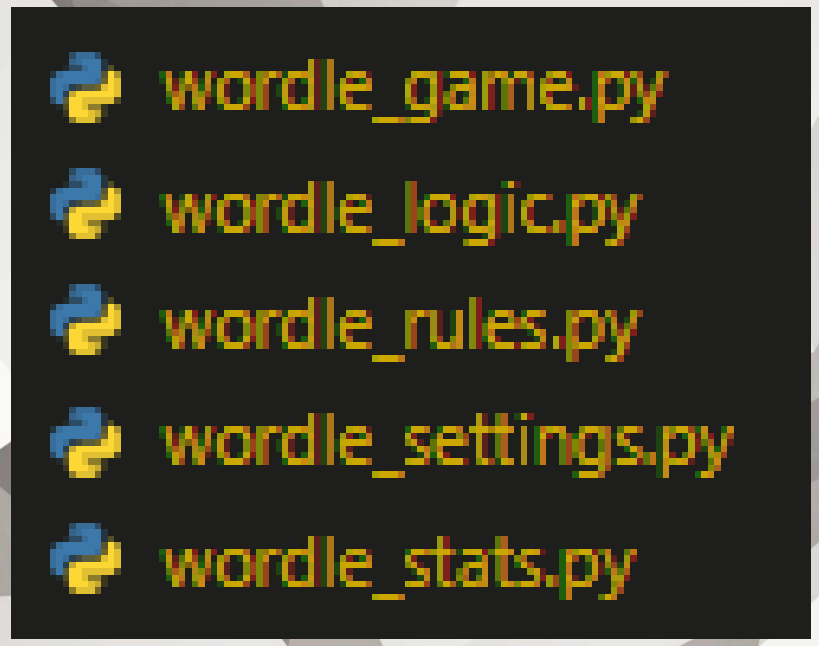
Welcome To PYWORDLE

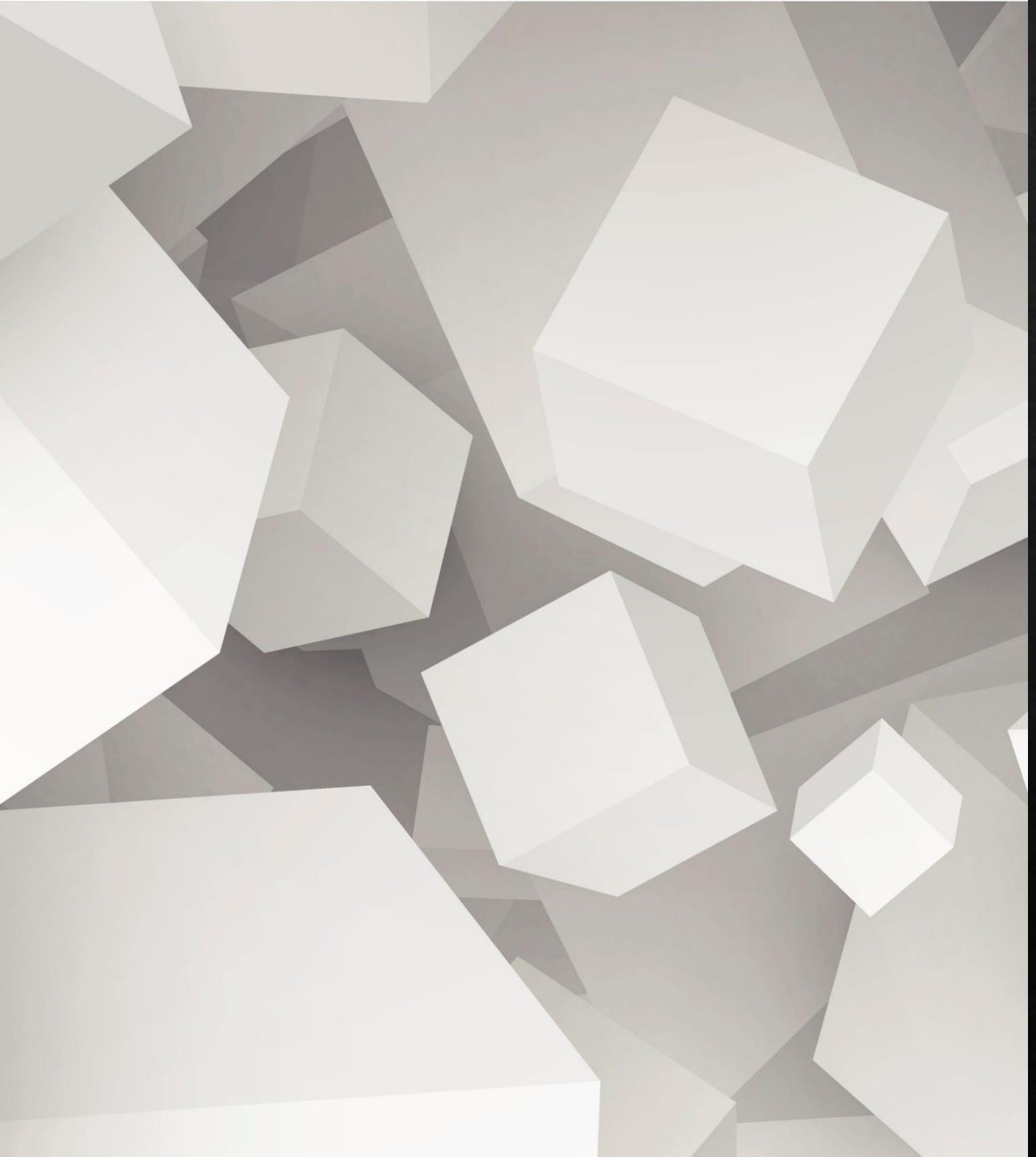
_ _ _ _ _

A Python Terminal Application
by Callum Rowston



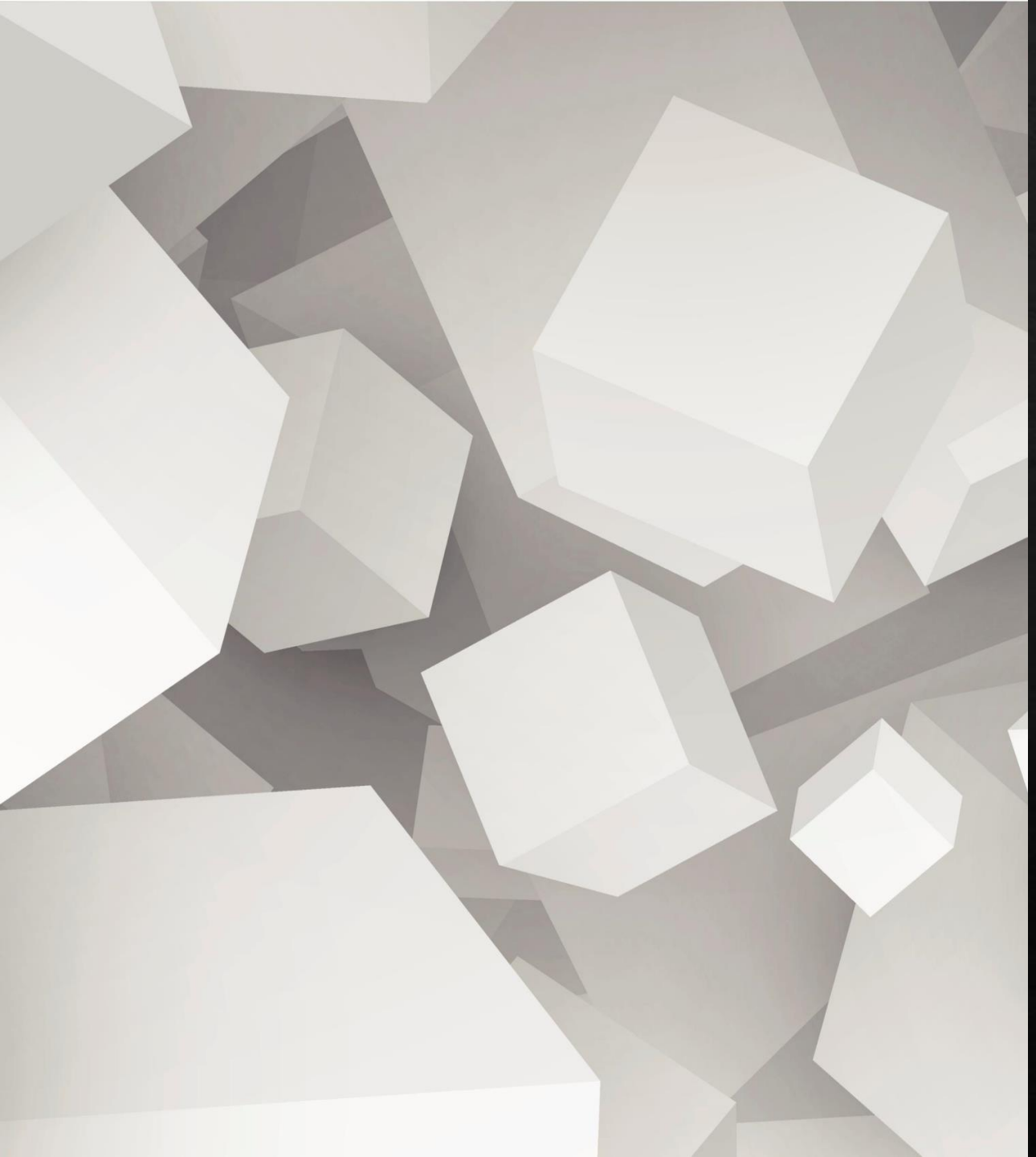
Walkthrough

- Main Menu
  - Play, Rules, Settings, Stats and Quit
- Modules
  - Modularization reflects the main menu layout
  - Separate module for game logic



Walkthrough

- Rules
  - Simple page explaining how to play
- Stats
  - Simple page displaying the user's stats
- Game Settings
  - Deeper menu allowing the user to change fundamental gameplay settings



Walkthrough

- Main Gameplay Loop
  - Works like regular Wordle
  - User guess is coloured depending on its match to the secret word
  - Game ends when the user runs out of guesses or guesses the secret word
  - Menu prompts to play again or quit

```
self.secret_word_list = []
self.secret_word = ""
self.current_guess = ""
self.guess_results = []
self.guess_count = 0
self.word_length = word_length
self.max_guesses = max_guesses
self.set_secret_word()
```

```
if len(user_guess) != self.word_length:
    raise WordLengthError(f"  Your guess must be {self.word_length} letters long. Guess again.")
if user_guess not in self.secret_word_list:
    raise NotRealWordError(f"  {user_guess} is not a valid word. Guess again.")

