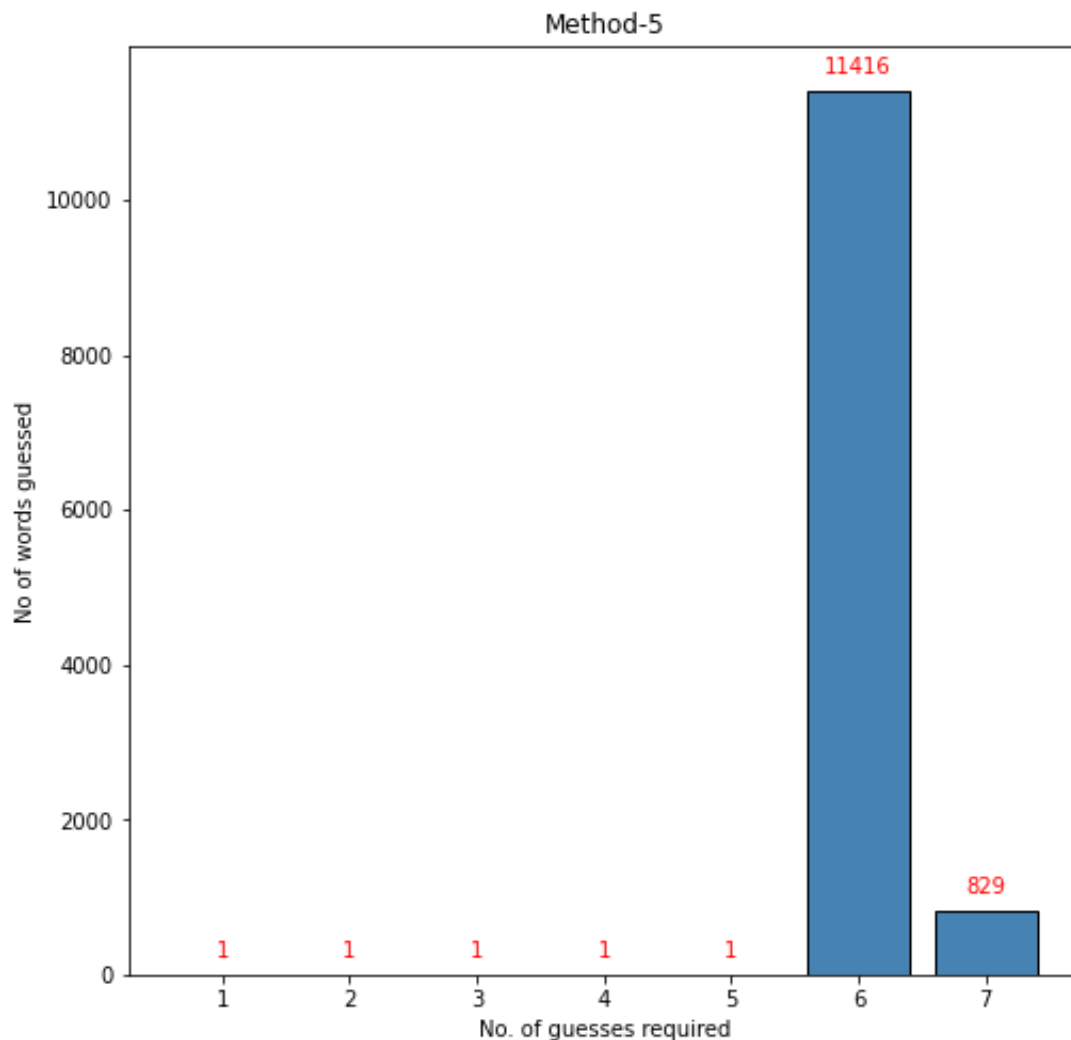
Let's choose the first 25 most occurring letters:

a e s r o i l t n u d c y m p h b g k f w v z x j

Make five words from it: {**FJORD**, **GUCKS**, **NYMPH**, **VIBEX**, **WALTZ**}

So, the first initial guess will be the word 'FJORD', the second guess will be the word 'GUCKS', the third guess will be the word 'NYMPH', the fourth guess will be the word 'VIBEX', and the fifth guess will be the word 'WALTZ' each time, and then the code will provide further guesses.

