**Project**

**On**

**To-Do List Application**

**\*Overview\***

This To-Do List Application helps users manage their daily tasks by allowing them to add, view, remove, and update the status of tasks. It runs in a loop until the user chooses to exit, making it easy to track tasks dynamically.

**\*Features\***

1. **Add a Task**: Users can add tasks to the to-do list, and each task is assigned a default status of "pending."
2. **View Tasks**: All tasks in the list are displayed with their corresponding status (pending or completed).
3. **Remove a Task**: Users can remove tasks by either specifying the task name or the task index.
4. **Update Task Status**: Users can toggle the status of a task between "pending" and "completed."
5. **Exit the Program**: The program will continue running until the user chooses to exit.

**\*Functions\***

**1. add\_task()**

* **Description**: Prompts the user to input a new task. The task is added to the global list with a status of "pending."
* **Input**: Task description (from the user).
* **Output**: Task is added to the list, and a success message is displayed.

**2. view\_tasks()**

* **Description**: Displays all the tasks in the to-do list with their respective statuses. If the list is empty, it notifies the user.
* **Input**: None.
* **Output**: Prints the tasks in a numbered list format.

**3. remove\_task()**

* **Description**: Allows the user to remove a task by entering either the task number or the task name.
* **Input**: Task number or task name (from the user).
* **Output**: Task is removed, and a success message is displayed. If invalid input is provided, an error message is shown.

**4. update\_task\_status()**

* **Description**: Prompts the user to select a task to update its status (pending/completed). The status is toggled between "pending" and "completed."
* **Input**: Task number (from the user).
* **Output**: Task status is updated, and a confirmation message is displayed.

**5. main()**

* **Description**: The main function runs the program in a loop, displaying a menu and accepting user inputs to perform different actions (add, view, remove, update tasks). The loop continues until the user chooses to exit.
* **Input**: User menu choices (1-5).
* **Output**: Executes functions based on user input.

**\*How to Use\***

1. Run the program.
2. Choose an option from the menu:
   * Add a task (option 1)
   * View tasks (option 2)
   * Remove a task (option 3)
   * Update task status (option 4)
   * Exit (option 5)
3. The program will continue running until you select the exit option.

**\*Code\***

# To-Do List Application

# Global list to store tasks

tasks = []

# Function to add a task to the list

def add\_task():

task = input("Enter the task you want to add: ")

tasks.append({"task": task, "status": "pending"})

print(f'Task "{task}" added successfully!')

# Function to display all tasks

def view\_tasks():

if len(tasks) == 0:

print("No tasks in the list.")

else:

print("\nTo-Do List:")

for idx, task in enumerate(tasks):

print(f"{idx + 1}. {task['task']} - {task['status']}")

print()

# Function to remove a task by index or task name

def remove\_task():

if len(tasks) == 0:

print("No tasks to remove.")

return

view\_tasks()

try:

option = input("Enter the task number or task name to remove: ").strip()

if option.isdigit():

task\_index = int(option) - 1

removed\_task = tasks.pop(task\_index)

else:

task\_names = [task['task'] for task in tasks]

task\_index = task\_names.index(option)

removed\_task = tasks.pop(task\_index)

print(f'Task "{removed\_task["task"]}" removed successfully!')

except (ValueError, IndexError):

print("Invalid task name or index.")

# Function to mark a task as completed or pending

def update\_task\_status():

if len(tasks) == 0:

print("No tasks to update.")

return

view\_tasks()

try:

task\_index = int(input("Enter the task number to update status: ")) - 1

current\_status = tasks[task\_index]['status']

new\_status = "completed" if current\_status == "pending" else "pending"

tasks[task\_index]['status'] = new\_status

print(f'Task "{tasks[task\_index]["task"]}" marked as {new\_status}.')

except (ValueError, IndexError):

print("Invalid task number.")

# Main function to keep the program running until the user exits

def main():

while True:

print("\n--- To-Do List Menu ---")

print("1. Add a Task")

print("2. View Tasks")

print("3. Remove a Task")

print("4. Update Task Status")

print("5. Exit")

choice = input("Choose an option (1-5): ").strip()

if choice == '1':

add\_task()

elif choice == '2':

view\_tasks()

elif choice == '3':

remove\_task()

elif choice == '4':

update\_task\_status()

elif choice == '5':

print("Exiting the program. Goodbye!")

break

else:

print("Invalid choice. Please select a valid option.")

**if \_\_name\_\_ == "\_\_main\_\_"**

**main()**

**\*Output\***

**A screenshot of a computer

Description automatically generated**

**\*Conclusion\***

This project effectively demonstrates core Python concepts such as list manipulation, function design, and control flow. By building this To-Do List Application, users can strengthen their programming skills while creating a practical and interactive tool. It's a great way to practice writing clean, modular code while gaining hands-on experience in developing a simple yet functional application. This project serves as a solid foundation for tackling more complex programs in the future.