STEP 1 : SETTING UP THE GITHUB REPOSITORY AND REQUIREMENTS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Description** | **Importance** | **If Skipped** | **Alternatives** | **Problem Solved** |
| 1. Setting up GitHub Repository | Create new repo on GitHub | Enables version control and collaboration | Difficult to track changes and collaborate | GitLab, Bitbucket | Organizes code in distributed environment |
| 2. Creating Local Project Directory | Create folder on local machine | Organizes project files | Files may be scattered | Using existing directory | Provides structured local environment |
| 3. Initializing Git Repository | Use 'git init' command | Enables local version control | Cannot track changes or sync with remote | None for Git-based control | Prepares directory for version control |
| 4. Creating README.md | Create README with project description | Provides project overview | Lack of initial documentation | Other doc formats | Offers quick project introduction |
| 5. Adding and Committing README.md | Use 'git add' and 'git commit' | Stages and commits file locally | Changes won't be tracked | None within Git workflow | Tracks changes, prepares for remote syncing |
| 6. Connecting Local and Remote Repos | Use 'git remote add origin [URL]' | Links local to GitHub repo | Cannot push/pull from GitHub | Manual file upload | Establishes local-remote connection |
| 7. Pushing to GitHub | Use 'git push -u origin main' | Uploads local commits to GitHub | Local changes won't appear on GitHub | None for initial push | Syncs local and remote repositories |
| 8. Creating .gitignore File | Create .gitignore with Python template | Specifies files to ignore | May commit unnecessary files | Manual specification | Keeps repository clean and secure |
| 9. Setting Up Python Environment | Create new Conda environment | Isolates project dependencies | Potential conflicts with system packages | virtualenv, venv | Creates isolated development environment |
| 10. Creating setup.py | Create setup.py with project metadata | Defines project as a package | Cannot build distributable package | Using only requirements.txt | Enables project packaging |
| 11. Creating requirements.txt | Create file with project dependencies | Lists all required packages | Difficulty reproducing environment | Specifying in setup.py only | Simplifies dependency management |
| 12. Implementing get\_requirements Function | Create function to read requirements | Dynamically reads requirements | Manual updates to setup.py needed | Hardcoding in setup.py | Ensures consistency in dependencies |
| 13. Creating Source Directory | Create 'src' with **init**.py | Establishes project structure | Difficulty importing project modules | Flat project structure | Enables proper Python packaging |
| 14. Installing Project in Editable Mode | Use 'pip install -e .' command | Allows development without reinstalling | Changes not immediately reflected | Regular installation | Facilitates easier development and testing |
| 15. Committing and Pushing Changes | Add, commit, push new files to GitHub | Updates remote repo with changes | Remote repo out of sync | None within Git workflow | Keeps remote updated, enables collaboration |