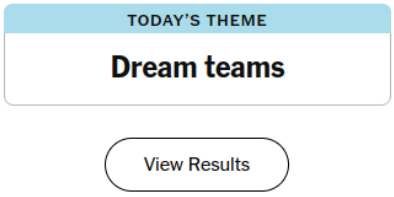
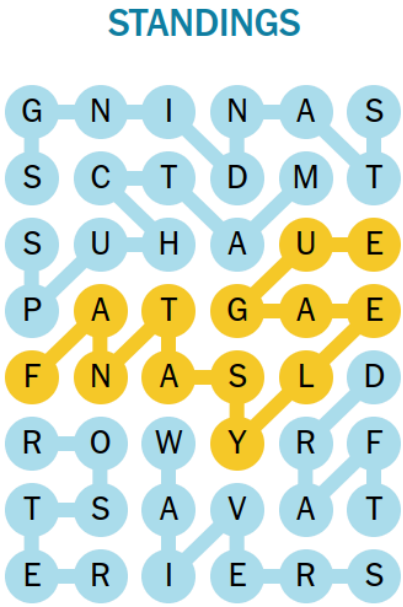
Strings: A New York Times Game Replica

The goal of this project is to create a replica of Strands, a New York Times puzzle game that consists of a group of words in an 8x6 grid. The words follow one theme with a spangram that traverses from one side of the board to the other side. The game ends when all the theme words in the puzzle have been found (win case) or when the player decides to reveal all the words (lose case).



