Final Project Proposal - Strands

Yuki, Brooke and Mia
Language: Python
Link to Google Colab: Strands in Python! ipynb

For our final project, we will attempt to use Python to create a version of Strands in different languages. While we plan to begin with Spanish and French (due to the fact that they share an alphabet with English), we may possibly expand into other languages (and possibly those with different alphabets) depending on our progress. As the game is relatively new (released March 4th with an English version only) there is not a ton of directly related code, but we plan to utilize code from different word games as a starting point. We plan to build upon this code by 1) integrating multiple language options in one Strands-like interface, and 2) providing a 'hint' button which would provide hints in a different language for users who speak a different language than the one that the program is operating in, allowing the game to not only operate as a way to pass time (or procrastinate), but also a tool for language learning.

Broadly speaking, the work required to execute this game will fall into the following categories:

- Visual interface/updates
- Explain instructions
- Randomly generate the crossword with words from a specific theme using a dictionary API
- Allow the user to click on the crossword to find the words and check each attempt
- Filtering for invalid inputs (not a word, too short)
- Compare the user's input to the scrambled words in the crossword
    - Color the letters correctly found and make them unable to be used for other attempts
    - If a wrong word is guessed 3 times, give a hint that shows the general location of one word
- Check for correctness → continue if not yet have all the words, exit when the game is completed
- Replay function after the game is over to give the option to play again with a different theme

The skills we are hoping to gain are as follows :

- General greater fluency in Python
- Greater facility combining Python with different languages
- Using files in python and learning how to parse them to generate words randomly
- More confidence in using indexing to access values of variables

Timeline (detailed week by week, to be updated to include weekly summary of work)

Week 1 (May 22)	1. Draft proposal 2. Review GitHub repositories 3. Setting up functions to index and parse in different

	4. Pseudocode 5. Initial functions → start with the English version first (instructions, creating a word dictionary, visual gameboard)
Week 2 (May 29)	1. Continue to write code (function/work breakdown below)
Week 3 (May 6)	1. Final improvements to UI/UX 2. (potentially) adding a new language 3. Debugging and getting ready to present!!!

Breakdown of Main Functions
*initialize the game board
*call the function to start the game

1) Main game loop
    a) Display the game board
    b) Repeat until the game is over
2) Generate themes and words
    a) Create a list of themes and words that correspond to those themes (one word has to be the spanagram), possibly using Dictionary API
    b) Randomly select a theme
    c) Strategically mix up the theme words in the game board
    d) For different languages, use dictionaries to supply definitions or define certain categories of words
3) User input
    a) User tests out different words (clicking on the gameboard)
4) Update game board
    a) Highlight in yellow for the spanagram
    b) Highlight in blue for the words related to the theme
5) Valid words management
    a) Keep a dictionary of valid words
    b) Validate user input against this dictionary
6) Hints
    a) Hint button
    b) Give hints after three unsuccessful but valid word guesses
    c) Draw a dotted circle around words when a hint is used
7) Game over handling
    a) Replay button

Brooke	Yuki	Mia

Mixing up the words and getting them on the gameboard in proper format Checking user input against theme words Valid words management (valid vs invalid, unsuccessful/valid words)	Dictionary API - Implementing and creating text files with filter Main gameboard	Hints, Update Game Board, Game over handling

Helpful Links:
https://pypi.org/project/pycrossword/
https://github.com/sealhuang/pycrossword
https://github.com/frostming/wordle-tui/blob/main/wordle_app.py

https://docs.python.org/3/library/gettext.html

https://github.com/huzecong/FlowFree
https://github.com/mithracodes/flow-free/blob/main/README.md
- Flow free uses similar mechanisms to strands

DUE: MAY 12

Hours Log

Date	**Yuki**	**Brooke**	**Mia**
April 22, 2024	1.5 - finalizing idea, finding source code	1.5 - finalizing idea, finding source code	1.5 - finalizing idea, finding source code
April 24, 2024	1.5 - pseudocode	1 - pseudocode (organizing words on game board)	1 - pseudocode
April 25, 2024		1.5 - working on words_on_board function (need to fix to ensure it doesn't cut off words/organizes words correctly on the board)	
April 26, 2024		2 - words_on_board function (got it to loop through the list and add more than just one word to the board)	

