Aufgabenliste

Grundlagen Programmierung

Felix Ritter, Alessio Saccullo, Tayfur Ayubi

14.12.2025

Inhalt

1.Einleitung

To Do Listen werden im Alltag regelmässig genutzt, daher lag es nahe, ein eigenes Tool zu entwickeln, das genau die Funktionen bietet, die man wirklich braucht ohne unnötige Komplexität. Ziel war eine einfache und zuverlässige Anwendung, mit der Aufgaben erstellt, verwaltet, gefiltert und dauerhaft gespeichert werden können. Der Fokus liegt bewusst auf den Kernfunktionen. Erweiterungen wie mehrbenutzerfähige Systeme oder Synchronisation wurden nicht umgesetzt. Damit bleibt das Projekt technisch überschaubar, gut wartbar und sowohl für Alltagsaufgaben als auch für kleine Projekte geeignet.

2. Anforderungen

2.2 Nicht funktionale Anforderungen

Der Aufgabenlisten Manager soll ohne komplizierte Installation sofort lauffähig sein. Das Projekt besteht aus mehreren Python Skripten, die über eine Batch Datei gestartet werden können. Dadurch ist sichergestellt, dass der Nutzer keine zusätzlichen Abhängigkeiten manuell konfigurieren muss.

Die Bedienung erfolgt ausschliesslich über die Konsole. Das Programm führt den Nutzer mit klaren numerischen Eingaben durch die Funktionen wie neue Aufgabe erstellen, Status ändern, sortieren oder laden. Damit bleibt die Anwendung einfach verständlich und für Anfänger gut benutzbar.

Ein weiterer nicht funktionaler Anspruch ist die dauerhafte Speicherung der Aufgaben. Das Projekt speichert die Daten in einer Textdatei, sodass der aktuelle Stand beim nächsten Start wieder geladen werden kann. Die Speicherung erfolgt zuverlässig und ohne komplexe Dateiformate, was die Wartbarkeit vereinfacht.

Das Programm muss stabil laufen, auch wenn der Benutzer falsche Eingaben macht. In mehreren Modulen wird geprüft, ob Eingaben gültig sind, bevor sie verarbeitet werden. Das verhindert Abstürze und sorgt für ein vorhersehbares Verhalten.

Insgesamt soll die Anwendung leicht verständlich, technisch unkompliziert und robust genug sein, um alltägliche Aufgabenlisten zuverlässig zu verwalten.

3. Systementwurf / Konzept

3.1 Programmaufbau

Das Projekt ist modular aufgebaut. Jede Hauptfunktion der To-Do-Verwaltung befindet sich in einem eigenen Python Modul. Dadurch bleibt der Code übersichtlich, leicht nachvollziehbar und erweiterbar.

app.py

Dieses Modul bildet die zentrale Steuereinheit des Programms. Nach dem Start wird der Startbildschirm angezeigt, anschließend die Aufgabenübersicht geladen und das Hauptmenü in einer Endlosschleife ausgeführt.
Alle Benutzereingaben werden hier verarbeitet. Je nach Menüauswahl ruft app.py die dafür zuständigen Module (new, edit, delete, status, sort, filter) auf.

graphics.py

graphics.py ist ausschließlich für die Darstellung der Oberfläche in der Konsole zuständig. Es zeigt den Startbildschirm (ASCII-Art), das Hauptmenü, Übersichten sowie die Titel der Untermenüs (Neu, Bearbeiten, Löschen, Status, Sortieren, Filtern).
Das Modul führt keine Eingabelogik aus – es erstellt nur eine einheitliche, gut lesbare Ausgabe.

new.py

Dieses Modul erstellt neue Aufgaben. Es fragt den Benutzer nach:

- Aufgabenbeschreibung
- Priorität (A, B oder C)
- Fälligkeitsdatum

Der Status wird automatisch auf „offen“ gesetzt, und das Erstelldatum wird automatisch hinzugefügt. Anschließend wird die nächste freie ID ermittelt und der neue Datensatz an die Datei daten/tasks.txt angehängt. Optional kann direkt eine weitere Aufgabe eingegeben werden.

edit.py

Das Modul bearbeitet bestehende Aufgaben anhand ihrer ID.
Der Benutzer kann:

- die Beschreibung ändern
- die Priorität ändern
- das Fälligkeitsdatum ändern

Nach einer Änderung wird die jeweilige Zeile aktualisiert und die gesamte Datei erneut gespeichert, damit der Datensatz dauerhaft übernommen wird.

delete.py

Dieses Modul löscht eine Aufgabe anhand ihrer ID.
Vor dem Löschen erfolgt eine Sicherheitsabfrage (J/N). Bei Bestätigung wird die Aufgabe aus dem Arbeitsspeicher entfernt und anschließend die Datei neu geschrieben.
Dabei werden alle IDs automatisch neu durchnummeriert, damit wieder eine saubere Reihenfolge 1..n entsteht.