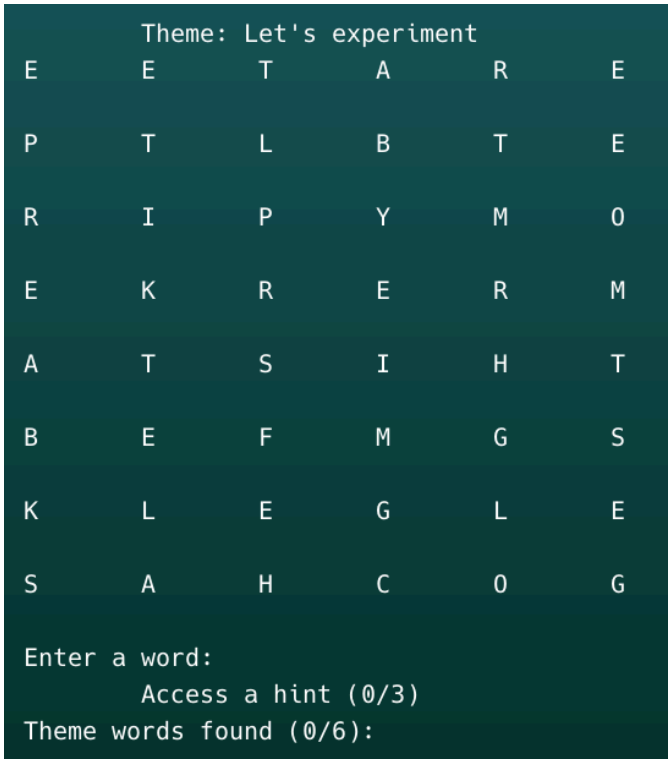


Implementation

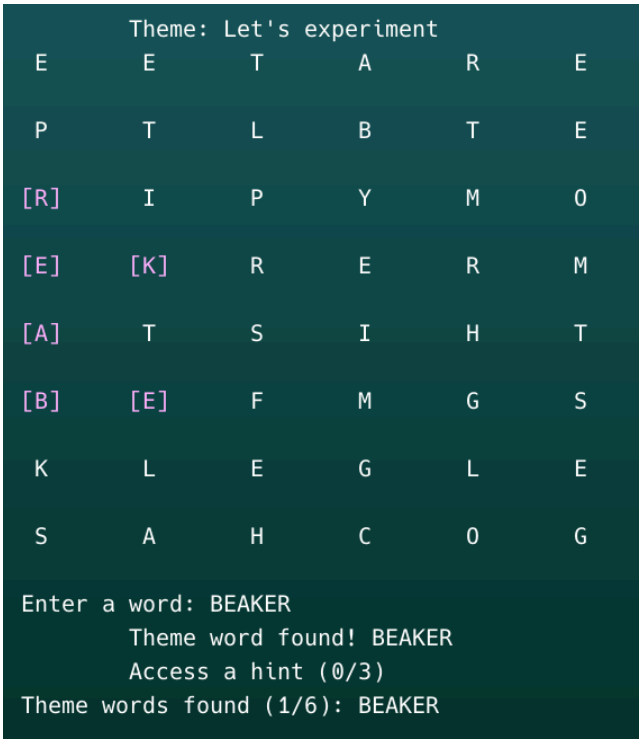
1. You will be creating three files containing data for one puzzle each. A file contains the following:
    a. The first line of the file is the theme of the puzzle
    b. The next eight lines constitute the puzzle matrix where each line has six characters each. This should form the 8x6 grid.
    c. The next line is a number which represents the number of theme words found in the puzzle. Let this number be n.
    d. The next eight lines contain the locations of the theme words. Set 0 as the location of the spangram. Location numbers are from 0 to n-1.
    e. The next n lines contain the theme words. Match this with the locations from part c. Using the test case as an example, 0: spangram, 1: beaker, 2: flask, …, 5: thermometer.
    f. The last n x 3 lines contain the hint words.

    See test_case1.txt for a sample of the file. You may get the puzzles from the Strands archive.

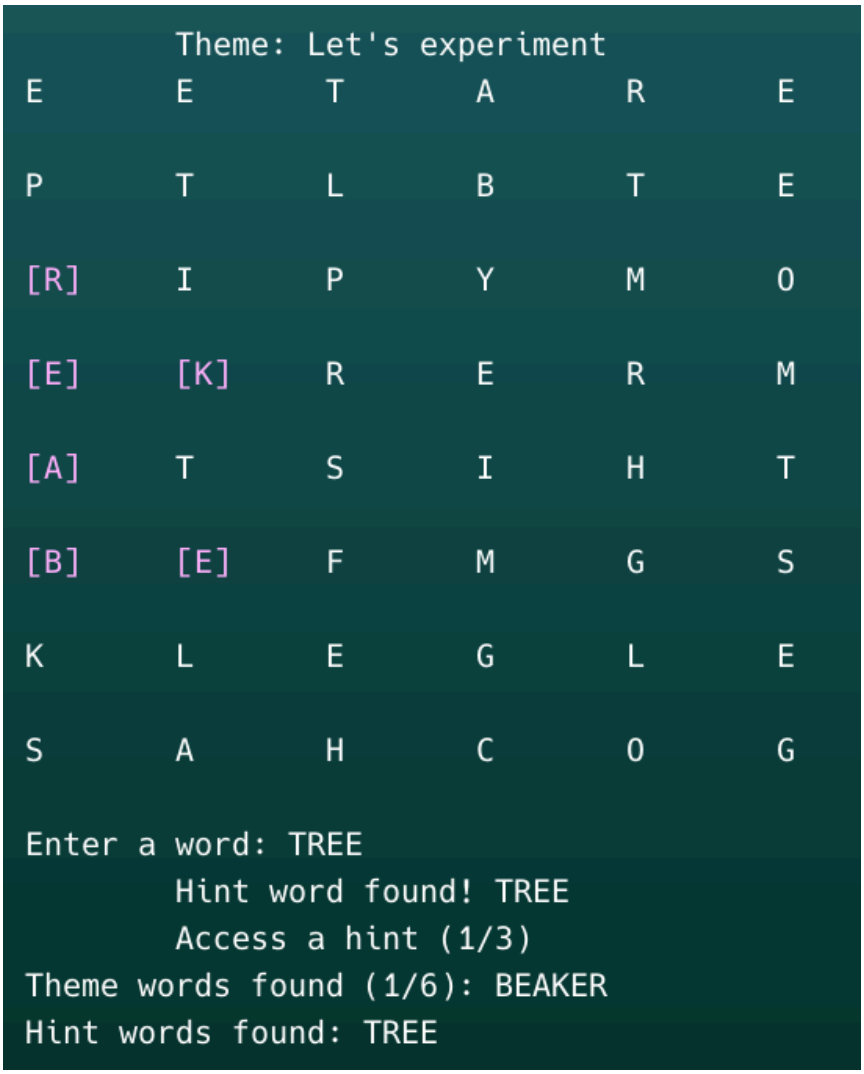
2. Upon running the program, the player can select any of the three puzzles. When a puzzle is selected, its corresponding file is loaded to the screen, similar to this:



