# TO-DO LIST APPLICATION USING PYTHON

MOTION CUTS INTERNSHIP

PYTHON PROGRAMMING INTERNSHIP

WEEK 3

**SUBMITTED BY:**

**BANUPRAKASH**

**COLLEGE NAME: DAYANANDA SAGAR COLLEGE OF ENGINEERING**

# CONTENTS

1. INTRODUCTION
2. KEY COMPONENTS
3. ALGORITHM
4. PROGRAM
5. APPLICATIONS
6. ADVANTAGES
7. CONCLUSION

# INTRODUCTION

1. Task Organization: To-Do List applications provide a structured and systematic way to organize tasks. Users can create, edit, and categorize tasks, making it easier to manage various aspects of their lives, such as work, personal, or household responsibilities.

2. Prioritization: Users can prioritize tasks by assigning due dates, deadlines, or importance levels. This helps them focus on critical tasks and maintain productivity.

3. Task Details: To-Do List applications often allow users to include detailed information for each task, such as descriptions, notes, attachments, or sub-tasks. This additional context can be essential for complex projects.

4. Reminders: Many To-Do List apps offer reminder features to alert users when a task is approaching its deadline, ensuring they don't miss important commitments.

5. Accessibility: Most modern To-Do List applications are available on multiple platforms, including smartphones, tablets, and desktop computers, enabling users to access their lists from anywhere.

6. Collaboration: Some To-Do List applications support collaboration, enabling multiple users to share and work on lists together. This is useful for project teams, family schedules, and more.

7. Synchronization: Many applications offer synchronization across devices, ensuring that changes made on one platform are reflected on all other connected devices, and keeping lists up to date.

8. Time Management: To-Do List applications can help users manage their time effectively by breaking down large tasks into smaller, manageable sub-tasks and tracking progress.

9. Productivity Boost: By providing a clear and organized overview of tasks, these applications can boost productivity by reducing stress and helping users stay on top of their commitments.

10. Historical Records: To-Do List applications often store completed tasks, providing users with a historical record of their accomplishments, which can be motivating and helpful for future planning.

# ALGORITHM

1. Create a `Task` class:
- Initialize the `Task` class with a description and a completed status.

2. Create a `ToDoList` class:
- Initialize the `ToDoList` class with an empty list to store tasks.

3. Implement methods in the `ToDoList` class:

- `add_task(description)`: Create a new task with the provided description and add it to the list of tasks.
- `remove_task(index)`: Remove a task at the specified index from the list.
- `update_task(index, new_description)`: Update the description of a task at the specified index.
- `mark_task_completed(index)`: Mark a task at the specified index as completed.
- `list_tasks()`: Display a list of tasks with their descriptions and completion status.

4. Create a `main` function:
- Create an instance of the `ToDoList` class.
- Enter a loop to display a menu to the user and handle their choices.

5. Inside the loop, prompt the user for a choice:
- If the choice is "1," prompt the user for a task description and add the task to the to-do list.
- If the choice is "2," prompt the user for the index of the task to remove and remove it from the list.
- If the choice is "3," prompt the user for the index of the task to update and the new description, then update the task.
- If the choice is "4," prompt the user for the index of the task to mark as completed and mark it as completed.
- If the choice is "5," display the list of tasks.
- If the choice is "6," exit the loop and end the program.

6. Repeat the loop until the user chooses to quit (option 6).

This algorithm outlines the flow of the To-Do List application code and how it allows users to perform various actions on their task list. Users can add tasks, remove tasks, update task descriptions, mark tasks as completed, and list tasks. The code provides a command-line interface for interacting with the to-do list.

