中山大学计算机学院

软件工程

课程	**软件工程**
姓名/学号	**马福泉 23336179**
文档题目	**YatToDo 开发文档**
实践时间	**2025 年 4 月**

一、项目简介

YatToDo 是一个简洁高效的待办事项管理应用程序，帮助用户组织和跟踪日常任务。该应用程序提供了一个用户友好的界面，让您能够轻松创建、编辑、删除和管理待办事项，提高工作和生活效率。

二、项目结构

```
YatToDo/
├─── code/
|    ├─── main.py            # 程序入口点
|    ├─── config.py          # 配置管理模块
|    ├─── todo.py            # 待办事项核心功能
|    ├─── ui.py              # 用户界面模块
|    └─── utils.py           # 工具函数
├─── data/
|    ├─── todos.json         # 待办事项数据文件
|    └─── backup/            # 自动备份目录
├─── config.json            # 用户配置文件
└─── README.md              # 项目说明文档
```

三、功能

（一）概述

- 创建、编辑、删除和查找待办事项
- 为待办事项设置优先级和截止日期
- 标记待办事项为已完成状态

- 根据不同条件筛选和排序待办事项
- 自动备份功能，防止数据丢失
- 自定义主题和界面设置
- 支持导入/导出待办事项列表
- 可调整窗口大小和字体大小

（二）实现（具体见代码）

1. 增加任务（增）

在 ui.py 中，用户通过输入框输入任务内容、选择优先级和截止日期后，点击“添加任务”按钮，触发 on_add_task 方法。

该方法获取输入的任务信息，创建一个包含任务内容、优先级、截止日期等信息的字典对象 task_data，并将其添加到 self.tasks 列表中。

调用 save_tasks 函数（位于 data.py），将更新后的 self.tasks 列表保存到文件中。

调用 apply_filter 方法更新任务列表的显示，使新添加的任务能够按照当前的过滤条件显示在界面上。

2. 删除任务（删）

在 ui.py 中，用户选中要删除的任务后，点击“删除任务”按钮，触发 on_delete_task 方法。

该方法通过 self.listbox.curselection()获取选中的任务在 self.filtered_tasks 列表中的索引，再通过 self.tasks.index()找到该任务在 self.tasks 列表中的索引。

使用 del 语句根据索引从 self.tasks 和 self.filtered_tasks 列表中删除对应的任务。

调用 save_tasks 函数保存更新后的任务列表到文件。

调用 self.listbox.delete()方法删除界面上的任务显示，并调用 self._update_status()方法更新状态栏信息。

3. 查询任务（查）

在 ui.py 中，用户可以通过菜单栏的“编辑”->“查找”打开查找对话框，输入关键词进行搜索。

调用 search_tasks 函数（位于 data.py），传入任务列表、关键词、是否区分大小写、完成状态过滤条件、优先级过滤条件和日期范围等参数。

search_tasks 函数会遍历任务列表，根据传入的过滤条件筛选出符合条件的任务，并返回匹配的任务列表。

在查找对话框中，将搜索结果展示在列表框中供用户查看。

4. 修改任务（改）

在 ui.py 中，用户选中要修改的任务后，点击“修改任务”按钮，触发 on_modify_task 方法。

该方法获取用户在输入框中输入的新任务内容、优先级和截止日期，通过 self.listbox.curselection()获取选中的任务在 self.filtered_tasks 列表中的索引，再找到该任务在 self.tasks 列表中的索引。