3. The player tries to look for theme words. When a player wants to submit a theme word, they type it on the screen. When a theme word is found, the word is highlighted with a square bracket (i.e. [ ]). Theme words are not allowed to overlap.



4. Should the player need a hint, they can find non-theme words. Non-theme words are found in the input file.
   Input Limitations
   a. If the player finds a non-theme word that is not in the input file, output an error "Not in dictionary". Should you want to add more non-theme words (i.e. more than n x 3), you may do so.
   b. Players are not allowed to enter words that have less than four letters.
5. When a non-theme word is found, the word is added to the list of hint words. The non-theme words are not highlighted.



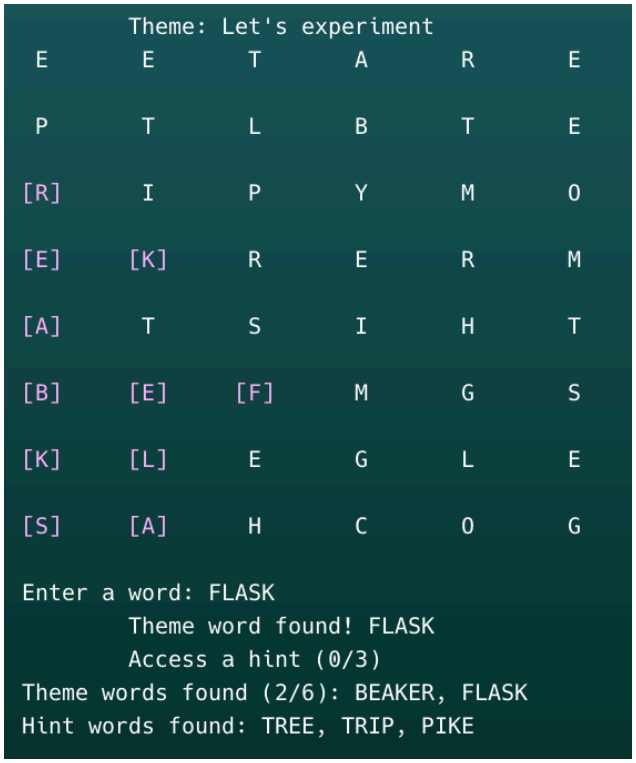
6. When three non-theme words are found, a hint becomes available to the player. To use a hint, type `hint` in lowercase.

```
        Theme: Let's experiment
E       E       T       A       R       E

P       T       L       B       T       E

[R]     I       P       Y       M       O

[E]     [K]     R       E       R       M

[A]     T       S       I       H       T

[B]     [E]     F       M       G       S

K       L       E       G       L       E

S       A       H       C       O       G

Enter a word: PIKE
        Hint word found! PIKE
        HINT AVAILABLE [Type "hint" and press enter]
Theme words found (1/6): BEAKER
Hint words found: TREE, TRIP, PIKE
```

