TO-DO LIST APPLICATION

OBJECTIVES

The primary objective of this project is to create a simple, text-based **To-Do List** application using Python. This application allows users to:

1. Add tasks with a due date.

2. Remove tasks.

3. Mark tasks as completed.

4. View the list of tasks with their status (completed or incomplete).

5. Exit the application.

CONCEPTS OF PYTHON USED

Data Structures

- Lists: Used to store the collection of tasks.
- Dictionaries: Used to store task details like task name, due date, and completion status.

Loops

- While Loop: Used to create an infinite loop that continues running until the user chooses to exit.
- For Loop: Used to iterate through the list of tasks for displaying them.

Conditional Statements

- If-Elif-Else Statements: Used to handle user choices and perform corresponding actions (add, remove, mark as completed, exit).

User Input

- input () Function: Used to take input from the user for various operations.

String Formatting

- f-strings: Used for formatting strings when displaying tasks.

Error Handling

- Basic validation to check if user input is within the valid range for task operations.

EXPLANATION OF CODE

Initial Setup

- An empty list named `tasks` is initialized to store tasks.

Infinite Loop

- An infinite loop starts to keep the application running until the user decides to exit.

Display Tasks

- The current list of tasks is displayed.
- An index is used to number the tasks.
- For each task, the status is checked and set to "Completed" or "Incomplete" based on the task's completion status.
- The tasks are printed with their index, task description, due date, and status.

Display Options

- The available options are displayed to the user.
- The user's choice is taken as input.

Option 1: Add Task

- The user is prompted to enter the task name and due date.
- A new task dictionary is created with the task details and `completed` status set to `False`.
- This dictionary is appended to the `tasks` list.
- A success message is printed to indicate the task has been added.

Option 2: Remove Task

- The user is prompted to enter the task number they want to remove.
- The input is converted to an integer and checked if it is within the valid range (1 to the number of tasks).
- If valid, the task is removed from the `tasks` list using the `pop` method.
- A success message is printed.
- If invalid, an error message is printed.

Option 3: Mark Task as Completed

- The user is prompted to enter the task number they want to mark as completed.
- The input is converted to an integer and checked if it is within the valid range.
- If valid, the `completed` status of the specified task is set to `True`.
- A success message is printed.
- If invalid, an error message is printed.

Option 4: Exit

- If the user selects the option to exit, a message is printed and the `break` statement is executed to terminate the loop and end the program.

Invalid Choice

- If the user enters an invalid choice (anything other than 1, 2, 3, or 4), an error message is printed.