**Python Project Setup Documentation**

**1. Project Overview**

This document outlines the step-by-step process for setting up a Python project, packaging it into a wheel, and installing it as a CLI tool.

**Project Name: basic-module**

**Purpose: Python package for pattern generation using a command-line interface (CLI).**

**2. Project Structure**

The final directory structure of the project is as follows:

basic-module/

├── build/ # Temporary build files

├── dist/ # Wheel output location

│ └── basic\_module-0.1.0-py3-none-any.whl

├── src/

│ └── basic\_module/

│ ├── \_\_init\_\_.py # Marks as package

│ ├── datastructure/

│ │ ├── \_\_init\_\_.py # Marks subpackage

│ │ └── pattern.py # Main function

├── venv/ # Virtual environment

├── pyproject.toml # Build configuration

├── README.md # Project description

└── .gitignore # Excluded files

**3. Virtual Environment Setup**

**3.1 Create Virtual Environment**

python3 -m venv venv

**3.2 Activate Virtual Environment**

source venv/bin/activate

**4. Project Configuration**

**4.1 pyproject.toml Configuration**

Create pyproject.toml in the project root:

[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"]

**4.2 requirements.txt (Optional)**

To list project dependencies:

setuptools>=42

wheel

**5. Write Project Code**

**5.1 src/basic\_module/datastructure/pattern.py**

def main():

N = 9

# Pattern 1

for i in range(0, N):

for j in range(0, i):

print(j, end=" ")

print("")

# Pattern 2

for i in range(0, N):

for j in range(0, i):

print(i, end=" ")

print("")

print()

# Pattern 3

for i in range(N, 0, -1):

for j in range(0, i):

print(i, end=" ")

print("")

print("")

# Pattern 4

for i in range(1, N + 1):

print(" " \* (N - i), end=" ")

print(f"{i} " \* i)

# Pattern 5

print(" ".join(map(str, range(1, 10))))

for i in range(1, N + 1):

print(" " \* (N - i) + " ".join(map(str, range(1, i + 1))))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**5.2 \_\_init\_\_.py Files**

Ensure the following files exist:

**src/basic\_module/\_\_init\_\_.py:**

from .datastructure.pattern import main

**src/basic\_module/datastructure/\_\_init\_\_.py:**

from .pattern import main

**6. Build the Wheel**

1. **Install Build Tool:**

pip install build

1. **Build the Wheel:**

python3 -m build --wheel

This generates the wheel file in the dist/ directory:

dist/

└── basic\_module-0.1.0-py3-none-any.whl

**7. Install and Test the Package**

1. **Uninstall Old Package (if any):**

pip uninstall -y basic-module

1. **Install the New Wheel:**

pip install dist/basic\_module-0.1.0-py3-none-any.whl --break-system-packages

1. **Verify Installation:**

pip show basic-module

**8. Run the CLI Command**

1. **Run the Script:**

basic-module

Expected Output:

Pattern output:

0

0 1

0 1 2

0 1 2 3

...

1. **Run as a Module:**

python3 -m basic\_module.datastructure.pattern

**9. Set PATH for System-Wide Access (Optional)**

Add the local bin directory to PATH:

echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc

source ~/.bashrc

**10. Copy Wheel to External Location (Optional)**

To copy the wheel to another location, for example, D:/wheel:

cp dist/basic\_module-0.1.0-py3-none-any.whl /mnt/d/wheel/

**11. Cleanup (Optional)**

To remove build artifacts:

rm -rf build dist \*.egg-info

**12. Conclusion**

* Successfully created a Python package.
* Packaged as a wheel for easy distribution.
* Installed and tested as a CLI tool.