

Data Science & AI for Economists

Lecture 6: Introduction to Python in VS Code

Zhaopeng Qu

Business School, Nanjing University

October 15 2025



Roadmap

Today's Agenda

1. Installing Basic Python

- Installing Python
- Installing VS Code and Python Extensions

2. Basic Python in Practice in VS Code:

- Manage Environment and Libraries
- Using VS Code/Jupyter Notebook/Lab with Copilot to Coding

Installing Basic Python

Installing Python

Three Ways to Install Python

Option 1: Direct Installation(*not recommended*)

- From Python.org
- Minimal setup
- Full control

Option 2: Anaconda Distribution(*recommended for beginners*)

- All-in-one solution
- Pre-configured for data science
- **Recommended for this course**

- **Remember:** Whatever you pick, double-check that your system **PATH** points to the new Python so the terminal and VS Code can find it.

Option 3: Miniforge(*recommended for intermediate users*)

- Lightweight alternative
- Community-driven
- For advanced users

Direct install from python.org

- Visit python.org/downloads and grab the latest stable release (e.g., Python 3.12).
- Windows notes:
 - Tick `Add python.exe to PATH` during setup, or add `%LocalAppData%\Programs\Python\Python312\` manually later.
 - Use Customize installation to ensure `pip` is included.
- MacOS notes:
 - The `.pkg` installer places Python under `/Library/Frameworks/Python.framework`.
 - Add `/Library/Frameworks/Python.framework/Versions/3.12/bin` to your shell profile if `python3 --version` fails.
- Verify with:

```
python3 --version
pip3 --version
```

Installing Python via Anaconda

- Download the Anaconda Individual Edition from anaconda.com.
- Run the installer (≈3 GB disk space). Allow it to update your PATH when prompted (recommended for beginners).
- Launch *Anaconda Navigator* for a GUI to manage packages, environments, Jupyter, and VS Code launchers.
- Confirm the install:

```
conda --version  
python --version
```

- If the commands fail, add the install path (e.g., `C:\Users\<user>\anaconda3\Scripts`) to PATH manually.

Installing Python via Miniforge

- **Miniforge** is a minimal conda-forge-first distribution—ideal when you want conda environments without the full Anaconda bundle.
- Download the installer that matches your OS / architecture from [conda-forge/miniforge](#).
- Installation footprint is small (~400 MB) and uses the **conda-forge** channel by default; the *Mambaforge* variant ships with **mamba** pre-installed.
- PATH considerations:
 - macOS/Linux: the installer updates your shell profile; reopen the terminal or `source ~/.zshrc`.
 - Windows: check *Add Miniforge3 to my PATH environment variable* or add `C:\Users\
<user>\miniforge3\Scripts` manually.
- Validate in your terminal:

```
conda --version
```

```
mamba --version    # if you chose Mambaforge
```


Manage Environment and Libraries

- Since **Python** is a free and open-source programming language, there are many package managers and environment management systems for Python.
 - **Package manager** is a tool that helps you install, update, and remove packages for your programming language.
 - **Environment management system** is a tool that helps you create, activate, and manage your virtual development environments.
- If you want to replicate someone else's project, you have to make sure that you have the same version of python and the same packages.

Manage Environment and Libraries

1. official package manager and environment management system

- `pip` is official package manager and `venv` is official environment management system for Python.

2. third-party package manager and environment management system

- `conda` is a package manager and environment management system for Python, R and many other languages for data science by `Anaconda` company(sort of like RStudio for R).
- `conda-forge` is a third-party package manager and environment management system for Python by `conda-forge` community.
- `mamba` is a faster alternative to `conda` by `conda-forge` community.

Basic Workflow in Python

Basic Workflow in Python

Basic Workflow

1. Create or import a virtual development environment
2. Activate the environment
3. Install or import necessary packages
4. Launch IDEs (such as Jupyter Lab or VS Code)
5. Select the environment and interpreter within the IDE
6. Using Copilot to help coding

Why we need VDE?

- Collaborative Coding Challenges:
 - Team members may have different versions of Python installed.
 - Even if everyone uses the same version of Python, they might have different versions of packages and dependencies.
 - You yourself may need different versions of packages or dependencies for different projects.
- A **Virtual Development Environment (VDE)** is a self-contained, isolated workspace where you can manage and run your Python projects.
 - Using a VDE helps prevent conflicts between dependencies required by different projects, ensuring each project has exactly the right versions it needs.
- Normally, there are two ways to create a VDE:
 - **venv** is official environment management system for Python.
 - **conda** is a package manager and environment management system for Python.

