TERMINAL APP

THI NGUYEN 2022

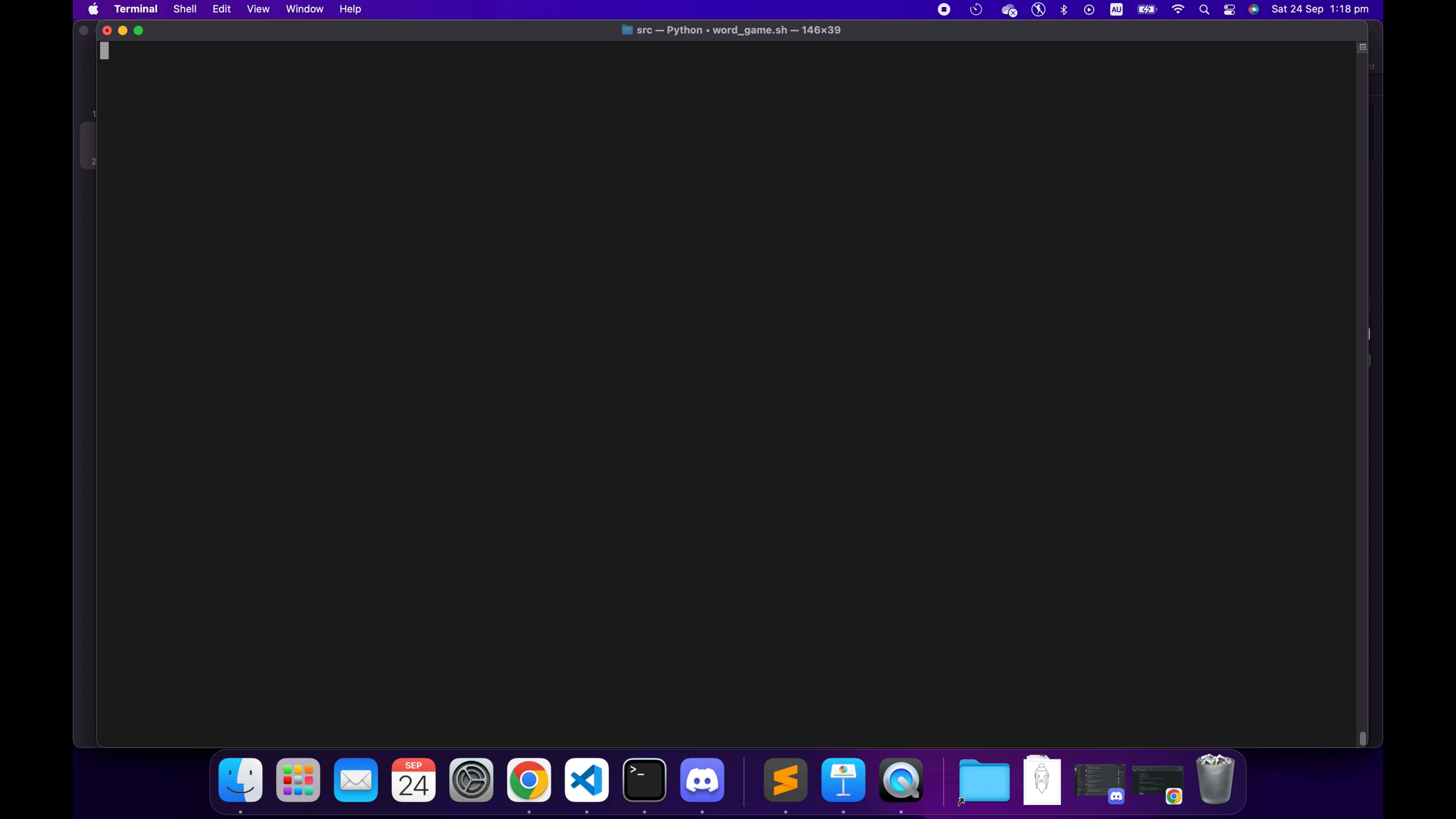
App Overview

1. Word game where a human player plays against a computer player.
2. Form a word from a provided ‘rack’ which has randomised letters
3. Whoever has the highest score after 7 rounds wins

Features:

1. Choose an opponent
2. Play a word
3. Get Help
4. Skip a turn
5. Quit any time

Project walk-through video


Terminal Shell Edit View Window Help
Sat 24 Sep 1:18 pm
src — Python ◂ word_game.sh — 146×39
SEP
24

src — Python ◂ word_game.sh — 146×39

```
------------------------------------------------------------
Round  1
Your point is: 0. The Word Master's point is: 0. Let's begin.


Your rack is: ['N', 'S', 'W', 'G', 'E', 'M', 'W']
The Word Master's rack is: ['R', 'C', 'R', 'U', 'E', 'O', 'N']

It's your turn. Play an English word.
***Type %Help to get help at any time***

>>>
```

src — Python ◂ word_game.sh — 146×39

```
------------------------------------------------------------
Round  1
Your point is: 0. The Word Master's point is: 0. Let's begin.


Your rack is: ['R', 'E', 'N', 'U', 'U', 'I', 'T']
The Word Master's rack is: ['Z', 'I', 'T', 'O', 'I', 'T', 'N']

It's your turn. Play an English word.
***Type %Help to get help at any time***

>>>
```

	Word	Data	Character	Game	Exceptions
Constants	english_words	letter_values letter_collection Greetings	N/A	N/A	N/A
Subclasses	N/A	N/A	Computer	N/A	InvalidInput, HelpRequired, SkipTurn, Quit
Attributes	word	N/A	name, word_length, rack, points	N/A	N/A
Methods	.verify()	N/A	.think(), .play(), .res ponse(),	.get_input(), announce(), calculate_points(), deal_letters(), play()	N/A

Game logic

Verify human player's word

Verify human player's word

STEP 1

* Checks if all letters in the word is valid
* If True, move to Step 2
* If False, loop back to input prompt

STEP 2

* Checks if the word is English
* If True, return the word
* If False, loop back to input prompt

Verify human player's word

STEP 1

```
        false_letters_invalid = set()
        false_letters_too_many = set()
        rack_letter_count = {self.word[i] : rack.count(self.word[i]) for i in range(len(self.
        word))}
        for letter in self.word:
            if letter not in rack:
                false_letters_invalid.add(letter)
            elif rack_letter_count[letter] > 0:
                rack_letter_count[letter] -= 1
            else:
                false_letters_too_many.add(letter)
```

Verify human player's word

STEP 2

```
    if not false_letters_too_many and not false_letters_invalid:
        if self.word.upper() in english_words:
            print(f'{self.word} is an excellent choice.')
            return self.word
        else:
            print(f'{self.word} isn\'t a valid English word.')
            return False
    else:
        if false_letters_invalid:
            print(f'{false_letters_invalid} {"is" if len(false_letters_invalid) == 1
            else "are"} not in your rack!')
        elif false_letters_too_many:
            print(f'{false_letters_too_many} {"is" if len(false_letters_invalid) == 1
            else "are"} used too many times!')
        return False
```

Process Review

OOP starts to click