**SOFTWARE ENGINEERING**

# ASSESSMENT TASK 1 - HANGMAN

**By: An Tran**

**Table of Contents**

[**ASSESSMENT TASK 1  
-  
HANGMAN**](#_yd5ml0cd3wos) 1

Justifications3

UI4

Pseudocode5

Flowchart7

Data Dictionary8

IPO Table10

Logbook11

**Justification for Using Python to Create a Hangman Game**

### **Reasons**

* **Beginner-Friendly** – Python has a simple syntax, making it easy to read and write code, especially for a game like Hangman.
* **Built-in String Manipulation** – Python provides efficient string handling methods, which are essential for checking guessed letters and displaying the word progress.
* **Looping and Conditional Statements** – Python’s while loops and if statements make it easy to manage game logic, such as checking win/loss conditions.
* **Easy User Input Handling** – The input() function allows seamless interaction with the player.
* **Cross-Platform Compatibility** – Python can run on multiple operating systems without modification.
* **Expandable** – Additional features like graphical interfaces using pygame can be added in the future.

**User Interface**

### **Functions**

* **Title & Display**: "DEVELOPER HANGMAN" is shown at the top, with the hangman image updating after incorrect guesses.
* **Word Display**: Hidden words appear as underscores, revealing correct guesses.
* **Alphabet Buttons**: Circular letter buttons (A-Z) arranged in two rows; disappear when clicked.
* **Hint Button**: A blue button on the left reveals one correct letter (usable once per game).
* **Difficulty Selection**: Players choose **Easy (5 mistakes)** or **Hard (3 mistakes)** via two large buttons.
* **End Screen**: Win/Loss message with **Restart (green)** and **Quit (red)** buttons.
* **Color Coding**: White background, black text, red for quitting, green for restarting, and blue for hints for clarity.

**Pseudocode**

### **Hangman**

**BEGIN**

**InitializePygame**

Set screen **width and height** (*WIDTH, HEIGHT*)

Create a **display window** with the specified width and height

Set window title to "*Hangman Game!*"

**Define button variables** (*RADIUS, GAP, letters, etc.*)

Set **starting coordinates** for letter buttons

Create a **list of letters** (*A to Z*) and store their positions

**Define fonts** (*LETTER\_FONT, WORD\_FONT, TITLE\_FONT, BUTTON\_FONT*)

**Load images** for **hangman** (*7 images for hangman stages*)

**Define color** constants (*WHITE, BLACK, RED, GREEN*)

**Define game variables** (*words list, max mistakes, etc.*)

**FUNCTION draw:**

- Fill the screen with **white background**

- Draw the **title** "*DEVELOPER HANGMAN*"

- Display the **word** with guessed letters and **underscores**

- Draw the **letters** (*buttons*) on the screen

- Draw the **hangman image** based on the current hangman status

- Update **the screen**

**FUNCTION display\_message(message):**

- Fill the screen with **white background**

- Display the provided message (like "*You WON!*" or "*You LOST!*")

- Create "*Restart*" **and** "*Quit*" buttons

- Draw the buttons with **text centered within them**

- Update the **screen**

- Return the **restart and quit** button objects

**FUNCTION main:**

- **Initialize** hangman status and **guessed letters**

- Choose a **random word** from the words list

- Reset the **visibility of the letter buttons**

- Set up a **game loop:**

- For each **event**:

- If the player quits, exit the game

- If a letter button is clicked, check if it's in the word:

- If not, increase hangman status

- If yes, mark the letter as guessed

- **Draw the game screen** (*word and letters*)

- Check if the player has **won** (*all letters guessed*) or **lost** (*max mistakes reached*)

- If won or lost, **display the message** and **show restart/quit options**

**FUNCTION handle\_end\_screen:**

- Wait for the player to **either restart or quit** after winning or losing

- If **restart**, return True (*to restart the game*)

- If **quit**, return False (*to exit the game*)

**FUNCTION difficulty\_screen:**

- Display the **difficulty selection** screen (*Easy/Hard*)

- Wait for the player to **choose difficulty** (*Easy sets max mistakes to 5, Hard sets max mistakes to 3*)

- Once the player chooses, return True to **start the game**

**MAIN GAME LOOP:**

- Show the **difficulty selection** screen

- Once difficulty is chosen, start the **main game loop** (*calling main()*)

- If the game **ends** (*either won or lost*), return to the **difficulty screen**

