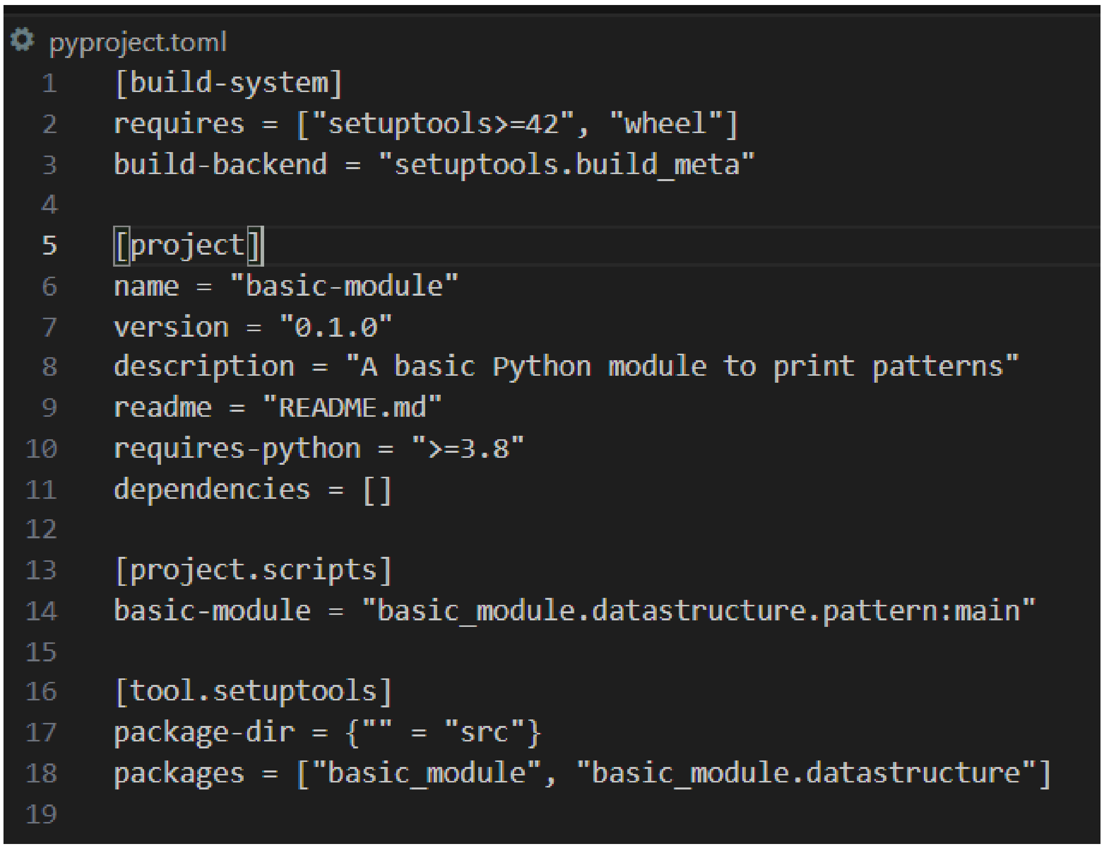
SRE Training (Day 10) - PYTHON

SETTING UP A PYTHON PROJECT

The **`pyproject.toml`** file is a configuration file used to define build instructions, project metadata, and dependencies for Python projects.

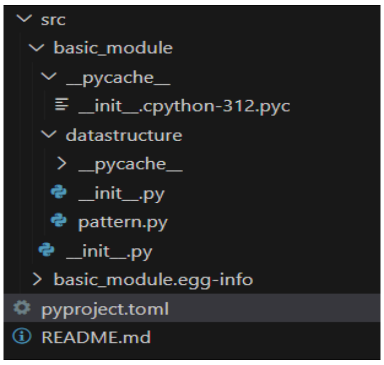


pyproject.toml

```toml
[build-system]
requires = ["setuptools>=42", "wheel"]
build-backend = "setuptools.build_meta"

[project]
name = "basic-module"
version = "0.1.0"
description = "A basic Python module to print patterns"
readme = "README.md"
requires-python = ">=3.8"
dependencies = []

[project.scripts]
basic-module = "basic_module.datastructure.pattern:main"

[tool.setuptools]
package-dir = {"" = "src"}
packages = ["basic_module", "basic_module.datastructure"]

```

__init__.py file

_init__.py is a special file used in Python to define packages and initialize their namespaces.



```
src
  basic_module
    __pycache__
      __init__.cpython-312.pyc
    datastructure
      __pycache__
      __init__.py
      pattern.py
    __init__.py
  basic_module.egg-info
pyproject.toml
README.md
```

PYTHON VIRTUAL ENVIRONMENT

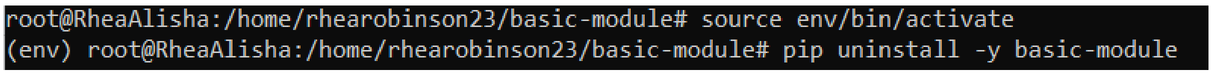
🎯 Why Virtual Environments are Important:

- Isolation of project dependencies.
- Avoiding conflicts with system Python packages.