**Result:** After applying this algorithm for all words, the output is as follows.



# Analysis

Method	Mean of guess count	Variance of guess count	Maximum guess count	No. of words guessed with guess count > 6
1	4.93	0.75	9	495
2	4.58	0.91	10	378
3	4.61	0.5	8	180
4	5.25	0.2	8	89
5	6.07	0.07	7	829

So, our mathematical model will be:

$$Loss = W_1 X_1 + W_2 X_2 + W_3 X_3 + W_4 X_4$$

$X_1$  : *Mean Guess Count*

A lower mean guess count means the method is more efficient.

- If the mean is high, the loss increases.
- If the mean is low, the loss decreases (better method).

$X_2$  : *Variance*

A lower variance means the method is consistent.

- High variance → High penalty (method is unpredictable).
- Low variance → Lower loss (method is stable and reliable).

$X_3$  : *Penalty for Max Guess Count, (max(0, Max Guess Count - 6))*

If a method sometimes requires more than 6 guesses, it fails the goal.

- If the max guess count is 6 or below, this term becomes 0 (no penalty).
- If the max guess count is greater than 6, it increases the loss (penalizing bad methods).

$X_4$  : *Penalty for Words guessed with guess count > 6*

If too many words take more than 6 guesses, the method is unreliable.

- More words >6 guesses → Higher penalty.
- Fewer words >6 guesses → Lower loss (better method).

Choosing the right weights depends on **how much importance** you give to each factor.

Our primary goal is:

1. **Guess within 6 guesses** (minimize failures beyond 6).
2. **Use as few guesses as possible** (minimize the mean guess count).



Factor	Importance	Weight
Mean Guess Count (lower is better)	Medium	w1 = 1.5
Variance (stability)	Low	w2 = 0.3
Max Guess Count (beyond 6)	Very High	w3 = 2
Words needing >6 guesses	High	w4 = 0.5

So, new loss function becomes:

$$Loss = 1.5X_1 + 0.3X_2 + 2X_3 + 0.5X_4$$

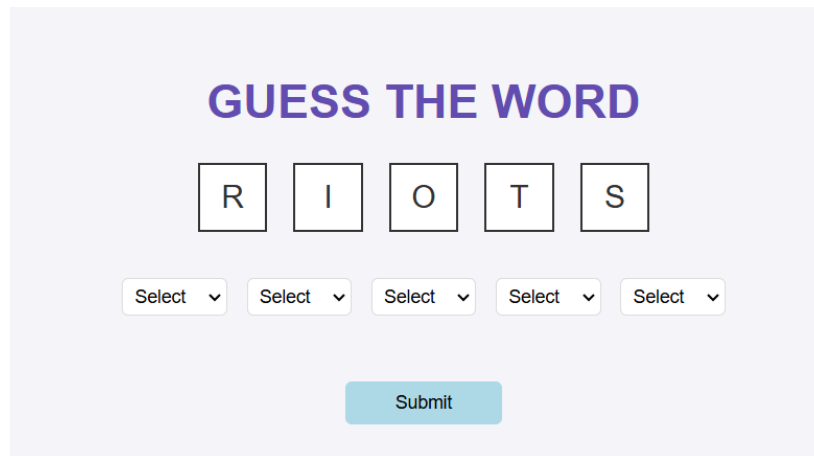
First, we will normalize our data and then calculate loss for each method gives:

Method	Mean of guess count	Variance of guess count	Maximum guess count	No. of words guessed with guess count > 6	Loss
1	0.2349	0.8095	0.6667	0.5486	2.2029
2	0	1	1	0.3905	2.49525
3	0.0201	0.5119	0.3333	0.123	0.91182
4	0.4497	0.1548	0.3333	0	1.38759
5	1	0	0	1	2

From above result, we can conclude that Method-3 has less loss hence it is most optimized method. I have developed a **web application** with a **Python backend** designed to assist in solving Wordle efficiently.

