

Python Project Setup Instructions for Windows

Welcome! This guide will help you create a clean project folder with a virtual environment and install packages from a requirements.txt file.

Before we begin - Heads up!

If you are relatively new to using the Command Prompt, here is an excellent guide with instructions and exercises. I'd suggest you work through this first to build some confidence.

There are 4 common gotchas to developing on Windows to be aware of:

1. **Permissions** - Please take a look at tutorials on permissions on Windows
2. **Anti-virus, Firewall, VPN** - These can interfere with installations and network access; try temporarily disabling them as needed
3. **The evil Windows 260 character limit** to filenames - consider enabling long path support
4. If you've not worked with Data Science packages on your computer before, you might need to install **Microsoft Build Tools**

Part 1: Create Your Project Directory

1. Open Command Prompt:

- Press `Win + R`, type `cmd`, and press Enter

2. Navigate to your desired location:

If you have a specific folder for projects, navigate to it using the `cd` command. For example:

```
cd C:\Users\YourUsername\Documents\Projects
```

Replace `YourUsername` with your actual Windows username. If you don't have a projects folder, you can create one:

```
mkdir C:\Users\YourUsername\Documents\Projects
cd C:\Users\YourUsername\Documents\Projects
```

3. Create your new project folder:

```
mkdir my_project
cd my_project
```

Replace `my_project` with your desired project name.

Part 2: Set Up Python Virtual Environment

1. Check your Python version:

```
python --version
```

Ideally you should be using Python 3.11 or 3.12. Python 3.13 may not yet work with all Data Science dependencies as of February 2025. If you need to install Python or install another version, you can download it here: <https://www.python.org/downloads/>

2. **Create a virtual environment:** From within your project directory, run:

```
python -m venv llms
```

3. **Activate the virtual environment:**

```
llms\Scripts\activate
```

You should see `(llms)` in your command prompt, which indicates your virtual environment is active.

Part 3: Install Packages from requirements.txt

1. **Upgrade pip (recommended):**

```
python -m pip install --upgrade pip
```

2. **Place the provided requirements.txt file:**

- Place the requirements.txt file that was provided to you in your project root directory
- Make sure it's in the same folder where you created your virtual environment

3. **Install all required packages:**

```
pip install -r requirements.txt
```

This may take a few minutes to install all the necessary packages. **If you see an error about Microsoft Visual C++:**

Microsoft Visual C++ 14.0 or greater is required. Get it with
"Microsoft C++ Build Tools": <https://visualstudio.microsoft.com/visual-cpp-build-tools/>

Please follow the link and install Microsoft C++ Build Tools. **If installation fails, try the bullet-proof version:**

```
pip install --retries 5 --timeout 15 --no-cache-dir --force-reinstall -r requirements.txt
```

Part 4: Start Jupyter Lab

Jupyter Lab should be included in your requirements.txt file. To start it:

1. **Make sure your virtual environment is active** (you should see `(llms)` in your prompt)
2. **Start Jupyter Lab:**

Jupyter Lab should open up in your browser, ready for you to create and run notebooks.

Part 5: Set Up API Keys and Environment Variables (Optional)

API Key Setup

You may want to set up API keys for various AI services:

For Cloud APIs:

- **OpenAI:** Create account at <https://platform.openai.com/>, add minimum \$5 credit, create API key
- **Anthropic (Claude):** Get API key at <https://console.anthropic.com/>
- **Google (Gemini):** Get API key at <https://ai.google.dev/gemini-api>
- **HuggingFace:** Free account at <https://huggingface.co>, create token in Avatar menu » Settings » Access Tokens

For Local AI with Ollama:

- **Install Ollama:** Download from <https://ollama.ai/> and follow installation instructions
- **No API key needed** - Ollama runs locally and doesn't require authentication
- **Default endpoint:** `http://localhost:11434` (automatically used by most libraries)
- **Test installation:** Open Command Prompt and run `ollama --version`
- **Pull a model:** `ollama pull llama2` or `ollama pull mistral` to get started

Environment Variables Setup

1. Create a .env file:

- Open Notepad (`Windows + R`, type `notepad`)
- Add your environment variables. Here's a complete example:

```
# Cloud API Keys
OPENAI_API_KEY=sk-proj-your_openai_key_here
ANTHROPIC_API_KEY=sk-ant-your_anthropic_key_here
GOOGLE_API_KEY=your_google_key_here
DEEPSEEK_API_KEY=your_deepseek_key_here
HF_TOKEN=your_huggingface_token_here

# Ollama Configuration (Local AI)
OLLAMA_BASE_URL=http://localhost:11434
OLLAMA_MODEL=llama2

# Other useful environment variables
WANDB_API_KEY=your_wandb_key_here
```

- **Important:** Double check there are no spaces before or after the = sign, and no spaces at the end of the keys
- Go to File > Save As
- In "Save as type" dropdown, select "All Files"
- In "File name" field, type exactly `.env`
- Save in your project root directory

2. Install python-dotenv to load environment variables:

```
pip install python-dotenv
```

3. Using environment variables in your code:

```
python

import os
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()

# Access your API keys
openai_key = os.getenv('OPENAI_API_KEY')
ollama_url = os.getenv('OLLAMA_BASE_URL', 'http://localhost:11434') # Default fallback
```

Using Ollama

Once Ollama is installed and running:

1. Check available models:

```
ollama list
```

2. Pull popular models:

```
ollama pull llama2  
ollama pull mistral  
ollama pull codellama
```

3. Test a model:

```
ollama run llama2
```

4. In your Python code, you can use libraries like:

- `ollama` - Official Python client
- `langchain` - With OllamaLLM
- `openai` - Using OpenAI-compatible endpoint

Note: Ollama runs as a local service, so no internet connection is required once models are downloaded, and your data stays completely private on your machine.

Starting Your Project in the Future

Every time you want to work on this project:

1. Open Command Prompt

2. Navigate to your project directory:

```
cd C:\Users\YourUsername\Documents\Projects\my_project
```

3. Activate your virtual environment:

```
llms\Scripts\activate
```

You should see `(llms)` in your prompt.

4. **Start working!** You can now run Python scripts, start Jupyter Lab, or install additional packages.

Deactivating the Virtual Environment

When you're done working on your project, you can deactivate the virtual environment:

```
deactivate
```

Adding More Packages Later (If Needed)

With your virtual environment activated, you can install additional packages if needed:

```
pip install package_name
```

If you add packages and want to update the requirements file for others:

```
pip freeze > requirements-updated.txt
```

Troubleshooting

- If you encounter permission errors, try running Command Prompt as Administrator
- If antivirus software is blocking installations, temporarily disable it
- For path length issues, consider enabling long path support in Windows
- If you have installation problems, feel free to reach out for help!

Your project structure should now look like this:

```
my_project/
├── llms/          # Virtual environment folder
├── requirements.txt # Package dependencies
├── .env           # Environment variables (optional)
└── [your project files] # Your Python scripts, notebooks, etc.
```