COMMANDS:

- Creating - python3 -m venv env
- Activating - source env/bin/activate

*env - environment name



```
root@RheaAlisha:/home/rhearobinson23/basic-module# source env/bin/activate
(env) root@RheaAlisha:/home/rhearobinson23/basic-module# pip uninstall -y basic-module
```

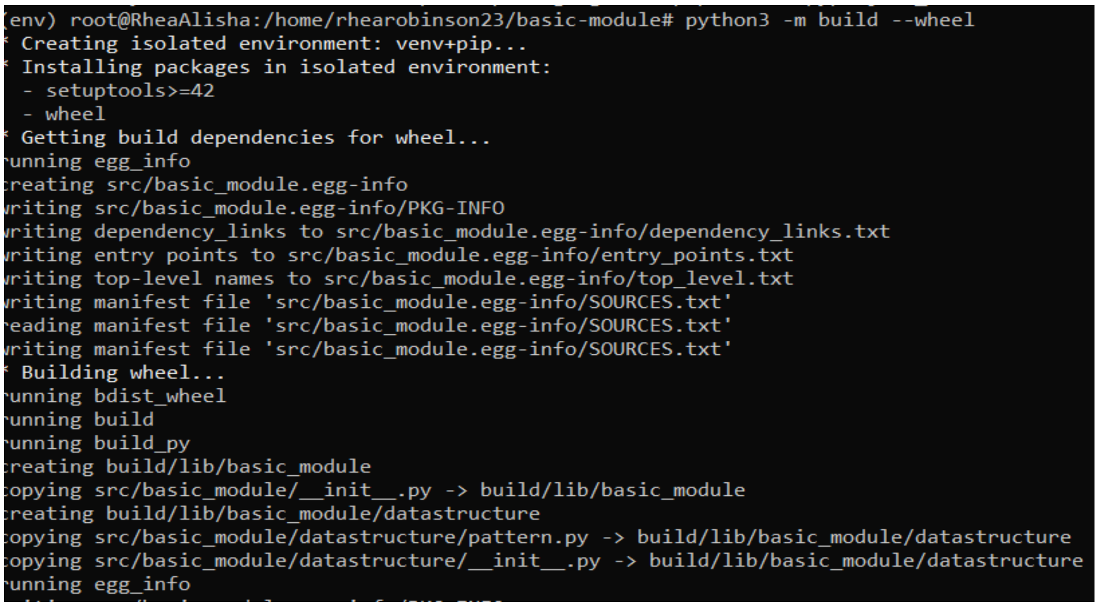
PYTHON PACKAGING & DISTRIBUTION

Using `pyproject.toml` for Modern Packaging

- `[build-system]`: Declaring build dependencies (`setuptools`, `wheel`).
- `[project]`: Defining package metadata (name, version, description).
- `[project.scripts]`: Creating CLI commands linked to Python functions.

Building Python Packages using `build`:

The **`build`** package in Python is a **modern, standardized tool** used to **build Python projects**. It works with the **`pyproject.toml`** file to create **distributable packages** like **wheels (`.whl`)** and **source archives (`.tar.gz`)**.



```
(env) root@RheaAlisha:/home/rhearobinson23/basic-module# python3 -m build --wheel
* Creating isolated environment: venv+pip...
* Installing packages in isolated environment:
  - setuptools>=42
  - wheel
* Getting build dependencies for wheel...
running egg_info
creating src/basic_module.egg-info
writing src/basic_module.egg-info/PKG-INFO
writing dependency_links to src/basic_module.egg-info/dependency_links.txt
writing entry points to src/basic_module.egg-info/entry_points.txt
writing top-level names to src/basic_module.egg-info/top_level.txt
writing manifest file 'src/basic_module.egg-info/SOURCES.txt'
reading manifest file 'src/basic_module.egg-info/SOURCES.txt'
writing manifest file 'src/basic_module.egg-info/SOURCES.txt'
* Building wheel...
running bdist_wheel
running build
running build_py
creating build/lib/basic_module
copying src/basic_module/__init__.py -> build/lib/basic_module
creating build/lib/basic_module/datastructure
copying src/basic_module/datastructure/pattern.py -> build/lib/basic_module/datastructure
copying src/basic_module/datastructure/__init__.py -> build/lib/basic_module/datastructure
running egg_info
```

