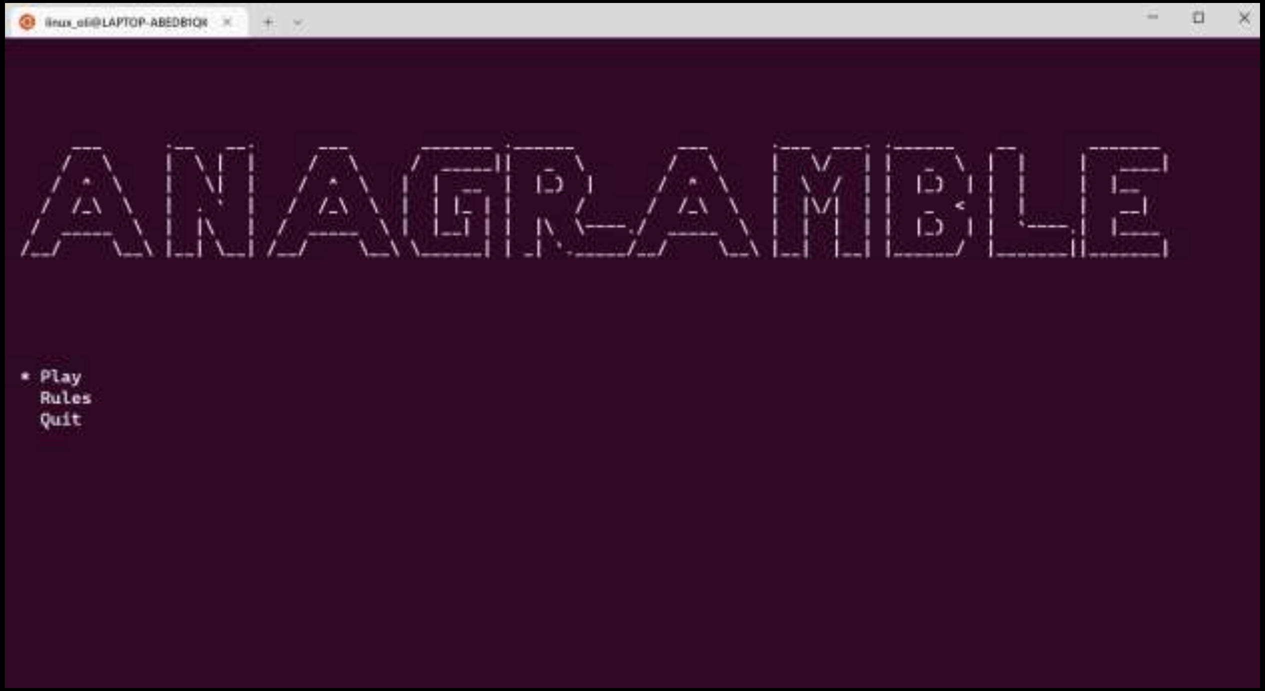
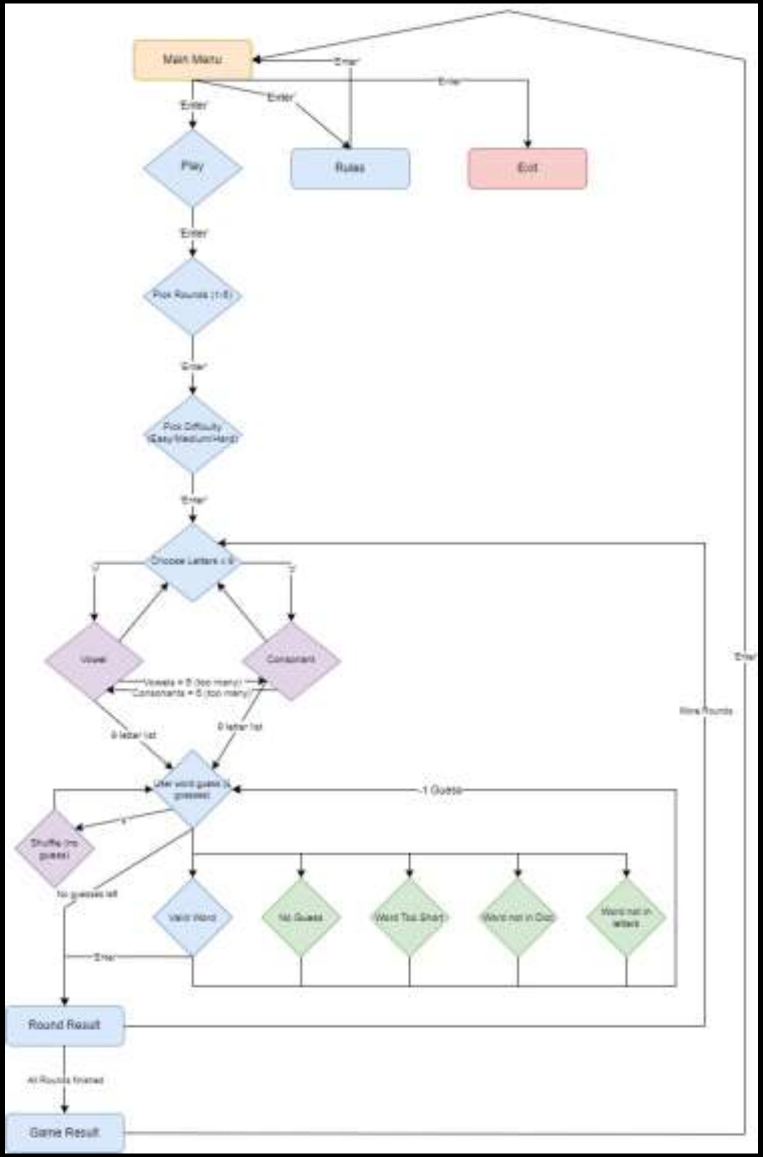
T1 A3 Slide Deck