April 29, 2024		3 - splitting up the words_board task into different functions/still working on word placement on board	
April 30, 2024	1 - worked on writing out all the pseudocode	4 - continued work on indexing and updating the board	4 - turning pseudocode into better code, learning how to use pygame functions in python
May 1, 2024		2 - creating a while loop to add words to the board	5 - building out game loop to create clickable letters, and updating functions for board updates, trying to get things to display
May 2, 2024	4 - finished code for selecting random words from text file and working on the general game loop	3- valid placement of words/working on why it stops mid word sometimes	4 - finally got mouse click to work with some debugging.
May 3, 2024	5 - looked into dictionary api, worked with getting words from it, worked a bit on the main loop and general game board creation	4 - new method: Hamiltonian paths to generate a way to display words across the board	2 - got words to display on the top of the board!
May 4, 2024	2 - working on dictionary api and game board	4 - got the letters to get placed on the board (but board must be square dimensions), working on getting it to also function in an 8x6 board	1 - hint button
May 5, 2024	5 - working on visuals 3.5 - incorporating click function into main loop and finalizing the crossword board	1 -	4 - making sure that only cells adjacent to previous, clearing board if someone clicks not on a letter
May 6, 2024	2.5 - adding click function v2 to main loop and fixing event	4 - Combined the click & paths functions, make the words blue	4 - combined code for design and absolute value click function,

	issues	once found, reset to try guessing a new word, generating paths for 8x6 board (file)	added word display with graphics
May 7, 2024	2 - fixing calling for dictionary api	2- trying again to get the 8x6 board to generate a file of the possible pathways	2 - dictionary API, starting hint button
May 8, 2024	5 - finished fixing dictionary api stuff, worked on combining the api and the random list creation and worked with main loop	3 - valid word management (trying to get valid words to go towards a hint but not make it count towards the word search → import eng dictionary and check it against test_list)	
May 9, 2024	1.5 - fixing main loop, debugging theme name error 2 - starting working on valid word text	5 - combining code, getting the words to place on the finalized visuals	8- working on hint button and pulling code from the API into the main loop while maintaining other functionality
May 10, 2024	5 - adding word list, trying to fix click motion (and failing), fixing api calling words	4 - getting the words to place on the board after it didn't save, importing a different library in valid words function, combining all code together	3 - trying to work through valid user logic and track indices and values of cells in correct words in order to 2- trying to figure out how to make sure the board is still clickable with new parametrics 3 - changed method for processing words to a time based one to detect a double click. Trying to make sure that this is registered
		2 - trying to get the click to work	3 hours on trying to get colors to show and reformat draw_board

**********INITIAL IDEA (don't read)

Wordle

- https://realpython.com/python-wordle-clone/
- https://www.freecodecamp.org/news/building-a-wordle-game/
- https://github.com/frostming/wordle-tui
- https://github.com/meicholtz/wordle/blob/main/wordle.py
- https://www.youtube.com/watch?v=NCgN4qtbh2Q
- https://codereview.stackexchange.com/questions/273894/wordle-puzzle-game
- https://medium.com/codex/building-a-wordle-solver-with-python-77e3c2388d63
- Wordle in Spanish
    - https://github.com/pablolop12/Python-Wordle-SPANISH/blob/main/Python%20Wordle/Wordle.py
    - https://github.com/Matyrela/PythonWordle/blob/main/main.py

    - List of Spanish words: https://drive.google.com/drive/folders/1Z5ONbcOvaynbehA0dbN3SrkEpkZ3F9Iw
- Shabdle: Wordle in Hindi:
    - https://github.com/kach/shabdle

Spelling Bee

- 6 attempts to guess a five letter hidden word
    - Different language

Initial functions (we will all work on this):

- Making language textfiles
    - Creating a dictionary for spanish and french with corresponding english translation

Breakdown of work:

- Main game loop
- Generate word randomly
- Input handling
- Word comparison
- Tracking (current state of guessed word and number of turns remaining)
- Game over handling
- Valid words management (can only guess real words)
- Visuals

Visuals:

- First page → 3 buttons to select which language you want
- 6 rows of blank boxes for the trials
- Boxes after a trial: correct word and position → green; correct word, incorrect position → yellow; incorrect word, incorrect position → gray
- For other languages, after the 3rd/4th trial give a hint box with the english translation of the hidden word
- After all trials or correct guess give message box at the end

What the program has to do:

- Explain instructions
- Randomly generate the word for the day
- Take the user's input and track their guesses
- Possible answers (can't input something that is not a word)
- Compare the user's input to the word for the day (update the gameboard with the colours based on the letters that did/didn't work)
    - Green = correct letter, correct position
    - Yellow = correct letter, incorrect position
    - Grey = incorrect letter, incorrect position
- Check for correctness → continue if not yet have the word, exit if you guess correctly or have used all 6 attempts

- Give hints if the user clicks on the hint button
- Replay function after the game is over to give the option to play again

*also create the visual gameboard
*could also make a hint button (i.e. can give the frequency of letters in the word if there is a repeat)

What we need to check for:
- Which letters were guessed correctly and in what positions
- Which letters we can still guess
- Which letters are not in the correct spot

Use indexing to check if the correct letter is in the correct position - example:
correct_word = “tough”
guess_user = “rough”

```
for i in range(len, correct_word):
        if correct_word[i] == guess_user[i]:
                print “correct”
        else:
                print “incorrect”
```

^this gets more complicated for letters that are correct but not in the correct spot

Reverse outline of code:
Defined before functions:
1. Correct_words: stores correct words
2. Theme words: stores theme words
3. May need a list to store indices of the cells of these words
4.