**END**

**Flowchart**

### **Diagram**

**Data Dictionary**

### **Table**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| WIDTH | int | Screen width (800 pixels). |
| HEIGHT | int | Screen height (500 pixels). |
| win | Surface | Pygame display window. |
| RADIUS | int | Radius of letter buttons (20 pixels). |
| GAP | int | Space between letter buttons (15 pixels). |
| letters | list | List storing letter button properties (x, y, letter, visibility). |
| LETTER\_FONT | Font | Font used for letters (size 40). |
| WORD\_FONT | Font | Font used for the word display (size 60). |
| TITLE\_FONT | Font | Font used for game title (size 70). |
| BUTTON\_FONT | Font | Font used for buttons (size 50). |
| images | list | List storing hangman stage images. |
| WHITE | tuple | RGB color code for white (255, 255, 255). |
| BLACK | tuple | RGB color code for black (0, 0, 0). |
| RED | tuple | RGB color code for red (255, 0, 0). |
| GREEN | tuple | RGB color code for green (0, 255, 0). |
| words | list | List of possible words for the game. |
| max\_mistakes | int | Maximum mistakes allowed before losing (3 or 5, based on difficulty). |
| hint\_used | bool | Tracks whether the hint has been used (True or False). |
| hangman\_status | int | Number of incorrect guesses made by the player. |
| guessed | list | List of letters guessed by the player. |
| word | str | The randomly chosen word for the game. |
| hint\_button | Rect | Button for showing a hint (positioned at the middle-left). |
| restart\_button | Rect | Button to restart the game after winning/losing. |
| quit\_butt  on | Rect | Button to quit the game. |

**IPO Chart**

### **Input Process Output Table**



**Logbook**

### **List of Changes Made to Code:**

**28/2/25**

**Added Difficulty Selection (Easy and Hard Modes):**

* Introduced a difficulty screen with **"Easy"** and **"Hard"** buttons.
* Based on user choice, the **maximum mistakes** allowed is set to **5 (Easy)** or **3 (Hard)**.

**5/3/25**

**Resized and Centered Buttons:**

* Adjusted the size and positioning of the buttons (**Easy, Hard, Restart, Quit**) to ensure they are **centered** properly.

**10/3/25**

**Restart and Quit Options after Win/Loss:**

* After winning or losing, a screen with **two buttons** ("Restart" and "Quit") was added.
* These buttons were centered inside **colored boxes**:  
  + **Green for Restart**
  + **Red for Quit**
* Clicking **"Restart"** resets the game, and **"Quit"** closes it.

**Added Hint Button:**

* Introduced a **"Hint"** button placed on the **left side** of the screen.
* The **hint reveals one letter** from the word when clicked and **disappears** after being used.
* The **hint button reappears** when the game restarts.

**17/3/25**

**Game Initialization and Setup:**

* Added **initialization** for the game, including:  
  + Window setup
  + Fonts
  + Button variables
* Called pygame.init() to set up Pygame components.

**21/3/25**

**Improved Button Layout and Centering:**

* Adjusted button positions to **prevent overlapping** and improve **visual organization**.
* Ensured **text inside buttons** is centered for better readability.

**Added Input Handling for Hint and Restart/Quit Options:**

* Implemented logic to **disable the Hint button** after it is used **once** per game.
* Corrected input handling for **"Restart"** and **"Quit"** buttons.

**22/3/25**

**Game State Reset after Restart:**

* When the user clicks **"Restart"**, the following game variables are **reset**:  
  + Guessed letters
  + Hangman status
  + Selected word
* After restarting, the game **returns to the difficulty selection screen**.

**29/3/25**

**Added More Comments for Clarity:**

* Inserted **detailed comments** throughout the code to explain:  
  + Initialization
  + Event handling
  + Game logic

**PEP 8 Compliance:**

* Updated code to **follow PEP 8** standards:  
  + Consistent **indentation**
  + Proper **line spacing**
  + **Shortened long lines**

**Hints Mechanism:**

* Improved **hint system** to provide **better clues** for players.

**30/3/25**

**Ensured All Buttons and Elements are Positioned Correctly:**

* Adjusted **spacing** to prevent **overlapping** between:  
  + Buttons
  + Letters
  + Images

**Game Loop Adjustments:**

* **Refactored** the main game loop to **smoothly handle**:  
  + Difficulty selection
  + Hint usage
  + Restart functionality

**Added Restart Logic After Losing or Winning:**

* When the game ends (**win or lose**), players can choose to:  
  + **Restart the game**
  + **Quit the game**
* These options are **clearly displayed** on the screen.