21-02-2025

Creates a Python package (basic-module), builds it as a wheel, installs it globally, and run the installed script (basic-module).

Step 1: Make a directory basic_module and go to it

```
root@Goosari:~# mkdir basic_module
```

```
root@Goosari:~# cd basic_module
```

In the VS code terminal (helps us keep track of the files created)

Step 2: Create a virtual environment named datastructure and activate it.

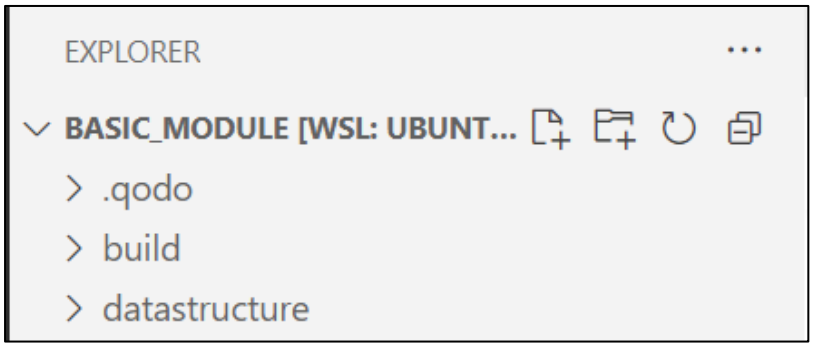
- Virtual environments allow you to isolate dependencies for Python projects.

```
root@Goosari:~/basic_module# sudo apt update
```

```
root@Goosari:~/basic_module# sudo apt install python3-venv python3-full
```

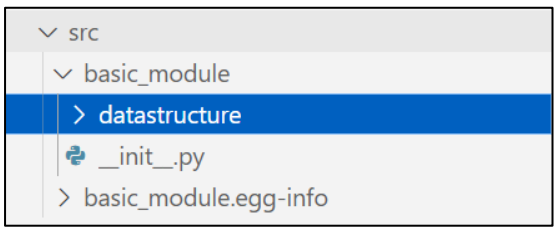
```
root@Goosari:~/basic_module# python3 -m venv datastructure
```

```
root@Goosari:~/basic_module# source datatstructure/bin/activate
```



Step 3: Creating the Project Directory and Writing a Python Script

```
(datastructure) root@Goosari:~/basic_module# mkdir -p src/basic_module/datastructure
```

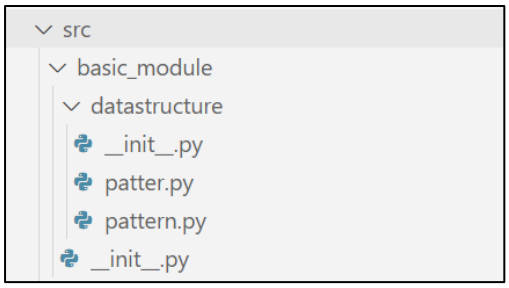


```
(datastructure) root@Goosari:~/basic_module# cat > src/basic_module/datastructure/patter.py << 'EOF'
> def main():
    N=9
    j=0
    for i in range(0,N):
        for j in range(0,i):
            print(j ,end=" ")
        print("")

if __name__ == "__main__":
    main()
> EOF
```

```
(datastructure) root@Goosari:~/basic_module# touch src/basic_module/__init__.py
```

```
(datastructure) root@Goosari:~/basic_module# touch src/basic_module/datastructure/__init__.py
```



Step 4: Creating the pyproject.toml (Build Configuration)

- Configuration file used by modern Python projects to specify build requirements, dependencies, and other metadata.

```
(datastructure) root@Goosari:~/basic_module# cat > pyproject.toml << "EOF"
> [build-system]
requires = ["setuptools>=42", "wheel"]
build-backend = "setuptools.build_meta"
[project]
name = "basic-module"
version = "0.1.0"
description = "A basic Python module"
readme = "README.md"
requires-python = ">=3.8"
dependencies = [
        "setuptools>=42",
        "wheel"
]
[project.scripts]
basic-module = "basic_module.datastructure.pattern:main"
[tool.setuptools]
package-dir = {"" = "src"}
packages = ["basic_module", "basic_module.datastructure"]
> EOF
```

```
(datastructure) root@Goosari:~/basic_module# echo "# Basic Module" > README.md
```

Step 5: Installing tools (build and pipx)

```
(datastructure) root@Goosari:~/basic_module# pip install build
```

- pipx is a tool specifically designed for installing and running Python applications that have command-line interfaces (CLIs)

Feature	pip	pipx
Purpose	Install and manage Python libraries and dependencies for projects	Install and manage standalone Python CLI applications
Focus	Libraries used in your code	Command-line tools
Isolation	Relies on manual virtual environment management	Automatically creates isolated environments for each application
Use Case	Installing packages needed by your project (e.g., within a virtual environment)	Installing and running CLI tools that you use across projects

```
(datastructure) root@Goosari:~/basic_module# python3 -m pip install --user pipx
```

Step 6: Building the Package

- Creates a wheel (.whl) file, which is a distributable Python package.
- A wheel file (.whl) is a built package format for Python.

```
(datastructure) root@Goosari:~/basic_module# python3 -m build --wheel
```

Step 7: Installing the Package Globally

```
(datastructure) root@Goosari:~/basic_module# pipx install dist/basic_module-0.1.0-py3-none-any.whl --force
```

Step 8: Deactivate the venv

```
(datastructure) root@Goosari:~/basic_module# deactivate
root@Goosari:~/basic_module#
```

Step 9: Updating the PATH

- This adds $HOME/.local/bin/ to the system's PATH in .bashrc, ensuring that any scripts or executables installed there can be run directly.

```
root@Goosari:~/basic_module# echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
```

```
root@Goosari:~/basic_module# chmod 777 ~/.local/bin/basic-module
```

```
root@Goosari:~/basic_module# source ~/.bashrc
```

- This reloads .bashrc so the changes take effect immediately, without needing to log out and back in.

```
root@Goosari:~/basic_module# which basic-module
/root/.local/bin/basic-module
root@Goosari:~/basic_module#
```

Step 10: Run the project

```
root@Goosari:~/basic_module# basic-module

0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
root@Goosari:~/basic_module#
```