# PROGRAM :

```
class Task:
    def __init__(self, description, completed=False):
        self.description = description
        self.completed = completed

class ToDoList:
    def __init__(self):
        self.tasks = []

    def add_task(self, description):
        task = Task(description)
        self.tasks.append(task)

    def remove_task(self, index):
        if 0 <= index < len(self.tasks):
            del self.tasks[index]

    def update_task(self, index, new_description):
        if 0 <= index < len(self.tasks):
            self.tasks[index].description = new_description

    def mark_task_completed(self, index):
        if 0 <= index < len(self.tasks):
            self.tasks[index].completed = True

    def list_tasks(self):
        for i, task in enumerate(self.tasks):
            status = "Completed" if task.completed else "Not Completed"
            print(f"{i+1}. {task.description} - {status}")

def main():
    to_do_list = ToDoList()
    while True:
        print("\nTo-Do List Menu:")
        print("1. Add Task")
        print("2. Remove Task")
        print("3. Update Task")
        print("4. Mark Task as Completed")
        print("5. List Tasks")
        print("6. Quit")
```

```
        choice = input("Enter your choice: ")

        if choice == "1":
            description = input("Enter task description: ")
            to_do_list.add_task(description)
        elif choice == "2":
            index = int(input("Enter the task number to remove: ")) - 1
            to_do_list.remove_task(index)
        elif choice == "3":
            index = int(input("Enter the task number to update: ")) - 1
            new_description = input("Enter the new task description: ")
            to_do_list.update_task(index, new_description)
        elif choice == "4":
            index = int(input("Enter the task number to mark as completed: ")) - 1
            to_do_list.mark_task_completed(index)
        elif choice == "5":
            to_do_list.list_tasks()
        elif choice == "6":
            break
        else:
            print("Invalid choice. Please try again.")

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

# APPLICATIONS

- Personal Task Management: - Individuals can use this code to manage their daily, weekly, or monthly tasks and responsibilities. It helps them stay organized and on top of their personal commitments.

- Work Task Tracking:- Professionals can use the code to track their work-related tasks and projects. They can prioritize tasks, set deadlines, and monitor their progress.

- Project Management: - Project managers and team leads can adapt and expand the code to create a simple project management tool for tracking tasks and milestones within a project.

- Academic Assignments:- Students can use the code to keep track of assignments, exams, and project deadlines. It helps them manage their academic workload effectively.

- Shopping Lists: - Users can repurpose the code for creating and managing shopping lists. They can add and remove items as they shop, ensuring they don't forget any essentials.

- Event Planning: Event planners can use the code to keep track of tasks related to event planning, such as contacting vendors, sending invitations, and scheduling activities.

- Household Chores:- Families can use the code to assign and track household chores among family members. It encourages responsibility and helps maintain a well-organized household.

- Fitness and Health Goals:- Individuals pursuing fitness and health goals can use the code to create workout routines, track progress, and set reminders for exercise sessions and dietary goals.

- Hobby or DIY Projects: - Enthusiasts working on hobbies or do-it-yourself (DIY) projects can use the code to manage their project tasks, supply lists, and project timelines.

- Collaborative Work:- Teams and collaborators can use the code to maintain shared to-do lists, enhancing coordination and ensuring that everyone is on the same page.

- Time Management and Productivity:- Users interested in improving time management and productivity can leverage the code to structure their tasks, allocate time for each, and stay organized.

- Personal Development:- People looking to track their personal development goals, such as reading lists, skill-building, or self-improvement tasks, can use the code to stay focused.

# ADVANTAGES

- Simplicity and Ease of Use - The code provides a straightforward and easy-to-understand interface for managing tasks. Users can quickly grasp how to add, remove, update, and mark tasks as completed.

- Customization - Users can easily customize the code to adapt it to their specific needs and preferences. It can be extended with additional features and functionalities if desired.

- Task Organization - The code helps users keep tasks organized by allowing them to categorize, prioritize, and provide detailed descriptions for each task.

- Prioritization - Users can assign due dates or prioritize tasks, which helps them focus on high-priority items and manage their time effectively.

- Flexibility - The code is versatile and can be applied to a wide range of tasks and applications, from personal to professional, academic, and more.

- Accessibility - The code can run on various platforms, making it accessible to users on different devices, including computers, smartphones, and tablets.

- Synchronization - Many To-Do List applications offer synchronization, ensuring that task lists are consistent across multiple devices, allowing users to access their lists wherever they are.

- Reminders - The code includes a reminder feature that can help users stay on top of their tasks by providing timely alerts.

- Collaboration: - Users can collaborate and share task lists with others, making it useful for team projects, household chores, or family scheduling.

- Time Management - The code helps users break down large tasks into smaller, manageable sub-tasks, which is a valuable time management technique.

- Productivity Boost - By providing an organized task list, the code can reduce stress and improve productivity by ensuring that users don't forget important commitments.

- Historical Records - The code keeps a record of completed tasks, serving as motivation and a reference for future planning and progress tracking.

- Cost-Effective - The code is open-source and can be used without any licensing fees or costs, making it an affordable solution for task management.

# CONCLUSION

In conclusion, the Python-based to-do list application project has successfully provided a user-friendly and efficient solution for organizing tasks and enhancing productivity. Through our development efforts, we have created a functional and intuitive tool that allows users to manage their tasks with ease. With features like task creation, prioritization, and completion tracking, our application serves as a valuable asset for improving time management. The project demonstrates the power of Python in creating practical and accessible software solutions, and we look forward to future enhancements and improvements to further enhance its utility.