**Artificial Intelligent (Lab)**

**Task # 01**

****

**Submitted To: Sir Rasikh Ali**

**Submitted By: Shumaila Maryam**

**Roll no: SU92-BSDSM-F24-062**

**Submitted On: 8-August-2025**

**Section: BSDS-3A**

**Department of Software Engineering, Superior University, Lahore**

**Hangman Game**

The Hangman game is a word guessing game where the player tries to figure out a hidden word by guessing one letter at a time.

**How and why my code work?**

1. import random

The program begins by choosing a random category, such as Animals or Fruits. From that category, it selects a random word for the player to guess. This random selection ensures the game is different every time, keeping it fun and challenging.

1. HANGMAN\_PICS = [''']

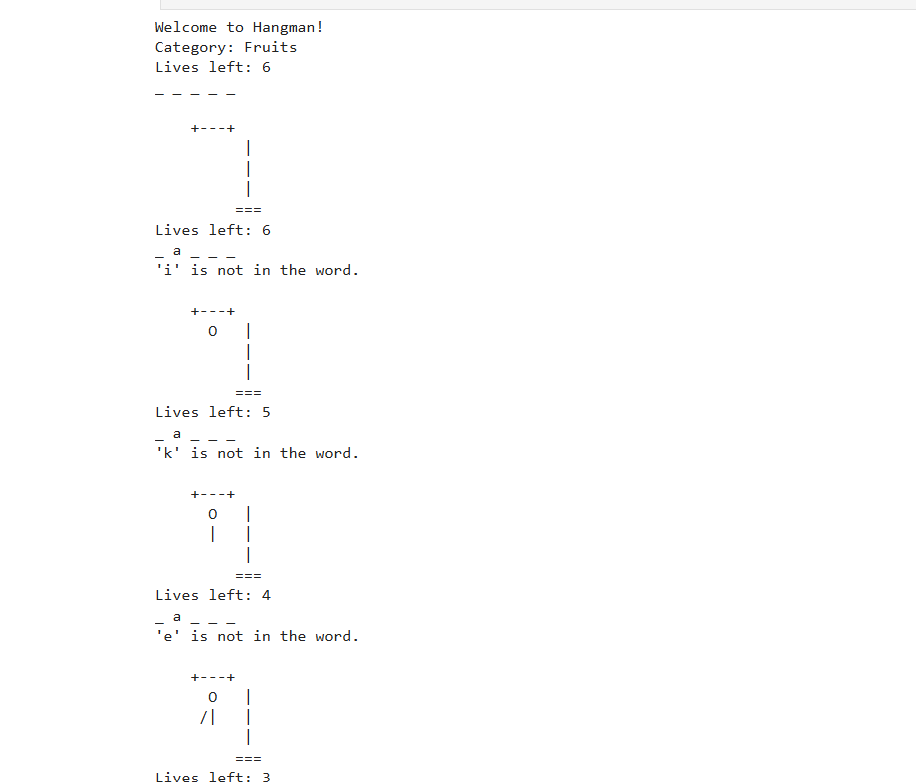
This is a list (an ordered collection) named HANGMAN\_PICS. Each item in this list is a multi-line string that visually represents a stage of the Hangman drawing. The drawings are ordered from the least complete (when the player has many lives) to the most complete (when the player has zero lives).

1. At the start, the game prepares blank spaces for the chosen word, showing one blank for each letter. This helps the player see how many letters they need to guess, giving a visual clue about the word’s length. The player also begins with a set number of lives, I set six, which represents how many incorrect guesses they can make before losing.
2. During the game, the player guesses one letter at a time. The program checks whether the letter has been guessed before. If it has, the player is informed, preventing repeated guesses and making the game fair. If the letter is in the word, the program reveals all occurrences of that letter in the word. If the letter is not in the word, the program reduces the player’s lives by one and updates the hangman drawing to reflect progress toward losing. This visual representation motivates the player to be careful with their guesses.
3. The game continues in a loop, repeatedly asking the player for letters, updating the displayed word, tracking guessed letters, and showing the hangman. This loop ensures the game is interactive and dynamic, reacting to each guess and keeping the player engaged.
4. The game ends in one of two ways: if the player successfully fills in all the blanks, they win. If the player runs out of lives before guessing the word, they lose, and the program reveals the correct word.

In Summary,

* **Random selection of categories and words to keep the game unpredictable.**
* **Tracking guessed letters to avoid repeated input.**
* **Updating the displayed word whenever a correct guess is made.**
* **Reducing lives and showing hangman drawings to visually indicate wrong guesses.**
* **Looping until the game ends, ensuring continuous interaction**.

These elements together create a fully functional, interactive Hangman game where the player can guess letters, see the results immediately, and either win by guessing the word or lose after too many wrong guesses. The combination of visual feedback, word tracking, and randomization makes the game engaging and fun to play.

**Output:**