使用新任务信息更新 self.tasks 列表中对应任务的数据，并调用 save_tasks 函数保存更新后的任务列表到文件。

调用 apply_filter 方法更新任务列表的显示，使修改后的任务能够按照当前的过滤条件显示在界面上。

5. 备份功能实现

在 ui.py 中，用户通过菜单栏的“文件”->“备份数据”触发备份操作。

调用 backup_tasks 函数（位于 data.py），传入任务列表和备份目录路径。

backup_tasks 函数会检查备份目录是否存在，如果不存在则创建。

使用时间戳生成备份文件名，将任务列表以 JSON 格式保存到备份文件中。

自动清理过旧的备份文件，保留最新的指定数量（如 5 个）备份文件。

6. 排序功能实现

在 ui.py 中，用户通过菜单栏的“视图”->“排序方式”选择排序方式，或者点击界面上的排序按钮触发排序操作。

调用 sort_tasks 方法，传入排序类型（如“priority”按优先级排序、“date”按截止日期排序或“none”不排序）和是否倒序的标志。

根据排序类型，调用不同的排序函数（如 sort_tasks_by_priority 或 sort_tasks_by_date）对 self.filtered_tasks 列表进行排序。

sort_tasks_by_priority 函数会根据任务的优先级（高、中、低）进行排序，优先级高的任务排在前面。

sort_tasks_by_date 函数会根据任务的截止日期进行排序，截止日期近的任务排在前面。

排序完成后，调用 reload_tasks 方法更新任务列表的显示，使任务按照新的排序顺序显示在界面上。

7. 设置功能实现

在 ui.py 中，用户通过菜单栏的“设置”->“设置”打开设置对话框，或者点击“切换主题”选项触发设置操作。

对于主题切换，调用 toggle_theme 方法，改变 self.current_theme 的值，并调用 apply_theme 方法更新界面的样式，包括背景颜色、字体颜色等，以应用新的主题。