Basic Workflow

Preparing Terminal in Windows

- There are many ways to open a terminal in Windows:
 - CMD
 - PowerShell
 - Git Bash
 - Anaconda Prompt
 - etc.
- If you installed `Anaconda`, you can open `Anaconda Prompt` as your terminal.
- Otherwise, you can try `Git Bash` as your best terminal to use shell commands.

Basic Workflow

Creating a Virtual Development Environment (VDE)

- Open Anaconda Prompt for Windows or Terminal for Mac/Linux.
- Verify the installation of conda and Python:

```
conda --version  
conda info  
python --version
```

- See existing environments:

```
conda env list
```

- **Remember:**
 - You should always create a new VDE for your projects and never use the base environment.

Basic Workflow

Creating a VDE

- Open Anaconda Prompt for Windows or Terminal for Mac/Linux.

- Create a new VDE:

```
conda create -name myenv python=3.12.4
```

- Verify again environment:

```
conda env list
```

Activate an VDE

- Activate an VDE:

```
conda activate myenv
```

- Verify the packages in the environment:

```
conda list
```


Basic Workflow

Install necessary packages

- Install necessary packages:

```
conda install numpy pandas jupyterlab dask h
```

- If you want to update a package, you can use:

```
conda update numpy
```

- If you want to remove a package, you can use:

```
conda remove numpy
```

- **Remember:** every installations or updates is in the same environment.

- install all packages from the anaconda channel:

```
conda install anaconda
```

- If you want to install the package from the conda-forge channel which is a third-party channel:

```
conda install -c conda-forge condastats
```

Basic Workflow

Export the environment

1. Export the complete environment (contains all dependencies, the most detailed)

```
conda env export > environment.yml
```

2. Only export the manually installed packages (recommended, the most concise)

```
conda env export --from-history > environment.yml
```

3. Export to requirements.txt format (pip style)

```
conda list --export > requirements.txt
```

4. pip style freeze (only contains pip installed packages)

```
pip freeze > requirements.txt
```

Basic Workflow

Import the environment

- create a new environment from the environment.yml file:

```
# the most common method  
cat environment.yml # check the environment.yml file  
conda env create -f environment.yml  
# activate the environment  
conda activate <environment name>
```

- or update the environment from the `environment.yml` file:

```
conda env update -f environment.yml
```

- or install some packages from the `requirements.txt` file:

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

Basic Workflow

- There are many ways to run or interact with Python code:

1. Shell Command line:

```
python  
print("Hello, World!")
```

2. Script File:

```
python main.py
```

1. Jupyter Notebook/Lab(IDE)

```
jupyter notebook
```

Basic Workflow

Launch IDEs or Editors

- Launch IDEs(such as Jupyter Notebook or VS Code)
 - launch jupyter notebook

```
jupyter notebook
```

- terminate jupyter notebook or **Control + C**

- Deactivate an VDE

```
conda deactivate
```

Basic Workflow

Using Anaconda Navigator to repeat the above steps

- Install Applications you want to use
- Manage Environments
 - Click Create
 - Click Clone
 - Click Backup
 - Click Remove
- Manage necessary packages
- Launch IDEs or Editors (such as Jupyter Notebook or VS Code) within the environment

Python in VS Code

Python in VS Code

Coding Editor Selection

- Recall: VS Code is a Source Code Editor which is a text editor program designed specifically for editing source code of computer programs.
 - Others examples: Sublime Text, Atom, Notepad++, etc.
- Recall: IDE is a software application that provides comprehensive facilities to computer programmers for software development.
 - RStudio, PyCharm, Jupyter Notebook/Lab, etc.
- Although I tell you many people prefer using IDEs for coding, like RStudio for using R and Jupyter Notebook/Lab for using Python, etc.
- I still strongly recommend you to use VS Code for coding for the following reasons:
 - It supports multiple programming languages, including R and Python, even Stata.

Installing VS Code and Python Extensions

- Install the `Python` extension for `VS Code` [here](#)
- Select Python as the default language.
 - Control + Shift + P to open the command palette.
 - Language Mode: Select Python
- Select *interpreter* for Python:
 - Which version of Python you want to use.
 - Which environment you want to use.

Installing Jupyter Notebook/Lab Extensions

- For data science, we usually use Jupyter Notebook/Lab to code for Python.
 - to see the output in the cell rather than the final result in the terminal.
- Install Jupyter Notebook/Lab Extensions for VS Code.
- Select *interpreter* for Python:
- Then the corresponding kernel will be automatically selected.

Practice with VS Code for Python