1. 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.

```
# Simple Calculator

# Function to perform the calculation
def calculate(num1, num2, operation):
    if operation == '+':
        return num1 + num2
    elif operation == '-':
        return num1 - num2
    elif operation == '*':
        return num1 * num2
    elif operation == '/':
        if num2 != 0:
            return num1 / num2
        else:
            return "Error: Division by zero"
    else:
```

```
        return "Invalid operation"

    # Prompt the user to input two numbers
    and an operation
    num1 = float(input("Enter the first
    number: "))
    num2 = float(input("Enter the second
    number: "))
    operation = input("Choose an operation
    (+, -, *, /): ")

    # Perform the calculation and display the
    result
    result = calculate(num1, num2, operation)
    print(f"The result is: {result}")
```

WORKING:

Input: The user is prompted to enter two numbers and choose an arithmetic operation (+, -, *, /).

Calculation: The program then performs the chosen operation on the two numbers.

Output:

Enter the first number: 5
Enter the second number: 10
Choose an operation (+, -, *, /): *
The result is: 50.0

2. A To-Do List application is a useful project that helps users manage and organize their tasks efficiently. This project aims to create a command-line or GUI-based application using Python, allowing users to create, update, and track their to-do lists.

```
# To-Do List CLI Application

# Initialize an empty list to store tasks
tasks = []

def display_tasks():
    if not tasks:
        print("No tasks available.")
    else:
        print("\nYour To-Do List:")
        for i, task in enumerate(tasks, 1):
            print(f"{i}. {task}")

def add_task(task):
    tasks.append(task)
    print(f"Task '{task}' added.")

def remove_task(index):
    try:
        removed_task = tasks.pop(index - 1)
```

```
        print(f"Task '{removed_task}'
removed.")
    except IndexError:
        print("Invalid task number.")

def main():
    while True:
        print("\nTo-Do List Menu:")
        print("1. View tasks")
        print("2. Add a task")
        print("3. Remove a task")
        print("4. Exit")

        choice = input("Enter your choice
(1-4): ")

        if choice == '1':
            display_tasks()
        elif choice == '2':
            task = input("Enter the task
description: ")
            add_task(task)
```

```
        elif choice == '3':
            display_tasks()
            try:
                index = int(input("Enter the
task number to remove: "))
                remove_task(index)
            except ValueError:
                print("Invalid input. Please
enter a number.")
        elif choice == '4':
            print("Exiting the To-Do List
application.")
            break
        else:
            print("Invalid choice. Please
select a number between 1 and 4.")

if __name__ == "__main__":
    main()

import tkinter as tk
from tkinter import messagebox
```

```
# Create the main application window
root = tk.Tk()
root.title("To-Do List")

# List to store tasks
tasks = []

# Function to update the task list display
def update_task_list():
    listbox.delete(0, tk.END)
    for task in tasks:
        listbox.insert(tk.END, task)

# Function to add a task
def add_task():
    task = task_entry.get()
    if task:
        tasks.append(task)
        update_task_list()
        task_entry.delete(0, tk.END)
    else:
```

```
messagebox.showwarning("Warning",
"Task cannot be empty.")

# Function to remove a selected task
def remove_task():
    try:
        selected_task_index =
listbox.curselection()[0]
        tasks.pop(selected_task_index)
        update_task_list()
    except IndexError:

messagebox.showwarning("Warning",
"No task selected.")

# Create and place widgets
task_entry = tk.Entry(root, width=40)
task_entry.pack(pady=10)

add_button = tk.Button(root, text="Add
Task", command=add_task)
```

```
add_button.pack(pady=5)

remove_button = tk.Button(root,
text="Remove Task",
command=remove_task)
remove_button.pack(pady=5)

listbox = tk.Listbox(root, width=50,
height=10)
listbox.pack(pady=10)

# Run the application
root.mainloop()
```

OUTPUT:

```
        To-Do List Menu:
1. View tasks
2. Add a task
3. Remove a task
4. Exit
Enter your choice (1-4): 2
```

Enter the task description: 2
Task '2' added.