对于其他设置项（如自动备份、窗口大小等），通过 load_config 和 save_config 函数（位于 config.py）加载和保存配置。

load_config 函数会从配置文件中读取配置信息，如果配置文件不存在则创建默认配置。

save_config 函数会将更新后的配置信息保存到配置文件中。

8. 搜索功能实现

在 ui.py 中，用户通过菜单栏的“编辑”->“查找”打开查找对话框，输入关键词进行搜索。

调用 search_tasks 函数（位于 data.py），传入任务列表、关键词、是否区分大小写、完成状态过滤条件、优先级过滤条件和日期范围等参数。

search_tasks 函数会遍历任务列表，根据传入的过滤条件筛选出符合条件的任务，并返回匹配的任务列表。

四、使用

（一）功能说明

添加任务：在任务输入框中输入任务内容，选择优先级和截止日期，点击“添加任务”按钮。

删除任务：选中要删除的任务，点击“删除任务”按钮。

修改任务：选中要修改的任务，编辑任务输入框中的内容，选择新的优先级和截止日期，点击“修改任务”按钮。

标记任务为已完成/未完成：选中任务，点击“标记完成”按钮。

排序任务：通过菜单栏中的“视图”->“排序方式”选择排序方式。

搜索任务：通过菜单栏中的“编辑”->“查找”打开查找对话框，输入关键词进行搜索。

备份任务：通过菜单栏中的“文件”->“备份数据”进行任务备份。

导出任务：通过菜单栏中的“文件”->“导出为文本”将任务导出为文本文件。

切换主题：通过菜单栏中的“设置”->“切换主题”切换应用主题。

设置：通过菜单栏中的“设置”->“设置”打开设置对话框，进行相关设置。

（二）安装与运行

确保已安装 Python 环境。

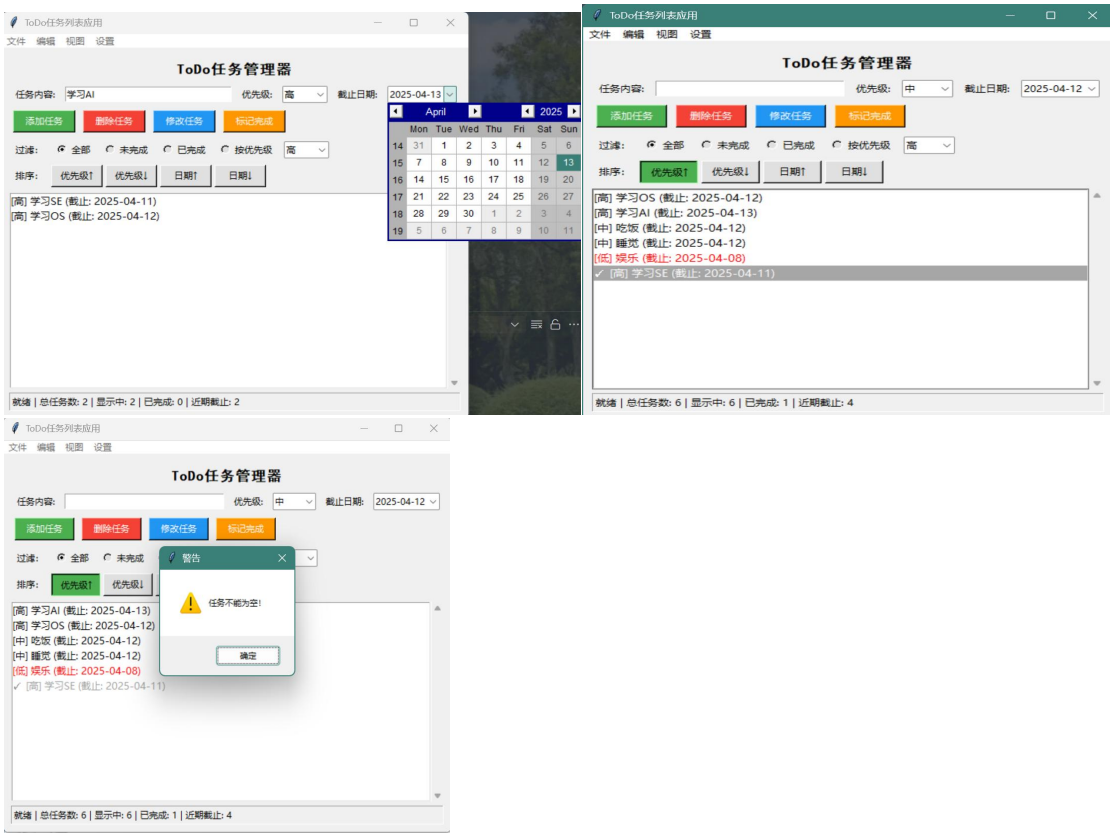
安装 tkcalendar 库：`pip install tkcalendar`。

