

Lesson 4 Demo: Prerequisite

Create *pyproject.toml* File to Install Requirements

Objective: To create a *pyproject.toml* file to install all requirements for the project and set up an *.env* file with the required API key

Tools and prerequisites

Editors: VSCode or PyCharm

1. PIPX - the lab should have this installed already (optional if already installed):

pipx is a Python tool that installs and runs Python applications in isolated virtual environments. It simplifies managing standalone Python command-line tools without interfering with your system's Python installation or the dependencies and environments of your other projects.

Installation on macOS:

- Use command: `brew install pipx`
- Run: `pipx ensurepath`

Installation on Windows:

- Use command: `scoop install pipx`
- Run: `pipx ensurepath`

2. Poetry - the lab should have this installed already (optional if already installed):

Poetry is a Python tool for managing dependencies, packaging projects, and handling virtual environments. It simplifies project setup and dependency installation and ensures consistency across different environments using a *pyproject.toml* and *poetry.lock* file. Poetry makes it easy to manage and publish Python packages with minimal effort.

Installation:

- Command: `pipx install poetry`
- Confirm installation with: `poetry --version`

A. Setting up the project

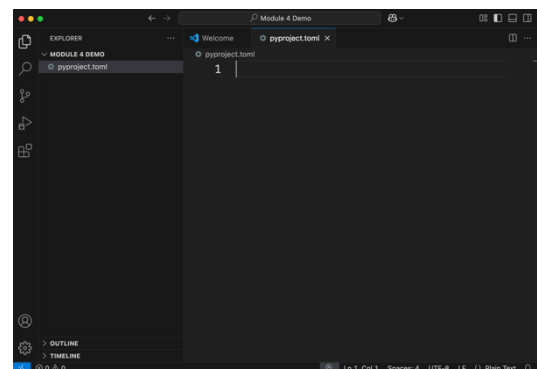
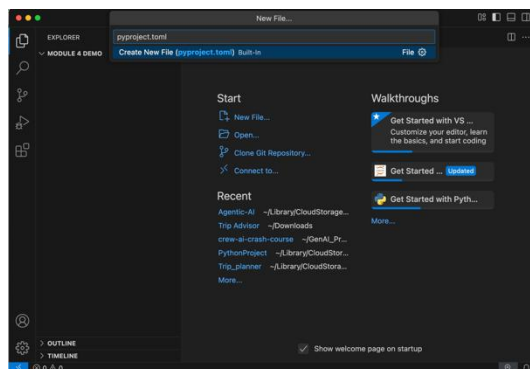
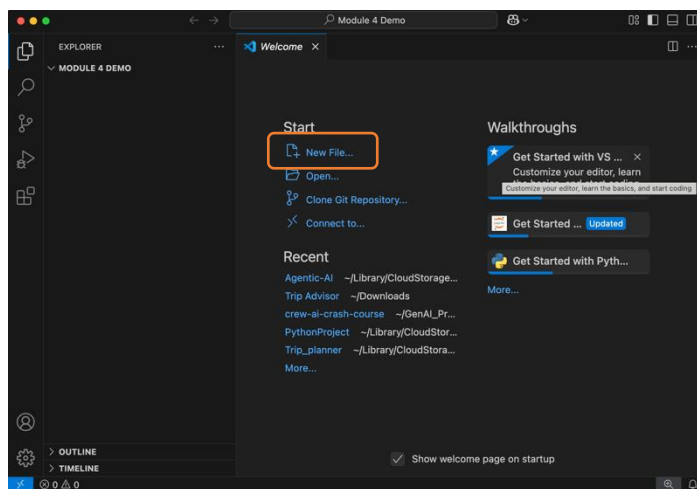
Step 1: Create a new folder named “Module_4_demo”

- 1.1 Open VSCode and create a new folder for your project

B. Creating the pyproject.toml file

Step 1: Create the file

- 1.1 Inside your project folder, create a new file named pyproject.toml



Step 2: Add the project metadata

- 2.1 Insert a `[tool.poetry]` section to define your project's metadata (like name, version, description, and more)

```
Welcome  pyproject.toml
pyproject.toml
1  [tool.poetry]
2  name = "trip-planner"
3  version = "0.1.0"
4  description = "Simple ai trip planner"
5  authors = ["Joao Moura"]
```

Step 3: Specify dependencies

- 3.1 Add a `[tool.poetry.dependencies]` section to list the required tools and their versions

```
6
7  [tool.poetry.dependencies]
8  python = ">=3.10.0,<3.12"
9  crewai = "0.1.24"
10 unstructured = "==0.10.25"
11 pyowm = "3.3.0"
12 python-dotenv = "1.0.0"
13 langchain-openai = "^0.0.5"
14 streamlit = "1.41.1"
```

