The game is implemented using a class (Hangman), which helps organize the code, making it reusable and easy to manage. Variables like self.word, self.word\_display, and self.attempts are encapsulated within the class, preventing unnecessary global variables. The game should ideally use random.choice(self.words), ensuring a new word each time the game starts. Using self.guess\_letters = set() avoids duplicate entries and speeds up lookup operations compared to lists. self.word\_display = ['\_'] \* len(self.word) makes replacing letters easy when the user makes a correct guess. Storing hangman\_stages outside \_\_init\_\_ avoids redundant copies of the list in each instance. The while loop ensures the game runs until attempts are exhausted or the word is guessed. The program checks if a letter was already guessed to prevent redundant attempts. A loop is better suited for this type of sequential game, as recursion could cause unnecessary stack usage. While possible, OOP better maintains the game state, making the code modular and extendable.