将项目文件下载到本地。

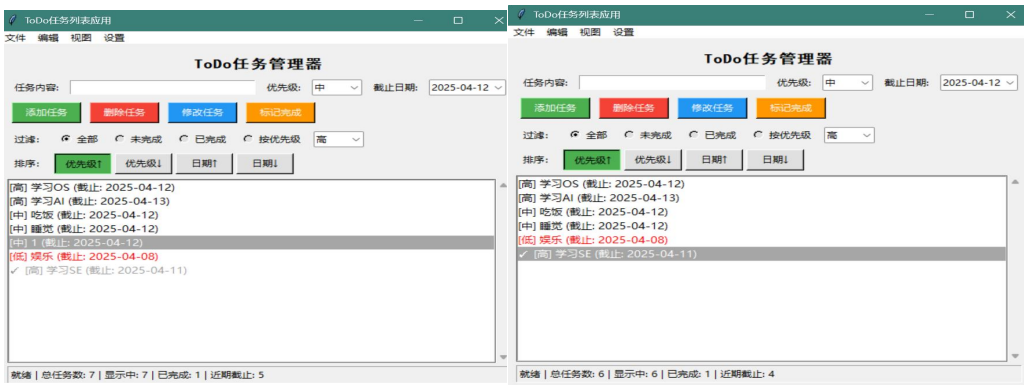
运行 `main.py` 文件启动应用

（三）使用示例

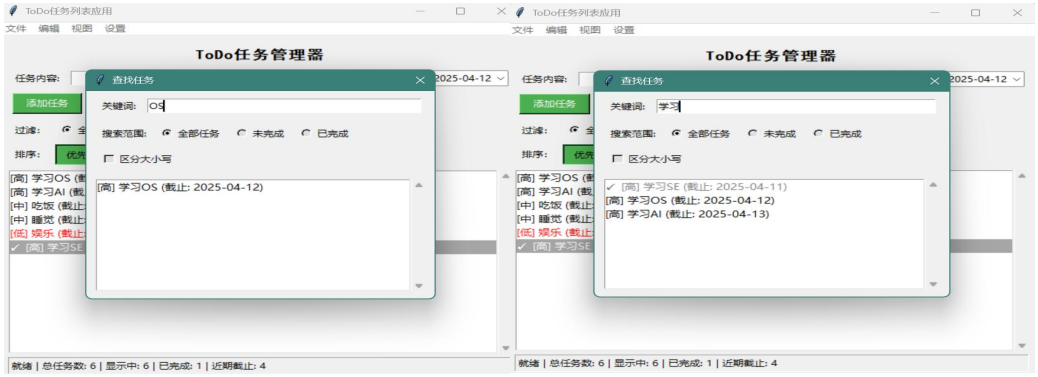
1. 增加任务



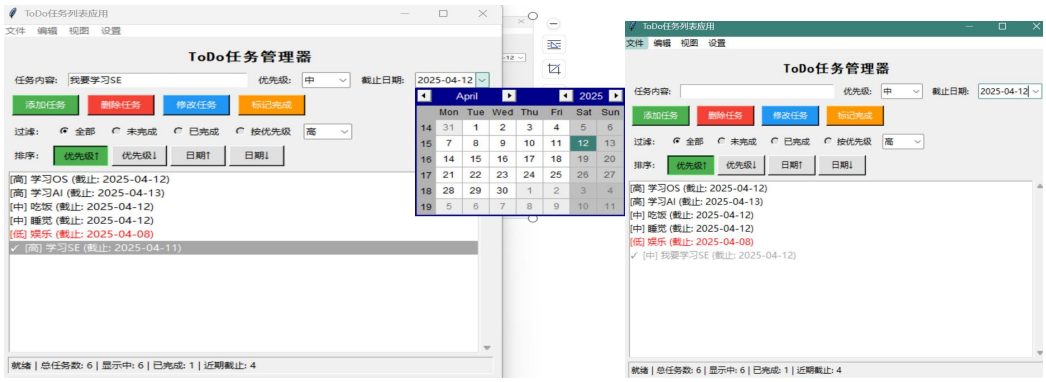
增加后默认是添加顺序
下方显示任务数，近期截止数
已完成的会打 √
已过期的事件会标红
未输入任务会自动报错
2. 删除任务
选定任务删除即可