self.current_guess = user_guess
self.guess_count += 1
```

Code

- Main Gameplay Loop
  - WordleLogic class
  - Pick secret word
  - Accept a valid user guess
  - Compare guess and secret word
  - Colours the user guess depending on how it matches the secret word
  - Prints the result and prompts for another guess if the user has guesses left and hasn't correctly guessed

```
save_secret_word = self.secret_word
guess_result = ["-"] * self.word_length
for index, (guess_char, target_char) in enumerate(zip(self.current_guess, self.secret_word)):
    if guess_char == target_char:
        guess_result[index] = guess_char + "green"
        self.secret_word = self.secret_word.replace(guess_char, "-")

for index, (guess_char, target_char) in enumerate(zip(self.current_guess, self.secret_word)):
    if guess_char in self.secret_word and guess_result[index] == "-":
        guess_result[index] = guess_char + "yellow"
        self.secret_word = self.secret_word.replace(guess_char, "-")

for index, letter in enumerate(guess_result):
    if letter == "-":
        guess_result[index] = self.current_guess[index]

self.secret_word = save_secret_word
return guess_result
```

Code

- Main Gameplay Loop
  - WordleLogic class
  - Pick secret word
  - Accept a valid user guess
  - Compare guess and secret word
  - Colour the user guess depending on how it matches the secret word
  - Prints the result and prompts for another guess if the user has guesses left and hasn't correctly guessed

Code: How WordleLogic is used

```
wordle = WordleLogic(WORD_LENGTH_SETTING, MAX_GUESS_SETTING)
print()
print(*(("  ") + "  _" * wordle.word_length for _ in range(wordle.max_guesses)), sep='\n')
while wordle.play_wordle:
    try:
        user_guess = input(f"\n\n  Enter a {WORD_LENGTH_SETTING} letter word:\n  ").upper()
        wordle.validate_user_guess(user_guess)
    except (WordLengthError, NotRealWordError) as err:
        print(err)
    else:
        clear_screen()
        wordle.display_colored_guess(wordle.compare_user_guess())
        if not wordle.play_wordle and wordle.is_default:
            wordle.add_game_stats()
        if wordle.user_wins:
            if wordle.guess_count == 1:
                print(f"\n  WOW! You guessed it in 1 attempt!\n")
            else:
                print(f"\n  You won in {wordle.guess_count} guesses!\n")
            end_menu()
        if wordle.user_loses:
            print("  You have used all your guesses. Game over!")
            print(f"  The correct word was {wordle.secret_word}\n")
            end_menu()
```

Thanks for listening