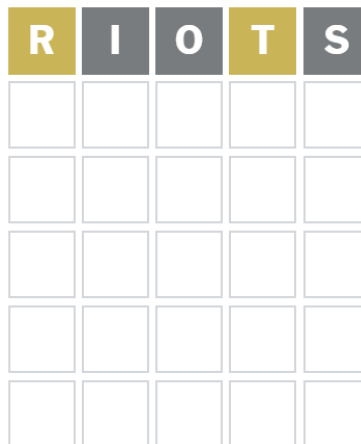
# Wordle-solver Web Application

**Step - 1:** The first code will give an initial guess of ' RIOTS'.



The screenshot shows the 'GUESS THE WORD' interface. At the top, the title 'GUESS THE WORD' is displayed in purple. Below it, the letters 'R', 'I', 'O', 'T', and 'S' are each in a white square. Underneath each letter is a dropdown menu with the text 'Select' and a downward arrow. At the bottom center is a light blue 'Submit' button.

**Step - 2:** Then we will enter the word ' RIOTS ' on the webpage of the WORDLE game.



The screenshot shows the Wordle game grid. The first row contains the letters 'R', 'I', 'O', 'T', and 'S'. The 'R' and 'T' tiles are yellow, while the 'I', 'O', and 'S' tiles are grey. Below the first row are four more rows of empty white squares, each containing five squares.

**Step - 3:** The respective colors we will fill in Wordle-solver webpage.



The screenshot shows the 'GUESS THE WORD' interface. The letters 'R', 'I', 'O', 'T', and 'S' are each in a colored square. The 'R' and 'T' tiles are orange, while the 'I', 'O', and 'S' tiles are grey. Below each letter is a dropdown menu with the text 'Orange' or 'Grey' and a downward arrow. At the bottom center is a light blue 'Submit' button.

Step - 4: After clicking on the ‘Submit’ button, the code will give us the next guess and this process continues until a word is guessed.

GUESS THE WORD

C

A

N

D

Y

Select

Select

Select

Select

Select

Submit

Check above guess

R

I

O

T

S

C

A

N

D

Y

C

A

N

D

Y

Grey

Orange

Grey

Grey

Grey

Submit

Check above guess

GUESS THE WORD

P

L

U

M

E

Select

Select

Select

Select

Select

Submit

Check above guess

R

I

O

T

S

C

A

N

D

Y

P

L

U

M

E

P

L

U

M

E

Grey

Green

Grey

Grey

Orange

Submit

Check above guess

GUESS THE WORD

A

L

T

E

R

Select

Select

Select

Select

Select

Submit

Check above guess

R

I

O

T

S

C

A

N

D

Y

P

L

U

M

E

A

L

T

E

R

A

L

T

E

R

Green

Green

Orange

Orange

Orange

Submit

Check above guess

GUESS THE WORD

A

L

E

R

T

Select

Select

Select

Select

Select

Submit

Check above guess

R

I

O

T

S

C

A

N

D

Y

P

L

U

M

E

A

L

T

E

R

A

L

E

R

T

A

L

E

R

T

Green

Green

Green

Green

Green

Submit

Gussed the correct word!!!

Run Again

# Inefficiencies in Wordle Guessing Due to High Similarity Words

In Wordle-solving algorithms, an issue arises when multiple remaining candidate words share a high degree of similarity.

For example, if the actual word is ***fight***, but our word list includes words like

bight <sub>12</sub>	fight <sub>12</sub>	might <sub>12</sub>	wight <sub>12</sub>	hight <sub>11</sub>	dight <sub>10</sub>	light <sub>10</sub>
night <sub>10</sub>	eight <sub>9</sub>	right <sub>9</sub>	sight <sub>9</sub>	tight <sub>9</sub>		

the algorithm may take multiple extra guesses to reach the correct answer. This is because each guess provides limited new information, primarily eliminating only one word at a time rather than significantly reducing the entire candidate pool.