Python Project Setup Instructions for Mac

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

If you're less familiar with using the Terminal, please review guides for some details and exercises.

If you're new to developing on your Mac, you may need to install XCode developer tools. This is usually prompted automatically when you first try to use git or python development tools.

One "gotcha" to keep in mind: if you run anti-virus software, VPN or a Firewall, it might interfere with installations or network access. Please temporarily disable if you have problems.

Part 1: Create Your Project Directory

1. **Check Git installation:**
   - Open Terminal (Applications > Utilities > Terminal)
   - Type `git --version`. If not installed, you'll be prompted to install it
2. **Navigate to your desired location:** If you have a specific folder for projects, navigate to it using the cd command. For example:

   ```
   cd ~/Documents/Projects
   ```

   If you don't have a projects folder, you can create one:

   ```
   mkdir ~/Documents/Projects
   cd ~/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
   ```

   or

   ```
   python3 --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
```

or if you need to use python3:

```
python3 -m venv llms
```

3. **Activate the virtual environment:**

```
source llms/bin/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 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
```

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 Terminal 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 using nano:**
   - Open Terminal and navigate to your project root directory
   - Create the .env file with: `nano .env`
   - 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

- Save the file: `Control + O`, then `Enter` (to confirm), then `Control + X` to exit

2. **Verify the .env file was created:**

```
ls -a
```

You should see the .env file listed.

3. **Install python-dotenv to load environment variables:**

```
pip install python-dotenv
```

4. **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 Terminal** (Applications > Utilities > Terminal)
2. **Navigate to your project directory:**

```
cd ~/Documents/Projects/my_project
```

3. **Activate your virtual environment:**

```
source llms/bin/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, you may need to use `sudo` for certain installations
- If antivirus software is blocking installations, temporarily disable it
- Make sure XCode developer tools are installed if you encounter build errors
- If you have installation problems, check that you're using the correct Python version

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.
```

For those new to Jupyter Lab / Jupyter Notebook, it's a delightful Data Science environment where you can simply hit shift+return in any cell to run it; start at the top and work your way down!