Clean Code Development (5 points)

1. Standard Naming Convention

```
self.failed_attempts = 0
self.total_attempts = 0
self.game_started = False
self.username = ""
self.difficulty = ""
self.game_progress = ""
self.secret_word = ""
self.word_category = ""
```

Used meaningful names for variables in the functions and classes and seperate words by underscores to improve readability.

2. Functions

```
def find_indexes(self, letter):
    return [i for i, char in enumerate(self.secret_word) if letter == char]

def is_invalid_letter(self, input_):
    return input_.isdigit() or (self.regex.search(input_) is not None) or (input_.isalpha() and len(input_) > 1)
```

Defined functions that is relatively small and do only one thing in a better way. Used descriptive names for the functions and passed few arguments.

3. Comments

```
# Validate the user input
if self.is_invalid_letter(user_input):
    print('Please enter a letter')
    continue
# Check if the letter is not already guessed
if user_input in self.game_progress:
    print('You already have guessed that letter')
    continue
```

Used Comments to explain the functionality of code where necessary. Avoid misleading and noise comments.

4. WhiteSpacing and Indentation

```
def print_game_status(self):
    print('\n')
    print('\n'.join(HANGMAN[:self.failed_attempts]))
    print('\n')
    print(' '.join(self.game_progress))

def update_progress(self, letter, indexes):
    for index in indexes:
        self.game_progress[index] = letter
```

Followed PEP8 standard. Two line spaces between classes and one line space between methods. Used four spaces per identation level.

5. Tests

```
def test_play_right_guess(self):

    game.secret_word = "apple"
    right_guess = 0

    for ch in "apple":
        letter_found = game.find_indexes(ch)
        if letter_found:
            right_guess += 1
    self.assertEqual(right_guess, len(game.secret_word))

def test_play_wrong_guess(self):

    game.secret_word = "apple"
    right_guess = 0

    for ch in "orange":
        letter_found = game.find_indexes(ch)
        if letter_found:
            right_guess += 1
    self.assertNotEqual(right_guess, len(game.secret_word))
```

Defined tests that are fast and independent. Used only one assert per test. These tests are easily readable and easy to run.