status.py

Hier wird der Status einer Aufgabe geändert. Das Modul sucht anhand der ID den entsprechenden Eintrag und schaltet den Status zwischen „offen“ und „erledigt“ um.
Beim Zurücksetzen auf „offen“ erfolgt erneut eine Bestätigung. Danach werden alle Aufgaben mit aktualisiertem Status in daten/tasks.txt gespeichert.

sort.py

Dieses Modul liest alle Aufgaben ein und bietet verschiedene Sortierkriterien an:

- Fälligkeitsdatum
- Erstelldatum
- Status
- Priorität
- Beschreibung
- ID

Nach der Sortierung wird die neue Reihenfolge dauerhaft in daten/tasks.txt gespeichert.
Dadurch bleibt die Sortierung auch nach einem Neustart des Programms erhalten.

filter.py

Führt eine temporäre Filterung der Aufgaben durch.
Der Benutzer kann filtern nach:

- Status (offen/erledigt)
- Priorität
- Teiltext in der Beschreibung

Die Datei bleibt unverändert – es wird nur die Ausgabe gefiltert, während alle Daten im Arbeitsspeicher verbleiben.

load.py

Dieses Modul zeigt alle vorhandenen Aufgaben im Tabellenformat an.
Dabei werden Statuswerte automatisch in lesbare Begriffe („offen“ / „erledigt“) umgewandelt.
load.py ist ausschließlich für das Laden und Anzeigen zuständig, nicht für das Speichern.

clear.py

Hilfsmodul zum Löschen des Konsolenbildschirms (cls unter Windows bzw. clear unter Unix).
Es dient nur der Übersichtlichkeit der Darstellung und löscht keine Aufgaben.

ToDoApp.bat

Eine Windows-Batchdatei, die das Programm per Doppelklick startet, ohne dass der Benutzer einen Python-Befehl eingeben muss.

3.2 Datenmodell

Das Projekt speichert alle Aufgaben dauerhaft in der Textdatei daten/tasks.txt. Jede Zeile der Datei repräsentiert genau eine Aufgabe. Eine Aufgabe besteht aktuell aus sechs Attributen in fester Reihenfolge:

1. ID
   Ganzzahl, als Text gespeichert. Die ID identifiziert die Aufgabe eindeutig. Beim Löschen werden alle IDs neu durchnummeriert, damit keine Lücken entstehen.
2. Beschreibung
   Textfeld, enthält die eigentliche Aufgabenbeschreibung. Längere Texte werden in der Übersicht automatisch gekürzt dargestellt, um die Tabellenbreite zu halten.

3. Priorität
   Buchstabe A, B oder C (intern in Kleinbuchstaben gespeichert).
   - A = hohe Priorität
   - B = mittlere Priorität
   - C = niedrige Priorität
4. Status
   String, entweder offen oder erledigt. Beim Laden werden ältere Werte wie True/False automatisch in diese Zustände umgewandelt.
5. Erstelldatum
   Datum, an dem die Aufgabe angelegt wurde, im Format dd.mm.yyyy. Das Datum wird automatisch gesetzt.
6. Fälligkeitsdatum
   Datum, das vom Benutzer eingegeben wird, üblicherweise im Format dd.mm.yyyy. Das Format wird nicht technisch geprüft, da die Verarbeitung bewusst einfach gehalten ist.

Das Datenmodell ist bewusst einfach gehalten. Es wird auf komplexe Formate wie JSON, CSV oder Datenbanken verzichtet, um die Nachvollziehbarkeit, Wartbarkeit und Projektbewertung zu erleichtern.

3.3 Benutzerführung

Die Anwendung wird ausschließlich in der Konsole bedient und zeigt dem Nutzer nach jedem Schleifendurchlauf die Hauptübersicht. Die Steuerung erfolgt über numerische Eingaben.

Hauptmenü

```
[1]: Aufgabe hinzufügen
[2]: Aufgabe bearbeiten
[3]: Aufgabe löschen
[4]: Aufgabenstatus ändern
[5]: Augaben sortieren
[6]: Aufgaben filtern
[0]: Programm beenden

Was möchtest du tun? :>
```

Untermenüs

Hinzufügen (new.py)

- Beschreibung eingeben
- Priorität wählen
- Fälligkeit eingeben
- Speichern bestätigen oder abbrechen
- Option: weitere Aufgaben hinzufügen

Bearbeiten (edit.py)

- ID wählen
- Auswahl des Attributs:
    - Beschreibung
    - Priorität
    - Fälligkeitsdatum
- Änderungen werden sofort gespeichert
- Wiederholbare Bearbeitung möglich

Löschen (delete.py)

- ID eingeben
- Sicherheitsabfrage (J/N)
- Neu-Nummerierung aller IDs

Status ändern (status.py)

- ID eingeben
- Task wird zwischen offen ↔ erledigt umgeschaltet
- Rückbestätigung beim Zurücksetzen auf „offen“

Sortieren (sort.py)
Sortierung ist dauerhaft – Datei wird überschrieben. Sortieroptionen:

- Fälligkeit
- Erstelldatum
- Status
- Priorität
- Beschreibung
- ID

Filtern (filter.py)
Filter ist nur temporär – Datei bleibt unverändert. Filteroptionen:

- Status
- Priorität
- Beschreibung (Teilstring)

Nach jedem Untermenü kehrt die Anwendung zum Hauptmenü zurück.

