How to... Python Project

Introduction

When starting a new Python project, I follow a consistent setup workflow. This approach ensures a fast, clean, and professional start every time.

> Note: Even if some steps (like creating certain subdirectories) aren't immediately necessary, I perform them upfront. It's better to be ready than to interrupt a session later to add structure.

The 7-Step Workflow

This document details the seven steps I follow to set up any new Python project.

1. Create the Project Folder
2. Initialize a Git Repository
3. Create Essential Subdirectories
4. Create Necessary Files
5. Create and Activate a Python Virtual Environment
6. Configure the `.gitignore` File
7. Add the Initial Commit to Git

1. Create the Project Folder

First, we create an empty directory to serve as our project's root.

```
mkdir my_project
```

Remember to navigate into your new project folder:

```
cd my_project
```

2. Initialize a Git Repository

To enable version control and code tracking, we initialize a new Git repository within the project folder.

```
git init
```

3. Create Essential Subdirectories

I always create the subdirectories src, data, and img, even if I don't need all of them at the start.

`src`: Stores all source code files (the heart of the project). `data`: Intended for project-specific datasets and external data. `img`: For images, figures, and plots generated or used by the project.

```
mkdir src data img
```

4. Create Necessary Files

A standard Python project requires a few essential configuration and documentation files:

- `README.md`: A Markdown file containing an overview, installation, and usage instructions for the project.
- `requirements.txt`: A text file listing all necessary Python package dependencies.
- `.env`: A special file for storing environment-specific variables, such as API keys or configuration settings (it should always be ignored by Git).
- `.gitignore`: A file listing all files and folders that Git should not track (e.g., virtual environments, large data files, and sensitive info).

```
touch README.md requirements.txt .env .gitignore
```

5. Create and Activate a Python Virtual Environment

To prevent package dependency conflicts with other projects or the system Python installation, it's crucial to use a new virtual environment (venv) for every project.

```
python3 -m venv .venv
```

> Note: This creates a folder named .venv containing the isolated Python environment

You must activate the environment before installing packages or running your code.

```
# For Mac/Linux
. .venv/bin/activate

# For Windowns (PowerShell)
.venv\Scripts\Activate.ps1

# For Windowns (CMD)
.\.venv\Scripts\activate.bat
```

> Note: Check your venv setup with `pip list`: there should be no installations listed except `pip`.

6. Configure the `.gitignore` File

The `.gitignore` file tells Git which files and folders to ignore. Git isn't designed to track large binary files (like datasets or images) or sensitive data (like passwords).

> Note: Also ignore the `.env` and `.venv` files.

```
echo ".venv" >> .gitignore
echo ".env" >> .gitignore
echo "data/" >> .gitignore
echo "img/" >> .gitignore
```

7. Add the Initial Commit to Git

Finally, we stage all the newly created structural files and make the project's first official commit.

```
git add -A
git commit -m "Initial project setup"
```

You are now ready to start coding your Python project!