1 Overview

A word game named WORDLE has gained its popularity recently. The game WORDLE is challenging, but simple: Once a day, players have six guesses to identify a new five-letter word (all players receive the same word on a given day). Each guess provides color-coded hints: a letter turns green if it is in the correct spot, yellow if it is part of the word but in a different spot, and gray if it is not in the word at all. An example to a single play is illustrated in Figure 1.

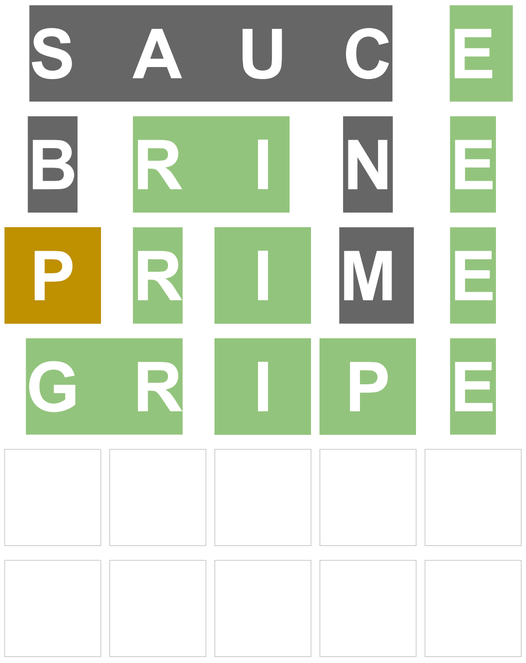

Figure 1: An illustration on how to play WORDLE game.

In this assignment, you are expected to develop a simplified version of this game. To do that follow the instructions below.

Instructions

- Create a Python file named `wordle.py`[2].
- Provide the 'word of the day' (WoD) as command-line argument. Note that the length of the word must not be less or greater than five. So, first check the length of WoD and display "**The length of WoD must be five!**" provided that the user/player violates this rule.
- Allow the user to provide **six words** in total from keyboard.

[2]

- After each try, your program must first check if the length of the provided word is less or greater than **five**. If so, it should print out "**The length of word must be five!**".
- Once WoD is guessed right by the user before (s)he runs out of all shots, the following message should be displayed, and no more input is expected from user:

  ```
  Congratulations! You guess right in kth shot!
  ```

  Note here that k is a counter for the attempt (i.e., $1 \leq k \leq 6$), and so the suffixes to represent the ordinal indicator (e.g., *-st, -nd, -rd, or -th*) must be consistent with the number k. (e.g., "**Congratulations! You guess right in [1st, 2nd, 3rd, 4th, ...] shot!**")
- If none of the above criteria is satisfied, it is time for your program to examine the word letter by letter. For example, the output should be as follows when WoD is given as **`BASIC`** and the user provides **`DASNB`** in an attempt:

  ```
  1. letter does not exist.
  2. letter exists and located in right position.
  3. letter exists and located in right position.
  4. letter does not exist.
  5. letter exists but located in wrong position.
  ```

- When user failed to find the **WoD** after he/she runs out of all tries, your program should print out "**You are failed!**".

Command for Running the Application

```
python wordle.py DoW
```
[3]

Demonstrations

Example 1

When your program is run with the **`python wordle.py GRIPE`** command, and user provided the following as a sequence[4]

1. LONDON
2. SAUCE
3. BRINE
4. PRIME
5. GRIPE

[3]it is a word of the day with length of five where all letters are written in uppercase.
[4]note user should provide one input at a time and wait for the feedback from the program before providing the next one, if any.

It should produce the following output. Note that the output of your program must match the one given here including the printing format shown below.

```
Try1 (LONDON): The length of word must be five!
Try2 (SAUCE):
1. letter does not exist.
2. letter does not exist.
3. letter does not exist.
4. letter does not exist.
5. letter exists and located in right position.
Try3 (BRINE):
1. letter does not exist.
2. letter exists and located in right position.
3. letter exists and located in right position.
4. letter does not exist.
5. letter exists and located in right position.
Try4 (PRIME):
1. letter exists but located in wrong position.
2. letter exists and located in right position.
3. letter exists and located in right position.
4. letter does not exist.
5. letter exists and located in right position.
Try5 (GRIPE): Congratulations! You guess right in 5th shot!
```

Example 2

When your program is run with **`python wordle.py IDEAL`** command, and user provided the following as a sequence:

1. NEVER
2. PAUSE
3. SAUCE
4. BRINE
5. PRIME
6. PRICE

It should produce the following output. Note that the output of your program has to match with the one given here including the printing format shown below.

```
Try1 (NEVER):
1. letter does not exist.
2. letter exists but located in wrong position.
3. letter does not exist.
4. letter exists but located in wrong position.
5. letter does not exist.
Try2 (PAUSE):
1. letter does not exist.
```

```
2. letter exists but located in wrong position.
3. letter does not exist.
4. letter does not exist.
5. letter exists but located in wrong position.
Try3 (SAUCE):
1. letter does not exist.
2. letter exists but located in wrong position.
3. letter does not exist.
4. letter does not exist.
5. letter exists but located in wrong position.
Try4 (BRINE):
1. letter does not exist.
2. letter does not exist.
3. letter exists but located in wrong position.
4. letter does not exist.
5. letter exists but located in wrong position.
Try5 (PRIME):
1. letter does not exist.
2. letter does not exist.
3. letter exists but located in wrong position.
4. letter does not exist.
5. letter exists but located in wrong position.
Try6 (PRICE):
1. letter does not exist.
2. letter does not exist.
3. letter exists but located in wrong position.
4. letter does not exist.
5. letter exists but located in wrong position.
You are failed!
```