

**Ist es sinnvoll, für mein Projekt einen venv in "G:\PythonEnv" zu erstellen**

Ja, das ist eine **sehr gute Idee**! 🎉 Basierend auf den Best Practices aus den Suchergebnissen ist eine zentrale venv-Verwaltung definitiv empfehlenswert.

**🎯 Empfohlene Struktur für dein Setup:**

G:\PythonEnv\  
├── glxy\_acrobat\_pro\ # Dein venv für dieses Projekt  
├── projekt2\ # Weitere venvs für andere Projekte   
└── projekt3\  
  
G:\GLXY\_ACROBAT\_PRO\ # Dein Projekt-Code (getrennt!)  
├── src\  
├── requirements.txt  
└── ...

**📋 Konkrete Schritte:**

**1. venv in G:\PythonEnv erstellen:**

cd G:\PythonEnv  
python -m venv glxy\_acrobat\_pro

**2. venv aktivieren:**

G:\PythonEnv\glxy\_acrobat\_pro\Scripts\Activate.ps1

**3. Zum Projekt wechseln:**

cd G:\GLXY\_ACROBAT\_PRO

**4. Abhängigkeiten installieren:**

pip install customtkinter pandas pyautogui keyboard

**🚀 Vorteile dieser Struktur:**

**✅ Zentrale Verwaltung**

* Alle venvs an einem Ort
* Einfaches Löschen alter Umgebungen
* IDE kann alle venvs automatisch erkennen

**✅ Projekt-Trennung**

* Projekt-Code bleibt sauber
* Keine venv-Dateien im Git-Repository
* Einfache Übertragung des Projekts

**✅ VS Code Integration**

// .vscode/settings.json  
{  
 "python.pythonPath": "G:\\PythonEnv\\glxy\_acrobat\_pro\\Scripts\\python.exe"  
}

**🔧 Workflow für die Zukunft:**

# 1. venv aktivieren  
G:\PythonEnv\glxy\_acrobat\_pro\Scripts\Activate.ps1  
  
# 2. Zum Projekt wechseln   
cd G:\GLXY\_ACROBAT\_PRO  
  
# 3. Entwickeln  
python src/main.py

**💡 Pro-Tipp:**

Erstelle ein **PowerShell-Script** für den schnellen Start:

**start\_project.ps1:**

G:\PythonEnv\glxy\_acrobat\_pro\Scripts\Activate.ps1  
cd G:\GLXY\_ACROBAT\_PRO  
Write-Host "🚀 Umgebung aktiviert! Projekt bereit." -ForegroundColor Green

Diese Struktur folgt den **Industry Best Practices** und macht dein Setup professionell und wartbar! 🎯

⁂