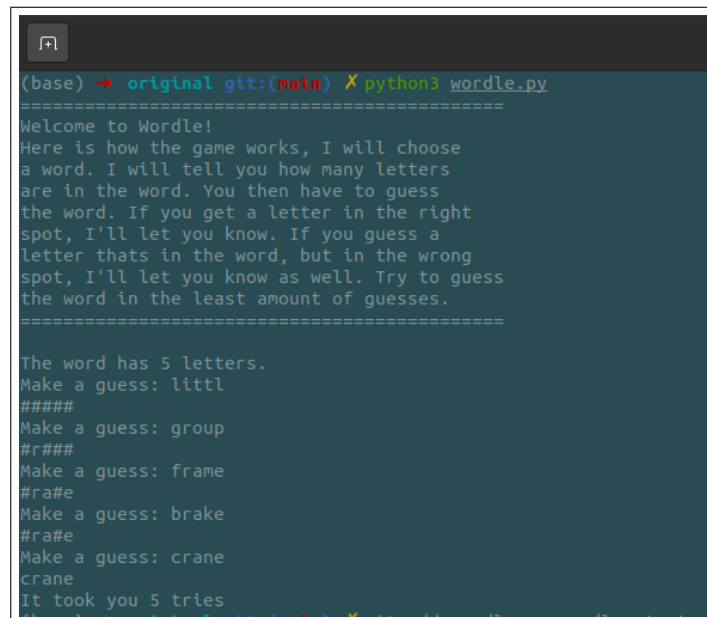


1 Wordle

For this project, you will be recreating the popular game *Wordle*! For those who haven't played, *Wordle* is a word game in which the player has a secret 5-letter word. After each guess, the user is given information regarding how each letter of their guess fits into the secret word (i.e. whether or not each letter is in the word and/or in the correct spot).



```
(base) -> original git:(main) X python3 wordle.py
=====
Welcome to Wordle!
Here is how the game works, I will choose
a word. I will tell you how many letters
are in the word. You then have to guess
the word. If you get a letter in the right
spot, I'll let you know. If you guess a
letter thats in the word, but in the wrong
spot, I'll let you know as well. Try to guess
the word in the least amount of guesses.
=====

The word has 5 letters.
Make a guess: littl
####
Make a guess: group
#r###
Make a guess: frame
#ra#e
Make a guess: brake
#ra#e
Make a guess: crane
crane
It took you 5 tries
```

Figure 1: Example for the secret word “CRANE”.

Of the code provided, the only file that you are allowed to modify is *wordle_starter.py*, which contains starter code for the project. To start the program, simply run *wordle_starter.py*. Your job will be to program the logic behind Wordle, which requires your implementation of three methods, outlined in the following subsections.

1.1 Part 1: Initialize the Game (10 pts.)

To select a secret word and to determine whether or not a guess is valid, *Wordle* uses a long list of “valid words.” The file *words.txt* (provided for you) contains over two thousand valid 5-letter words, one per line. Your job is to implement the *init_game()* method, which accepts the file path (of type *string*) as input, and outputs a randomly selected secret word. Implement the following inside of the *init_game()* method:

1. Randomly select the secret word
2. Return the secret word.

1.2 Part 2: The Game Loop (10 pts)

Now that we have a word selected, we must begin accepting user input in what is called the “Game Loop”. This is a loop that will continue until the game is over.

1. Keep track of how many times the user guesses

2. Check that the player's guess is exactly 5 letters long.
3. If the guess is 5 letters long, then use the `eval_guess` function
4. Return the number of guesses at the end of the function

1.3 Part 3: Evaluate a Guess (20 pts)

Now that you have verified that a guess is valid, we need to evaluate the guess and tell the user how they did. In *Wordle*, each letter in the guess is evaluated to be: *a*) correct letter in the correct position, *b*) correct letter in the wrong position, or *c*) incorrect letter. Your job is to implement the `eval_guess()` method, which accepts the secret word and the user's guess as input, and outputs a letter-by-letter evaluation of the guess. Implement the following inside of the `eval_guess()` method:

1. Loop through each letter of the guess and determine how it should be evaluated, based on the secret word.
2. Construct an output string that contains each correct letter in the correct position, a '*' for each correct letter in the incorrect position, and a '#' for each incorrect letter. For example, if the secret word is "CRANE" and the user guesses "LEARN", the evaluation should be the string "#*A**".
3. Return the evaluation string, as constructed in the previous step.

1.4 Code Submission

Congratulations – you just recreated the game *Wordle*! The only code that you must include in your submission is the `wordle_starter.py` file.

1.5 Questions (10 pts)

1.5.1 For Part 1, how did you ensure that you selected a random word? (2 pts)

1.5.2 For Part 2, how did you ensure the guess was a valid guess? What did you do if it wasn't valid? (3 pts)

1.5.3 For Part 3, what challenges did you face when handling this portion of the project? How did you check that you were correctly evaluating the guess? (5 pts)