Step 4: Add any additional sections if needed

- 4.1 Include any extra configuration sections that your project might require

```
16 [tool.pyright]
17 # https://github.com/microsoft/pyright/blob/main/docs/configuration.md
18 useLibraryCodeForTypes = true
19 exclude = [".cache"]
20
21 [tool.ruff]
22 # https://beta.ruff.rs/docs/configuration/
23 select = ['E', 'W', 'F', 'I', 'B', 'C4', 'ARG', 'SIM']
24 ignore = ['W291', 'W292', 'W293']
```

Step 5: Build system setup

- 5.1 Add a `[build-system]` section specifying the build backend and its dependencies

```
26 [build-system]
27 requires = ["poetry-core>=1.0.0"]
28 build-backend = "poetry.core.masonry.api"
```

Step 6: Generate poetry.lock

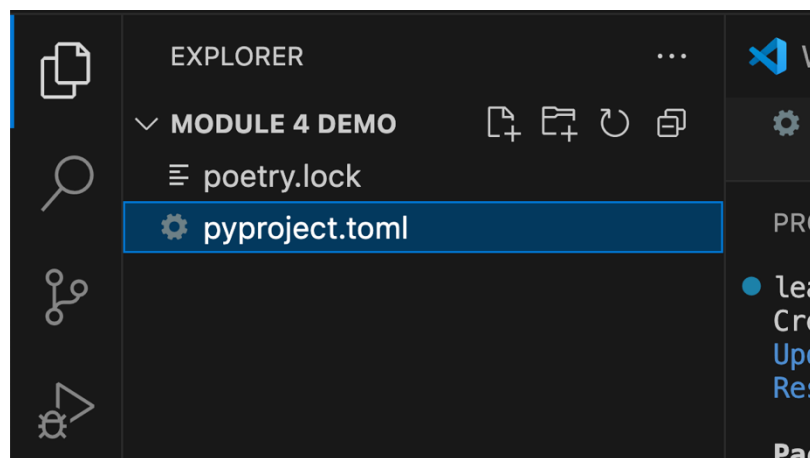
- 6.1 Save your pyproject.toml file and run the appropriate command (usually poetry install) to generate the poetry.lock file

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
learnersgalaxy@192 Module 4 Demo % poetry install --no-root
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
learnersgalaxy@192 Module 4 Demo % poetry install --no-root
Creating virtualenv trip-planner-2007w80d-py3.11 in /Users/learnersgalaxy/Library/Caches/pypoetry/virtualenvs
Updating dependencies
Resolving dependencies... (10.3s)

Package operations: 95 installs, 0 updates, 0 removals
- Installing typing-extensions (4.12.2)
- Installing certifi (2025.1.31)
- Installing charset-normalizer (3.4.1)
- Installing annotated-types (0.7.0)
- Installing idna (3.10)
- Installing pydantic-core (2.27.2)
- Installing urllib3 (2.3.0)
- Installing attrs (25.1.0)
- Installing multidict (6.1.0)
- Installing frozenlist (1.5.0)
- Installing mypy_extensions (1.0.0)
- Installing packaging (23.2)
```



C. Accessing API keys

Fetch SerperAPI AI key

Step 1: Create a SerperAPI account

- 1.1 Visit the SerperAPI website: <https://serperapi.com>
- 1.2 Click the **Sign Up** button if you don't have an account or **Log In** if you already do
- 1.3 Sign up with your email or log in using an existing account

Step 2: Verify your email

- 2.1 After signing up, check your inbox for a verification email
- 2.2 Click the verification link in the email to confirm your account

Step 3: Choose a plan

- 3.1 After signing in, go to the **Pricing** page or dashboard to choose a subscription plan that suits your needs
- 3.2 Select the free trial or a paid plan, depending on your usage requirements

Step 4: Access the API keys section

- 4.1 Once you're logged in, go to the **Dashboard** or the **API Keys** page
- 4.2 You should be able to see an option to generate or view your API key.

Step 5: Generate the API key

- 5.1 Click the **Generate New Key** button if one isn't automatically provided
- 5.2 The key will be displayed on the screen. Copy it immediately and save it somewhere secure.

Step 6: Use the API key

- 6.1 Now that you have your API key, you can use it to make requests to the SerperAPI.
- 6.2 In your application or code, include the API key in the header or request body to authenticate and access the data

By following the above-mentioned steps, you have successfully set up your Python project with pipx and poetry and securely integrated your API keys using a .env file. This streamlined approach lays a solid foundation for efficient and organized development.