4. Implementierung

4.1 Wichtige Funktionen

Aufgaben anlegen

Fragt Beschreibung, Status und Priorität ab. Die Werte werden in die interne Aufgabenliste übernommen und dauerhaft gespeichert.

Aufgaben löschen

Entfernt einen Eintrag anhand der angegebenen Listenposition. Danach wird die Datei neu geschrieben.

Aufgaben bearbeiten

Ersetzt Werte eines bestehenden Tasks. Diese Änderung wird direkt gespeichert.

Status ändern

Sucht den gewünschten Task und ersetzt den Statuswert.

Sortieren

Sortiert die Datenliste anhand definierter Kriterien (z. B. Status oder Priorität). Die Sortierung ist nur sichtbar, die Datei bleibt unverändert.

Filtern

Die Filterfunktion zeigt nur Elemente an, die den ausgewählten Kriterien entsprechen. Dies geschieht temporär in der Ansicht. Die Datei wird nicht verändert, es findet keine permanente Reorganisation statt.

Speichern & Laden

Die Daten werden mit open() in Textdateien geschrieben bzw. gelesen. Alle Attribute einer Aufgabe werden als Zeichenkette gespeichert und beim Lesen wieder getrennt. So bleibt die Struktur stabil.

4.2 Fehlerbehandlung

Das Programm prüft in allen Menüs die Nutzereingaben. Zahleneingaben werden validiert, damit keine Buchstaben oder ungültigen Indexwerte akzeptiert werden. Bei falscher Eingabe erscheint ein Hinweis, und der Nutzer wird zurückgeführt, ohne dass die Anwendung abstürzt.

Fehlt die Datei beim Laden, wird dies durch try und except abgefangen. In diesem Fall wird eine neue Datei angelegt und der Programmablauf nicht unterbrochen. Wenn die Datei leer ist, startet die Liste ohne Einträge.

Die App ist robust gegenüber falschen Eingaben:

- **Ungültige IDs** → Meldung + erneute Abfrage
- **Nicht-numerische Eingaben** → Fehlerausgabe
- **Ungültige Prioritäten** → erneute Eingabeaufforderung
- **Fehlende Datei** → Meldung statt Absturz
- **Bestätigungen (J/N)** → Wiederholung bei falscher Eingabe

Durch die Schleifenstruktur wird kein Programmabsturz erzeugt – der Nutzer wird immer wieder sauber zurückgeführt.

5. Testen

5.1 Testkonzept

Alle Funktionen wurden anhand des CRUD-Prinzips getestet. CRUD steht für Create, Read, Update und Delete, also die grundlegenden Operationen eines Datenmanagement Systems.

- Create (Erstellen): Es wurde geprüft, ob neue Aufgaben korrekt aufgenommen, angezeigt und in der Datei gespeichert werden.
- Read (Anzeigen/Laden): Das Laden aus der Textdatei wurde getestet, um sicherzustellen, dass Aufgaben beim Neustart vollständig wiederhergestellt werden.
- Update (Bearbeiten/Status ändern): Änderungen an Beschreibung, Status oder Priorität wurden getestet, inklusive erneuter Speicherung.
- Delete (Löschen): Einzelne Aufgaben wurden gelöscht, und es wurde kontrolliert, ob die Datei danach ohne Reste neu geschrieben wird.

Zusätzlich wurden die Filter und Sortierfunktionen überprüft, da sie die Anzeige beeinflussen, ohne die Datei zu verändern. Für jede Funktion wurden korrekte sowie fehlerhafte Eingaben getestet, damit das Verhalten auch bei ungültigen Werten stabil bleibt.

5.2 Probleme & Lösungen

Ein häufiges Problem war die Platzierung von verschachtelten While Schleifen. Das Programm sprang teilweise in das falsche Menü zurück. Die Lösung bestand darin, die Menülogik klar zu trennen und Rückgabewerte konsequent auszuwerten.

Weitere typische Fehler waren fehlende Klammern oder falsche Variablenreferenzen. Durch intensives Testen und schrittweises Debugging konnten diese Probleme behoben werden.

6. Fazit

Die Umsetzung des Aufgabenlisten Managers war lehrreich. Besonders die Modularisierung, die Benutzerführung über die Konsole und die dauerhafte Speicherung brauchten Zeit und viel Ausprobieren. Das Projekt zeigt, wie wichtig saubere Struktur, Fehlerbehandlung und Testen sind.