Python Installation Guide (WSL2 on Windows Visual Studio Code)

Unit: Supply Chain Modelling and Optimisation

Version: 2025

Why Python, Jupyter, and VS Code?

Python is an interpreted, general-purpose, object-oriented, high-level programming language with dynamic semantics. It has a strong position in scientific computing with a large community of users and available documentation. With an extensive ecosystem of scientific libraries and environments such as *Numpy*, *Scipy*, and *Matplotlib*, it allows user to work on various scientific topics, especially computer science, mathematics, statistics, and optimisation.

Jupyter provides the web-based notebook application, which allows user to write, run, display, and document the output produced by the code. This means that we are able to capture the entire workflow in a single file, which can be saved, restored, and reused later on.

With Python and Jupyter extensions installed on VS Code, we can create Jupyter notebook, interactive programming and computing that supports IntelliSense, debugging and more. The Python extension generally supports code completion and IntelliSense using the currently selected interpreter. IntelliSense is a general term for features, including code completion, parameter info, quick info, and member lists, which are very useful for developing Python applications.

Why WSL2?

WSL2 (Windows Subsystem for Linux version 2) provides a full Linux kernel running directly on Windows, offering native Linux performance, compatibility, and system call support. It allows users to run a genuine Linux environment without using a virtual machine or dual boot. For developers working with Python and Jupyter, WSL2 ensures that Linux-based tools, packages, and dependencies work as expected, eliminating compatibility issues that often occur on Windows. It also enables seamless integration with VS Code through the *Remote — WSL* extension, allowing users to write and execute code inside Linux while using the familiar Windows interface. This approach combines the speed and flexibility of Linux with the accessibility of Windows, making it ideal for scientific computing, data analysis, and reproducible research workflows.

Quick Installation

1. Enable WSL2 and install Ubuntu
   Open PowerShell as Administrator, run the following commands:

   ```
   wsl --install -d Ubuntu
   wsl --set-default-version 2
   wsl --update
   ```

   Reboot if prompted, then launch Ubuntu from Start Menu and create your Linux user.

2. Update Ubuntu and other essentials

   ```
   sudo apt update && sudo apt upgrade -y
   sudo apt install -y build-essential curl git zip unzip
   ```

3. Install Python (recommended: `pyenv` for clean version control)

```
# deps for building Python
sudo apt install -y make libssl-dev zlib1g-dev \
  libbz2-dev libreadline-dev libsqlite3-dev wget llvm \
  libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev \
  libffi-dev liblzma-dev

# install pyenv
curl https://pyenv.run | bash

# add to shell (Ubuntu default is bash; if you use zsh, update ~/.zshrc)
echo -e '\n# pyenv\nexport PATH="$HOME/.pyenv/bin:$PATH"\n' >> ~/.bashrc
echo 'eval "$(pyenv init -)"' >> ~/.bashrc
echo 'eval "$(pyenv virtualenv-init -)"' >> ~/.bashrc
exec $SHELL

# install & select a Python version
pyenv install 3.12.6
pyenv global 3.12.6
python -V
```

Successful installation should return Python version.
Alternatively, you can use Conda instead of `pyenv` (optional). If you prefer Anaconda/Miniconda:

```
# Miniconda (lighter)
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.
   sh -O ~/miniconda.sh
bash ~/miniconda.sh -b -p $HOME/miniconda
echo -e '\nexport PATH="$HOME/miniconda/bin:$PATH"\n' >> ~/.bashrc
exec $SHELL

conda create -n myproj python=3.12 -y
conda activate myproj
pip install ipykernel jupyter
python -m ipykernel install --user --name myproj
```

4. Create a project folder in WSL (not on C:)

```
mkdir -p ~/code/myproj && cd ~/code/myproj
python -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
pip install ipykernel jupyter pandas numpy matplotlib
python -m ipykernel install --user --name myproj
```

Alternatively, you can clone GitHub repo inside WSL. Check the next section for more details.

5. Download and install the latest version of Visual Studio Code. If VS Code is previously installed, run the following command on CMD/Windows PowerShell to upgrade VS Code version

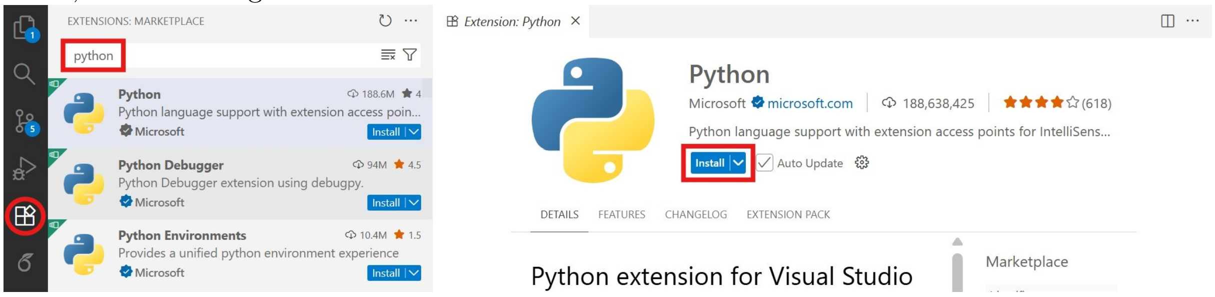
```
winget upgrade --id Microsoft.VisualStudioCode
```

