My Approach:

- My statistical technique involves counting the maximum number of letters in a word as one. This ensures that if a letter appears several times, it is filled at each position. I cleaned the word, eliminated spaces, searched for comparable lettered words in the dictionary, and guessed the most often occurring letter that was not guessed in the current iteration.
- After analyzing the ratio of vowels to word length, I discovered that if 55% of a word is made up of vowels, it is improbable that there will be additional vowels (based on the distribution).

Input Processing:

- The word parameter is a string representing the current state of the word being guessed, with spaces between letters and underscores for unknown letters (e.g., "_ p p _ e").
- The code cleans the word by removing spaces and replacing underscores with a dot (.) to use in regular expressions. This results in clean_word, which can match any character at the positions of the dots.

Dictionary Filtering:

- len_word is the length of the cleaned word (excluding spaces).
- current_dictionary is a list of possible words from previous guesses.
- new_dictionary is initialized to store words from current_dictionary that match the pattern of clean_word.

Filtering Possible Words:

- The code iterates through each word in current_dictionary.
- If a word's length does not match len_word, it is skipped.
- If a word matches the pattern of clean_word using re.match, it is added to new_dictionary.
- After iterating, self.current_dictionary is updated with new_dictionary.

Character Frequency Analysis:

- A Counter object character_frequency is created to count the frequency of each character in the plausible words.
- The characters are sorted by frequency in descending order using most_common()

Guessing the Next Character:

- The code iterates through the sorted characters.
- It returns the first character that has not already been guessed (self.guessed_letters).

Fallback:

- If no characters are left to guess (all have been guessed already), it returns '!'

Note:

I initially reached an accuracy of more than 50%, but after executing the API numerous times, I discovered that the accuracy was falling.

And finally my account got deactivated while executing 1000 trials and it stopped at 572nd game