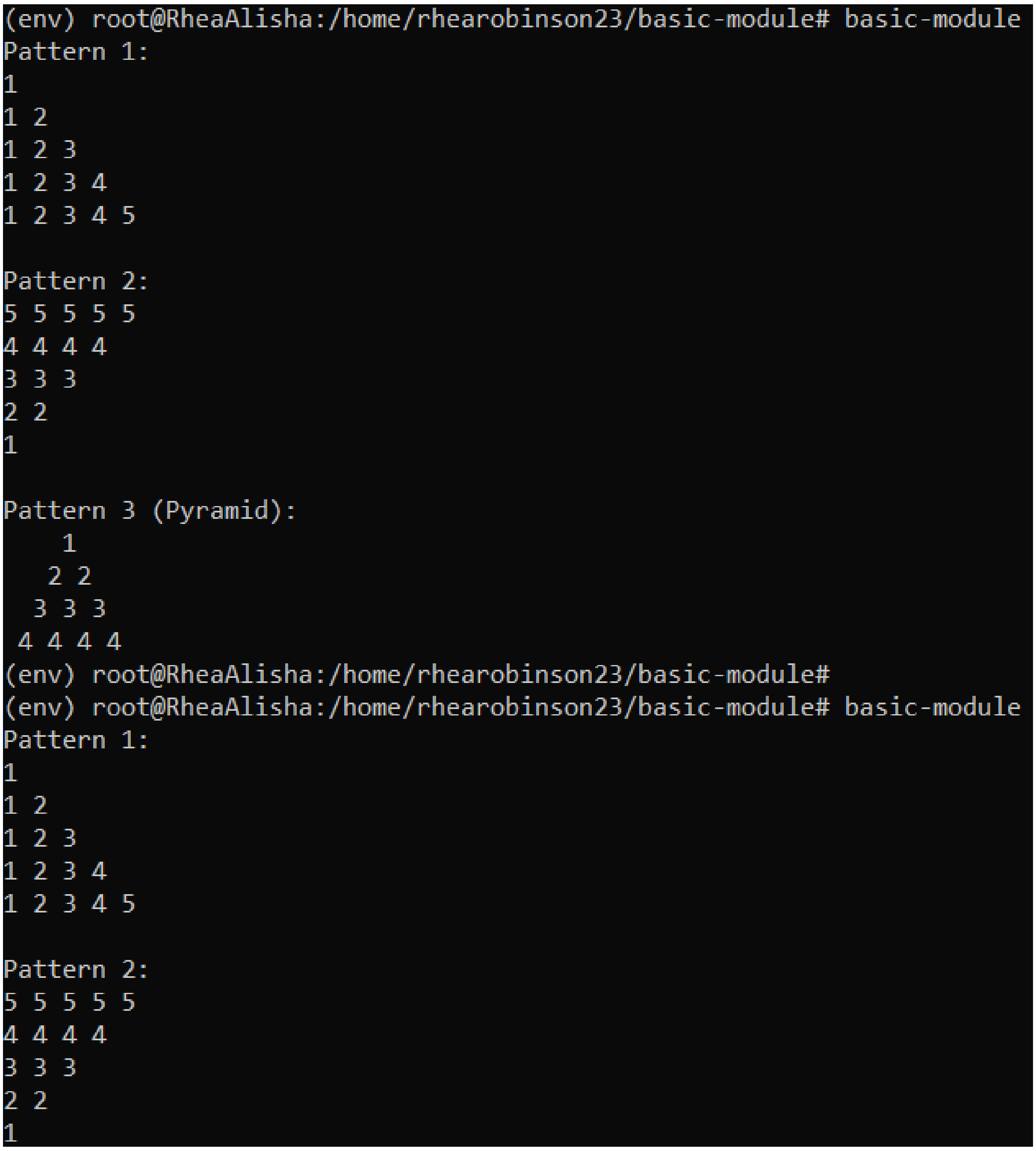
A wheel is a binary distribution format that allows faster installations without compilation.

The command creates a `.whl` file in the `dist/` directory.

EDITABLE INSTALLATION VS. BUILD

pip install -e .

- Installs the package directly from the source without rebuilding.
- Changes in code are immediately reflected.



```
(env) root@RheaAlisha:/home/rhearobinson23/basic-module# basic-module
Pattern 1:
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

Pattern 2:
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1

Pattern 3 (Pyramid):
    1
   2 2
  3 3 3
 4 4 4 4
(env) root@RheaAlisha:/home/rhearobinson23/basic-module#
(env) root@RheaAlisha:/home/rhearobinson23/basic-module# basic-module
Pattern 1:
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

Pattern 2:
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```