TASK 1:

CALCULATOR

● Design a simple calculator with basic arithmetic operations.

● Prompt the user to input two numbers and an operation choice.

● Perform the calculation and display the result.

I have designed a simple calculator here, using basic python programming.

My calculator performs basic arithmetic operations like addition, subtraction, multiplication and division.

TASK 2:

PASSWORD GENERATOR

● A password generator is a useful tool that generates strong and random passwords

for users. This project aims to create a password generator application using

Python, allowing user to specify the length and complexity of the password.

● User Input: Prompt the user to specify the desired length of the password.

● Generate Password: Use a combination of random characters to generate a

password of the specified length.

● Display the Password: Print the generated password on the screen.

I have made use of a python in-built module ‘random’. It is use to generate random numbers, select random items, and shuffle data.

🡪random.choice(seq): Used to randomly select letters, digits, and symbols to build the password.

🡪random.shuffle(x): Used to shuffle the combined list of letters, digits, and symbols to ensure the final password is not patterned and is fully randomized.­

TASK 3

ROCK, PAPER, SCISSORS GAME

● User Input: Prompt users to select rock, paper, or scissors.

● Computer Selection: Randomly generate the computer's choice.

● Game Logic: Determine the winner based on user and computer selections. Rock

beats scissors, scissors beat paper, and paper beats rock.

● Display Result: Present user and computer choices. Display outcomes: win, lose, or

tie.

● Score Tracking (Optional): Record user and computer scores for multiple rounds.

● Play Again: Ask if users want another round.

● User Interface: Design a user-friendly interface with clear instructions and

feedback.

Rock wins against sicssor

Scissor wins against paper

Paper wins against rock

0 for rock

1 for paper

2for scissor

Total 9 cases will be there based on user choice and computer choice

**User Choice Computer choice Result**

0 0 Tie

0 1 Computer wins

0 2 user wins

1 0 user wins

1 1 Tie

1 2 Computer wins

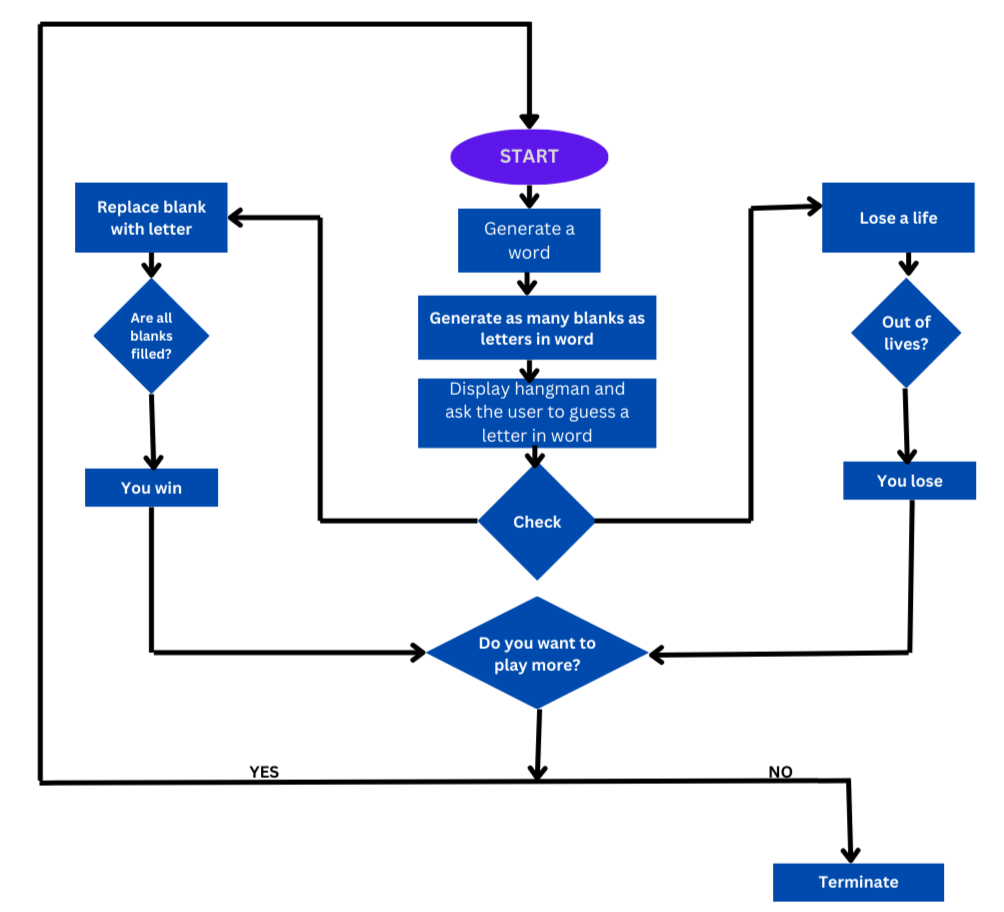
2 0 computer wins

2 1 user wins

2 2 Tie

TASK 4:  
HANGMAN GAME

* Word List: Create a list of words for the game.
* Random Word: Select a random word from the list.
* Initial Display: Show empty hangman figure and underscores for unguessed letters.
* User Input: Prompt the user for a letter guess.
* Check Letter: Validate the guess and check if it's in the word.
* Update State: Reveal correctly guessed letters in the word.
* Hangman Display: Display hangman figure for incorrect guesses.
* Win/Loss Check: Determine win or loss conditions.
* Play Again: Ask if the player wants to play another round.
* User Interface: Design a clear interface showing hangman figure, word state, and feedback.



TASK 5:

* Create a simple Python program that simulates rolling dice. Users can specify the number of sides on the dice and the number of rolls.
* The program generates random numbers for each roll and displays the results.
* It's a quick and fun way to emulate the experience of rolling dice, commonly used in board games or table top role-playing games.
* The simulator allows users to explore the outcomes of dice rolls without the physical dice.

This program prompts the user for entering no. of sides in a dice and the no. of rolls he/she can play for a chance. Using these inputs results of each roll is generated and saved in a list and at the end displayed to the user on which roll he got what result.

I have made use of randint() function of python in-built module ‘random’. It is use to generate random numbers in the range of no. of rolls selected by the user.

Also used the concept of exception handling to smoothly terminate the program and prevent abrupt program crash.

User can enter invalid inputs, so to smoothly handle those events I have made use of try and except block.