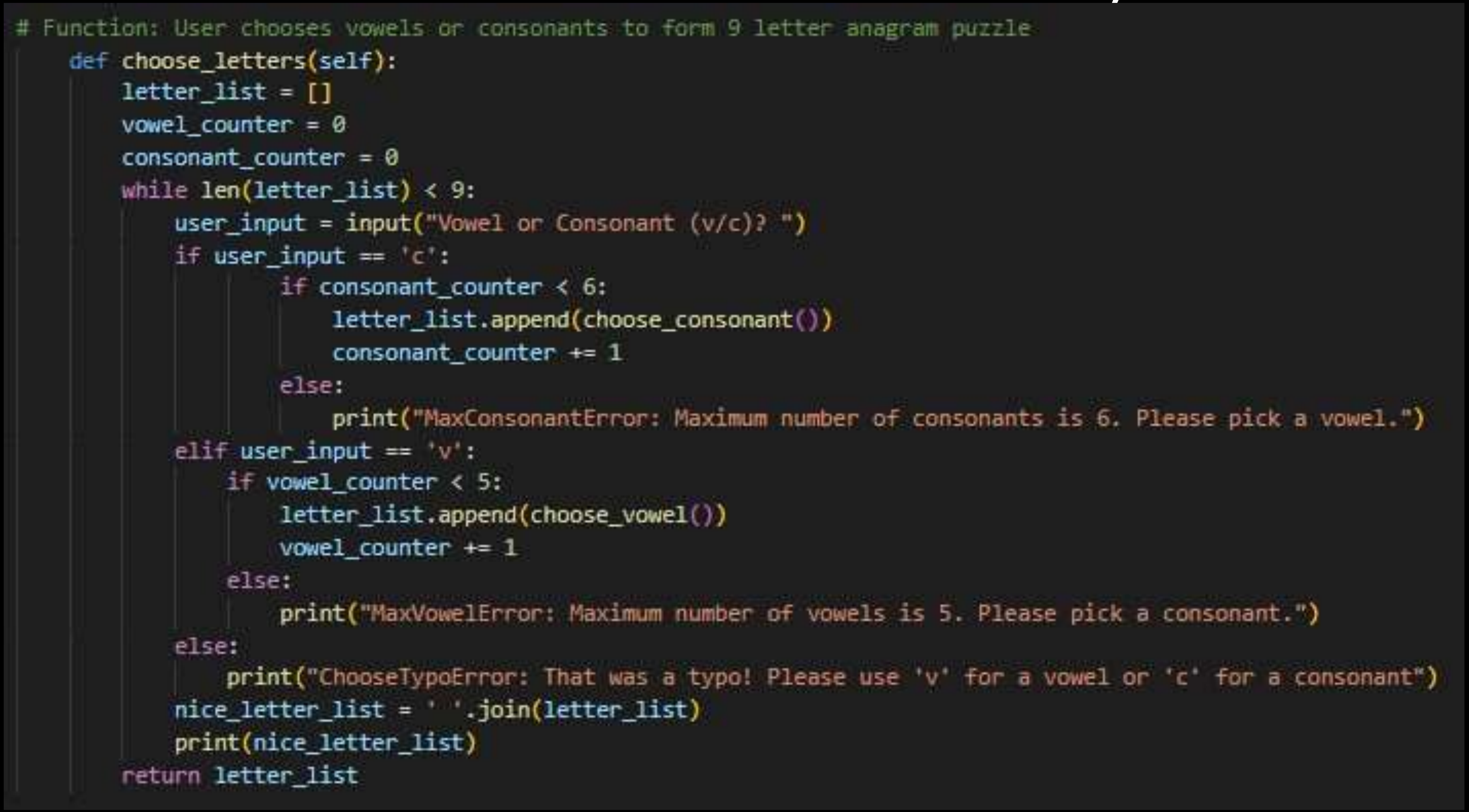


Oliver Wong

Flowchart!

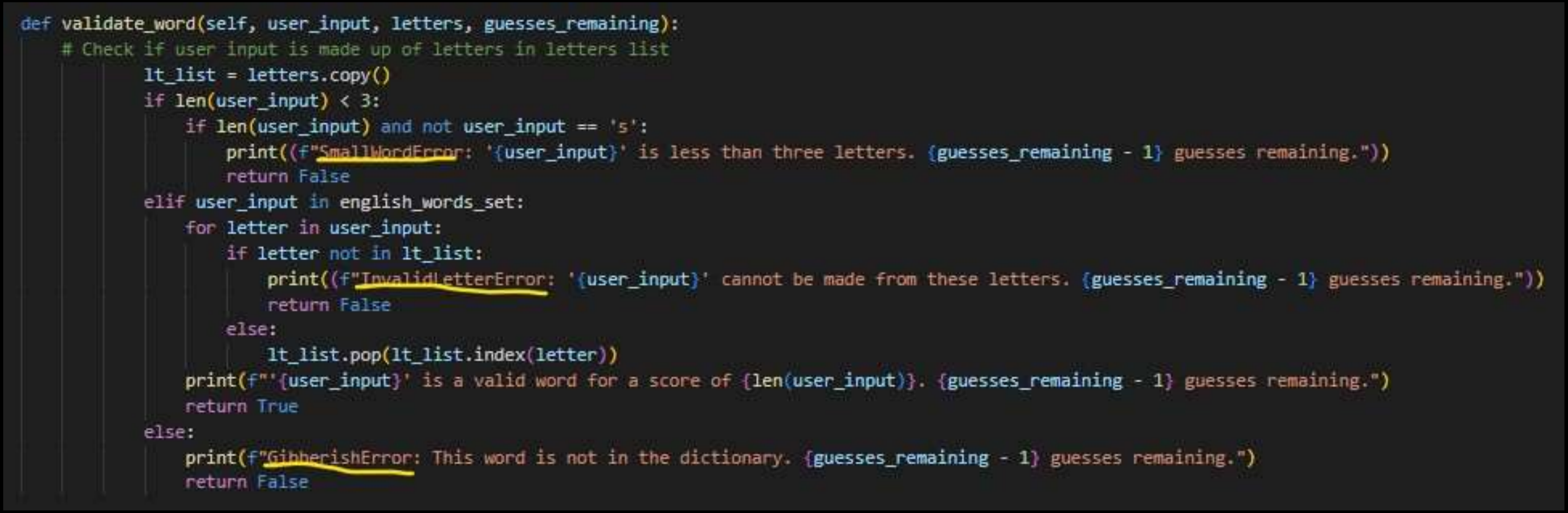

Main Menu
'Enter'
'Enter'
'Enter'
'Enter'
Play
Rules
Exit
'Enter'
Pick Rounds (1-5)
'Enter'
Pick Difficulty (Easy/Medium/Hard)
'Enter'
Choose Letters x 9
Vowel
Consonant
Vowels = 5 (too many)
Consonants = 6 (too many)
9 letter list
9 letter list
User word guess (5 guesses)
-1 Guess
Shuffle (no guess)
No guesses left
Valid Word
No Guess
Word Too Short
Word not in Dict
Word not in letters
'Enter'
Round Result
More Rounds
'Enter'
All Rounds finished
Game Result

Snippet Walkthrough (How the user chooses letters!)