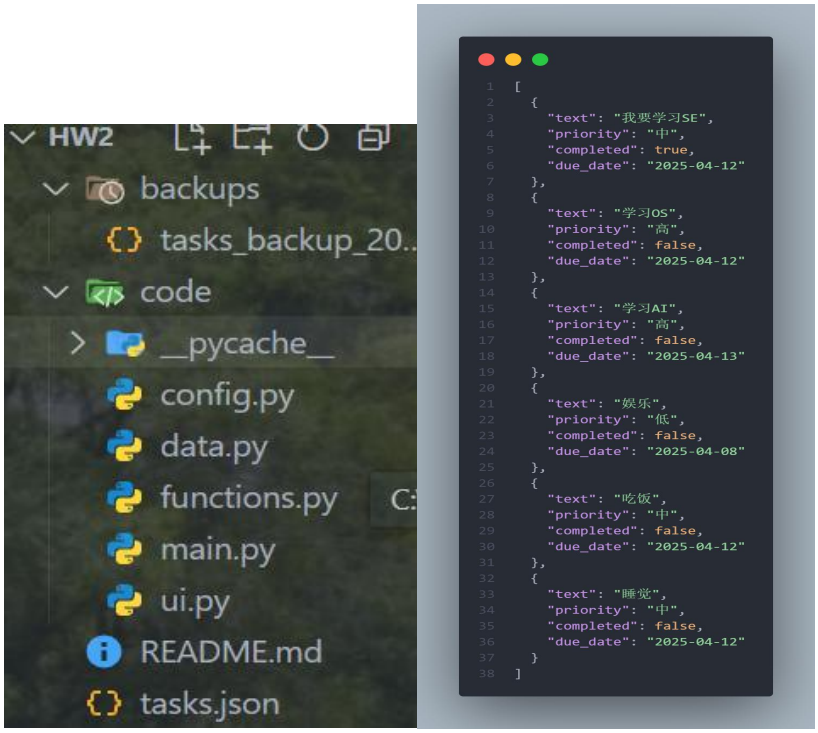
3. 查询任务
按 CTRL+F 输入关键词，可以选择区分大小写



4. 修改任务
选定任务，改变任务名称，优先度，截止时期



5. 备份功能
点击文件点击备份，文件就会保存 backups



6. 排序功能

支持四种排序

7. 设置功能

五、开发过程

（一） 借助网络资源和 AI 大模型，确定目标需求和功能规划，并进行技术选型。

（二） 确定技术选型和功能需求后，借助 AI 确定代码规模大小，并确定项目结构。

（三） 使用 Kimi 或者 Deepseek 编写 prompt，并且根据需求进行修改，然后将 prompt 给 vsc 的 copilot 编辑代码。

比如以下 prompt：

1. main.py

功能：主入口文件，启动应用，创建主窗口并设置基本属性，调用 ToDoAppUI 类初始化界面，处理程序关闭逻辑。

2. functions.py

功能：提供任务操作的基础函数，如添加、删除、修改任务，以及标记任务为已完成或未完成。

3. data.py

功能：负责任务数据的存储、加载、备份、导出和搜索，以及任务排序和统计信息的获取。

4. config.py

功能：管理应用配置，加载、保存和更新配置文件，支持主题切换等设置。

5. ui.py

功能：定义用户界面，实现任务列表显示、任务操作按钮、菜单栏等功能，提供丰富的交互体验。

（四） 进行运行测试，并且根据所需扩展功能不断提交 prompt 进行代码优化。

（五） 阅读代码，借助 AI 添加注释，增强代码可读性。

（六） 检查代码，测试样例，编写开发文档。

六、未来开发方向

（一） 移动应用开发

将 Todo 软件扩展到移动平台，开发 iOS 和 Android 应用，使用户能够在手机和平板上随时随地管理任务。

利用移动设备的特性，如语音输入、日历集成、推送通知等，提供更便捷的用户体验。

（二） 云同步功能

实现任务数据的云同步，用户可以在不同设备上登录账号，实时同步任务列表和状态。

提供数据备份和恢复功能，确保用户数据的安全性和可靠性。

（三） 团队协作功能

增加团队协作功能，允许用户创建团队任务列表，邀请成员加入并分配任务。

提供任务进度跟踪和评论功能，方便团队成员之间的沟通和协作。

（四） 智能提醒与预测

利用 AI 技术进一步优化任务提醒功能，根据用户的习惯和任务优先级，智能安排提醒时间。

实现任务完成时间预测，为用户提供合理的时间规划建议。

（五） 数据分析与可视化

提供任务完成情况的数据分析功能，如生成任务完成率、时间分布等图表，帮助用户更好地了解自己的任务管理情况。

增加任务标签功能，用户可以根据项目、类型等对任务进行分类，并提供基于标签的数据分析。

（六） 用户体验优化

持续关注用户反馈，优化界面设计和交互流程，使操作更加直观和便捷。

提供丰富的自定义选项，让用户可以根据自己的喜好调整主题、字体、布局等。

（七） 安全性增强

如果涉及用户账号和数据同步功能，需要加强安全措施，如使用加密技术保护用户数据，防止数据泄露。

对用户输入进行严格验证，防止 SQL 注入、XSS 等安全漏洞。

（八） 国际化支持

如果希望应用面向更广泛的用户群体，可以考虑支持多语言，提供国际化界面和功能。

对应用中的文本内容进行本地化处理，支持不同语言的显示和输入