Python for Linguistic Data Analysis

with VS Code and LLMs

Workshop

Table of contents

Setup Python and VS Code

Operating System

Choose based on what you have:

- **Windows**: PowerShell or Windows Terminal (recommended)
- **macOS**: Built-in Terminal or iTerm2
- **Linux**: Native terminal (GNOME, Konsole, etc.)

Most common in linguistics labs is Windows. WSL2 is advanced option. Quick setup: use defaults on your system.

Terminal/Shell

Where you run Python commands:

- **PowerShell** (Windows) – Modern, cross-platform, recommended
- **Command Prompt** (Windows) – Traditional, sufficient
- **Bash** (macOS/Linux/WSL) – Standard Unix
- **Zsh** (macOS) – Bash-compatible with nice defaults

Recommendation: Use whatever came with your OS. PowerShell on Windows.

Install VS Code + Extensions

Reference extensions for Python development:

1. Download Visual Studio Code
2. Install from marketplace:

    - **Python** (Microsoft) – Core language support
    - **Jupyter** (Microsoft) – Notebook support
    - **Pylance** (Microsoft) – Type checking & intellisense

Install Python

- **Required**: Python 3.9+
- Download from python.org
- macOS: `brew install python3`
- Linux: `apt install python3`
- **Optional**: Use pyenv for multiple versions

Verify Installation

Open terminal and run:

```
python --version    # Should show Python 3.9+
pip --version       # Package manager version
```

Clone Repository & Git

These slides as well as two datasets are in a public repo called 'pyws' on my GitHub Account 'jobschepens'. You can clone these files using the command line:

```
git clone https://github.com/jobschepens/pyws.git
cd pyws
```

Before cloning, make sure `git` is installed on your system. Open your terminal and navigate to the directory where you want to store the repository (e.g., `C:\GitHub` on Windows or `~/projects` on macOS/Linux). You can also use GitHub Desktop if you prefer a graphical interface.

For today's workflow, you can work directly in the cloned repository: create new subfolders for your analyses, then commit and push changes to the `main` branch. For collaborative or more advanced workflows, consider forking the repository on GitHub, cloning your fork, creating a feature branch for your work, committing and pushing your changes, and then opening a Pull Request to merge your work into the original repository. This approach is recommended for larger teams or when contributing to shared projects.

Create Virtual Environment

Isolate project dependencies (venv recommended; conda/poetry/pipenv are alternatives):

```
# Create
python -m venv .venv

# Activate
# Windows (PowerShell):
.venv\Scripts\Activate.ps1
# Windows (Command Prompt):
.venv\Scripts\activate
# macOS/Linux:
source .venv/bin/activate
```

Install Packages

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

Installs: **pandas** (data), **numpy** (computing), **matplotlib** (visualization), **jupyter** (notebooks). Note that these packages are relatively large (often 100–200 MB total for all dependencies).

Verify in VS Code

1. Open VS Code, open `pyws` folder
2. Terminal: 'Ctrl+" (backtick) → Activate venv

3. Run verification:

```
python --version
pip list
```

LLM Configuration & Setup

Choose a Provider

Pick one or more (free tiers available):

Provider	Note	Cost
GWDG CoCo AI	Recommended for academic use	Free
OpenAI/Claude/Gemini	"Top tier" models	Monthly payments
OpenRouter	Many models, unified API	Pay-per-use
GitHub Copilot Pro	Limited top-tier models	Free with educational account
Ollama	Local models, no internet	Free
Groq	Fast inference	Free tier available

What is CoCo AI?

CoCo AI: Code completion service via Chat AI on GWDG's AcademicCloud

- **Integrated LLMs**: Llama, Codestral, Qwen models
- **No local setup needed**: Remote via API
- **Access control**: SAIA API key (institution login)
- **Ideal for**: Research institutions with GWDG access

Get API Credentials

- **Most providers**: Generate API key in account settings
- **GitHub Copilot**: Use GitHub login (no separate key)
- **Store safely**: Use environment variables, never commit to git

```
# Example: .env file (add to .gitignore!)
SAIA_API_KEY=...
OPENAI_API_KEY=sk-...
ANTHROPIC_API_KEY=sk-ant-...
```

Install VS Code Extension

Popular options for different workflows:

- **GitHub Copilot Chat**
- **Continue**
- **Cline**
- **Roo Code**

Configure & Test

1. Enter API key or GitHub login
2. Select model (GPT-4, Claude 3.5, Gemini, Llama)
3. Test with: `"Write a Python function to read a CSV file"`

Optional MCP Servers

Advanced: Model Context Protocol for complex workflows. Not required for basic setup.

Cloud Alternatives: Overview

Popular cloud-based alternatives to local setup:

Platform	Best for	Setup	Limits	Use in Workshop
Google Colab	Quick prototyping	Sign in with Google	Resets after 12h	Good for demos
Binder	Sharing reproducible environments	GitHub repo → mybinder.org	Public repo only	Good for homework
GitHub Codespaces	Full IDE in browser	Click "Code" on GitHub	60h/month free	Best backup

General CLI vs. Extension Philosophy

Core toolkit (CLI): `pip`, `git`, `gcloud`, `docker`, `poetry`, `nox`

Aspect	CLI Approach	VS Code Extension Approach
Reproducibility	Best (scripts/config in version control)	Good (`.vscode/` settings + extensions list)
Automation	Best (native scripting in bash/PowerShell)	Limited (requires extension APIs)
Accessibility	Works everywhere (SSH, containers, servers)	VS Code only
Discoverability	Steeper learning curve	Visual feedback; easier exploration
Team Collaboration	Best (commands are portable, language-agnostic)	Good (share `.vscode/settings.json`)

CLI Workflow

```
# 1. Set up reproducible environment (in version control)
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt

# 2. Run batch processing / CI validation
cline-agent --task "refactor test suite" --approve-all

# 3. Final validation & deployment
nox -s test
nox -s lint
gcloud deploy app.yaml
```