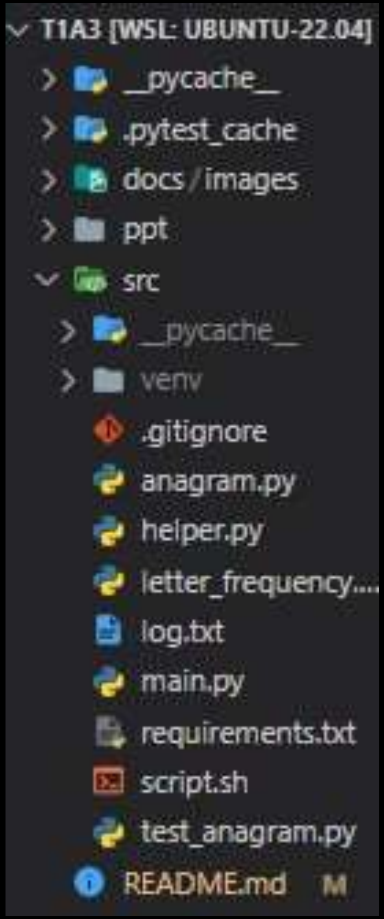
```python
# Function: User chooses vowels or consonants to form 9 letter anagram puzzle
    def choose_letters(self):
        letter_list = []
        vowel_counter = 0
        consonant_counter = 0
        while len(letter_list) < 9:
            user_input = input("Vowel or Consonant (v/c)? ")
            if user_input == 'c':
                    if consonant_counter < 6:
                        letter_list.append(choose_consonant())
                        consonant_counter += 1
                    else:
                        print("MaxConsonantError: Maximum number of consonants is 6. Please pick a vowel.")
            elif user_input == 'v':
                if vowel_counter < 5:
                    letter_list.append(choose_vowel())
                    vowel_counter += 1
                else:
                    print("MaxVowelError: Maximum number of vowels is 5. Please pick a consonant.")
            else:
                print("ChooseTypoError: That was a typo! Please use 'v' for a vowel or 'c' for a consonant")
            nice_letter_list = ' '.join(letter_list)
            print(nice_letter_list)
        return letter_list
```

Errors



```python
def validate_word(self, user_input, letters, guesses_remaining):
    # Check if user input is made up of letters in letters list
            lt_list = letters.copy()
            if len(user_input) < 3:
                if len(user_input) and not user_input == 's':
                    print((f"SmallWordError: '{user_input}' is less than three letters. {guesses_remaining - 1} guesses remaining."))
                    return False
            elif user_input in english_words_set:
                for letter in user_input:
                    if letter not in lt_list:
                        print((f"InvalidLetterError: '{user_input}' cannot be made from these letters. {guesses_remaining - 1} guesses remaining."))
                        return False
                    else:
                        lt_list.pop(lt_list.index(letter))
                print(f"'{user_input}' is a valid word for a score of {len(user_input)}. {guesses_remaining - 1} guesses remaining.")
                return True
            else:
                print(f"GibberishError: This word is not in the dictionary. {guesses_remaining - 1} guesses remaining.")
                return False
```

File Structure



```
T1A3 [WSL: UBUNTU-22.04]
  > __pycache__
  > .pytest_cache
  > docs / images
  > ppt
  v src
    > __pycache__
    > venv
      .gitignore
      anagram.py
      helper.py
      letter_frequency....
      log.txt
      main.py
      requirements.txt
      script.sh
      test_anagram.py
    README.md    M
```

- The File Structure is that of the assignment’s deliverables criteria.
- The paths to the markdown file are relative.
- The python files “main”, “anagram”, “helper”, “letter_frequency” and “test_anagram” are all in the src doc.
- The ‘log.txt’ file is the git log until the final commit
- The ‘help.md’ file is the installation instructions from the repo.
- The ‘script.sh’ file is the bash script to install the dependencies required. These are formally listed in the ‘requirements.txt’ file.

Review:

Challenges	Ethical Issues	Favourite Parts
- Errors could be handled better, but happy it is functional. - Finding an appropriate data set for English words was a compromise. - Testing (as a result of bad error handling) was more difficult and more ineffective. - Believing that I could actually do it was as much of a battle as the algorithms.	- Uncertainty about importing packages and the copyrights to them. I had to make sure but my understanding is that they are largely open-source and free for use. - The fundamental idea of the game could be interpreted as a copy/derivative for other word games. It would be interesting to unpack the validity of this assertion to see what is okay.	- I really enjoyed making the difficulty algorithm. I would like to see how more complicated games write elevated difficulties. - The constraints of the terminal/no GUI made the problem solving more math based. I got a thrill out of solving the issues on my own.

Questions?