6. In VS Code, install extensions:

   - Remote – WSL
   - Python
   - Jupyter

```
code --install-extension ms-vscode-remote.remote-wsl
code --install-extension ms-python.python
code --install-extension ms-toolsai.jupyter
```

Alternatively, this can be done by locating the **Extensions** tab, searching for the extension by name, and installing it.


EXTENSIONS: MARKETPLACE
python
Python
188.6M 4
Python language support with extension access poin...
Microsoft
Install
Python Debugger
94M 4.5
Python Debugger extension using debugpy.
Microsoft
Install
Python Environments
10.4M 1.5
Provides a unified python environment experience
Microsoft
Install
Extension: Python
Python
Microsoft microsoft.com 188,638,425 (618)
Python language support with extension access points for IntelliSens...
Install
Auto Update
DETAILS
FEATURES
CHANGELOG
EXTENSION PACK
Python extension for Visual Studio
Marketplace

7. Open the WSL folder in VS Code (Remote – WSL)
   - Press **F1** → **"WSL: Connect to WSL"**.
   - Then **File** → **Open Folder...** and pick `\\wsl\textdollar\Ubuntu\home\<user>\code\myproj` (or use the "Open Folder in WSL. . . " button).
   - VS Code will auto-install its server into WSL.

8. Select your interpreter / kernel
   Bottom-right status bar → "Python: Select Interpreter" → choose `~/code/myproj/.venv/bin/python`. (You may have to open or create a .ipynb notebook first in order to see the status bar). To create one: Press **F1** → **"Create: New Jupyter Notebook"**.

9. In the top-right of the notebook, set **Kernel** = **myproj** (or your venv interpreter).

You're ready—run a cell and verify.

Git Integration

Why We Use Git (and GitHub)? Git is a version control system – it helps you track changes in your code over time, collaborate with others, and recover older versions if something breaks. Think of it like a "time machine + collaboration tool" for your projects. It's especially valuable for:

- **Tracking changes:** Every edit you make is saved as a commit with a message (like a checkpoint).
- **Experimenting safely:** You can create branches to test new ideas without touching the main code.
- **Undoing mistakes:** Easily revert to a previous version.
- **Working in teams:** Multiple people can edit files simultaneously – Git merges everyone's work.

Why It's Useful in VS Code + WSL? When you clone your repo in WSL2, you're keeping the Linux environment (great for Python, Jupyter, etc.) while still using VS Code's GUI. Git lets you:

- **Pull** updates from teammates
- **Commit** and **Push** your own changes
- **Sync** your local work with GitHub in one click

You'll see all this right inside VS Code's **Source Control** tab (the branch icon on the left).

Clone your GitHub repo inside WSL

1. Make sure `git` is installed

   ```
   sudo apt install -y git
   ```

2. Authenticate with GitHub
   The easiest way is SSH, so check if you already have a key:

   ```
   ls ~/.ssh
   ```

   If you don't see `id_rsa.pub` or `id_ed25519.pub`, create one:

   ```
   ssh-keygen -t ed25519 -C "your_email@example.com"
   ```

   If prompt to "Enter file", press Enter to use the default location. If prompt to "Enter passphrase", press Enter again for no passphrase (simpler, but less secure), or set a passphrase for extra protection (you'll need to enter it each time unless you use `ssh-agent`). Then, copy the key:

   ```
   cat ~/.ssh/id_ed25519.pub | clip
   ```

   Add that public key to your GitHub account: → GitHub → Settings → SSH and GPG keys → New SSH key. Now, test it:

   ```
   ssh -T git@github.com
   ```

   If this is the first time you connect to GitHub via SSH from a new system, there might be a message warning you that it can't verify that this server is truly GitHub, and **"Are you sure you want to continue connecting (yes/no/[fingerprint])?"**. Type **yes** and press Enter. SSH will print **"Hi <username>! You've successfully authenticated, but GitHub does not provide shell access."**, which means your SSH key is working perfectly.

3. Clone your repo

   ```
   mkdir -p ~/code
   cd ~/code
   git clone git@github.com:YourUsername/your-repo.git
   # Or use HTTPS if you havent set up SSH keys yet:
   git clone https://github.com/YourUsername/your-repo.git
   cd your-repo
   ```