3. 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 users 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.

```
import random
import string

def generate_password(length):
    # Define the character sets to use
in the password
    lower = string.ascii_lowercase
    upper = string.ascii_uppercase
    digits = string.digits
    special = string.punctuation

    # Combine all character sets
    all_characters = lower + upper +
digits + special

    # Generate a random password
```

```
    password =
''.join(random.choice(all_characters)
for _ in range(length))
    return password

def main():
    while True:
        try:
            # Prompt the user to specify
the desired length of the password
            length = int(input("Enter the
desired length of the password: "))
            if length < 1:
                print("Password length must
be at least 1.")
            else:
                # Generate and display the
password
                password =
generate_password(length)
```

```
            print(f"Generated password:
{password}")
            break
        except ValueError:
            print("Invalid input. Please
enter a valid number.")

if __name__ == "__main__":
    main()
```

OUTPUT:

Enter the desired length of the password: 8
Generated password: q*u!%?s

EXPLAIN:

1. **Import Libraries**:

- random for generating random choices.

- string for pre-defined character sets (letters, digits, and punctuation).

2. **Define generate_password Function**:

- Combines lowercase, uppercase letters, digits, and special characters into a single string.
- Randomly selects characters from this combined set to create a password of the specified length.

3. **Define main Function**:

- Prompts the user for the desired password length.
- Checks if the input is a valid integer and ensures the length is positive.
- Generates the password using the generate_password function and prints it.

4. **Run the Application**:

- The main function runs, prompting the user for input and displaying the generated password.

4. User Input: Prompt the user to choose rock, paper, or scissors. Computer Selection: Generate a random choice (rock, paper, or scissors) for the computer. Game Logic: Determine the winner based on the user's choice and the computer's choice. Rock beats scissors, scissors beat paper, and paper beats rock. Display Result: Show the user's choice and the computer's choice. Display the

result, whether the user wins, loses, or it's a tie. Score Tracking (Optional): Keep track of the user's and computer's scores for multiple rounds. Play Again: Ask the user if they want to play another round. User Interface: Design a user-friendly interface with clear instructions and feedback

```
import random

def get_computer_choice():
    """Generate a random choice for the computer."""
    choices = ['rock', 'paper', 'scissors']
```

```
    return random.choice(choices)

def
determine_winner(user_choice,
computer_choice):
    """Determine the winner based
on user and computer choices."""
    if user_choice ==
computer_choice:
        return "It's a tie!"

    if (user_choice == 'rock' and
computer_choice == 'scissors') or \
       (user_choice == 'scissors' and
computer_choice == 'paper') or \
       (user_choice == 'paper' and
computer_choice == 'rock'):
        return "You win!"
```

```
    return "You lose!"

def main():
    user_score = 0
    computer_score = 0

    while True:
        # Get user input
        user_choice = input("Enter
rock, paper, or scissors: ").lower()
        if user_choice not in ['rock',
'paper', 'scissors']:
            print("Invalid choice. Please
enter rock, paper, or scissors.")
            continue

        # Get computer choice
```

```
        computer_choice =
get_computer_choice()
        print(f"Computer chose:
{computer_choice}")

        # Determine the winner
        result =
determine_winner(user_choice,
computer_choice)
        print(result)

        # Update scores
        if result == "You win!":
            user_score += 1
        elif result == "You lose!":
            computer_score += 1

        # Display scores
```

```
        print(f"Score - You:
{user_score}, Computer:
{computer_score}")

        # Ask to play again
        play_again = input("Do you
want to play again? (yes/no):
").lower()
        if play_again != 'yes':
            print("Thanks for playing!")
            break

if __name__ == "__main__":
    main()
```

OUTPUT:

Enter rock, paper, or scissors:
ROCK
Computer chose: rock
It's a tie!
Score - You: 0, Computer: 0
Do you want to play again?
(yes/no):