7. Hints are highlighted with parentheses.

```
        Theme: Let's experiment
 E       E       T       A       R       E

 P       T       L       B       T       E

[R]      I       P       Y       M       O

[E]     [K]      R       E       R       M

[A]      T       S       I       H       T

[B]     [E]     (F)      M       G       S

(K)     (L)      E       G       L       E

(S)     (A)      H       C       O       G

Enter a word:
        Access a hint (0/3)
Theme words found (1/6): BEAKER
Hint words found: TREE, TRIP, PIKE
```

8. The hint highlight is removed when the player guesses the word correctly.



9. When the player guesses the spangram, the prompt “SPANGRAM!” is shown on the screen.

10. When the player guesses all the words, they can access the high scores (type `hs`) or exit the game (type `q`).

```
        Theme: Let's experiment
[E]     [E]     [T]     [A]     [R]     [E]

[P]     [T]     [L]     [B]     [T]     [E]

[R]     [I]     [P]     [Y]     [M]     [O]

[E]     [K]     [R]     [E]     [R]     [M]

[A]     [T]     [S]     [I]     [H]     [T]

[B]     [E]     [F]     [M]     [G]     [S]

[K]     [L]     [E]     [G]     [L]     [E]

[S]     [A]     [H]     [C]     [O]     [G]

Enter a word: THERMOMETER
        Access a hint (0/3)
Theme words found (6/6): CHEMISTRYLAB, BEAKER, FLASK, PIPETTE, FLASK,
GOGGLES, THERMOMETER
Hint words found: TREE, TRIP, PIKE
All theme words found. Congratulations!
        Access high scores [Type "hs" and press enter]
        Quit game [Type "q" and press enter]
```

11. The player can choose which scoreboard to check from the three puzzles.

```
Choose scoreboard to view:
[1] Let's experiment
[2] Dream teams
[3] Solar system
```

12. High scores are arranged by how many hints the player used in ascending order, then by time elapsed.

```
        HIGH SCORES: Let's experiment
NAME          |   HINTS USED  |   TIME (s)
Abi           |   0           |   114.21
Erika         |   0           |   234.13
Torena        |   1           |   116.04
JC            |   3           |   89.34
JM            |   4           |   120.15
```

13. When the player is unable to guess all the words, they may choose to reveal all the words. This is the lose case of the game. You are free to implement the reveal words/lose case as you see fit.

Supplementary Links

1. Strands: https://www.nytimes.com/games/strands
2. Puzzle archive of Strands: https://strandsarchivenyt.com
3. Test case: https://drive.google.com/file/d/1RNc_IF2CwkMw--9Me9sx6fPNz-btbnlL/view?usp=sharing

Grading

The project is 15% of your grade, with the following point system below.

A. Correct usage of programming concepts in Python (8 points total)
   a. Functions (2 points)
      Split up the code into readable and usable functions. If there are code blocks that look better as functions, they should be written as such.
   b. Lists or dictionaries (2 points)
      Prioritize correct usage. Consider if it is more appropriate to use one data structure over the other for a given type of data.
   c. Files (2 points)
      Implement a load mechanic for the puzzle data. Implement also a save and load mechanic for the high scores.
   d. Import (1 point)
      Create user-defined modules that you can use in your main program.
   e. Documentation (1 point)

B. Working game (7 points total)
   a. Milestone presentations
      Show your progress in the project by explaining your work. The final output should not differ from the milestone presentations.
      i. Milestone 1 (2 points): November 20 for sections G4L and Y5L; November 26 for G1L
      ii. Milestone 2 (2 points): November 27 for sections G4L and Y5L; December 3 for G1L
   b. Game mechanics (3 points)
      See implementation section.

C. Bonus (maximum of 2 points): Design and implement a better user interface on the project using the command-line interface only (i.e., no external GUI or game libraries like Tkinter or PyGame).

Notes

1. Do not use PyGame, third-party libraries, or any kind of data parser (even built-in ones like a JSON parser are NOT allowed).
2. If you use an IDE like PyCharm, make sure that your code still runs even when you execute it in the terminal without PyCharm. You will not get points for code that doesn't run.