Creating a wheel file in Python:

- we create a .toml extension file which tells the structure to convert our python file into veil file
- install qodogen extension on vs code for creating .toml setup for our folder basic-module
- what are artifacts: when u combine multiple python file it is known as veil file, and multiple java file combines to form jar file. and so on and these are known as artifacts.

Welcome | pyproject.toml

pyproject.toml

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

[project]
name = "basic-module"
version = "0.1.0"
description = "A basic Python module"
readme = "README.md"
requires-python = ">=3.8"
classifiers = [
    "Programming Language :: Python :: 3",
    "License :: OSI Approved :: MIT License",
    "Operating System :: OS Independent",
]

[project.urls]
Homepage = "https://github.com/yourusername/basic-module"

[project.dependencies]
setuptools = ">=42"
wheel = "*"

[project.urls]
```

Then we create a requirements.txt for storing basic requirements to convert our python files into wheel file:

Welcome | pyproject.toml | requirements.txt

requirements.txt

```
python>=3.10,<3.11
setuptools>=60.0
wheel>=0.37.0
pandas>=1.5.0
numpy>=1.23.0
flask>=2.3.0
fastapi>=0.68.0
pytest>=7.0.0
mypy>=0.990
asyncio>=3.4.3
```

and then in integrated terminal we would install requirements.txt

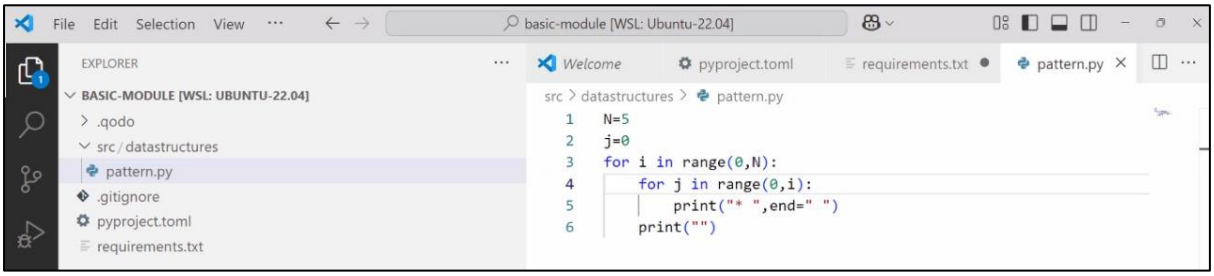
PROBLEMS OUTPUT TERMINAL ··· pip - basic-module

```
PS Microsoft.PowerShell.Core\FileSystem::\\wsl.localhost\Ubuntu\home
\root1\basic-module>  pip install -r requirements.txt
Requirement already satisfied: setuptools>=42.0 in c:\users\hp\anaco
nda3\lib\site-packages (from -r requirements.txt (line 1)) (61.2.0)
Requirement already satisfied: wheel>=0.37.0 in c:\users\hp\anaconda
3\lib\site-packages (from -r requirements.txt (line 2)) (0.37.1)
Collecting pandas>=1.5.0a
  Downloading pandas-2.2.3-cp39-cp39-win_amd64.whl (11.6 MB)
     |                                        | 11.6 MB 939 kB/s
```

get me .gitignore for a python project for files which i dont need to checkin if build project locally and do things

Use this prompt and qodogen would create the gitignore file for you.

and then we would run a simple python file to code for pattern:



we used end="" as by default end refers to new line, therefore to print a star pattern.

```
for i in range(1, N + 1):
    print(" " * (N - i), end=" ")
    print("* " * i)
```

```
for i in range(1, N + 1):
    print(" " * (N - i)+" ".join(map(str,range(1,i))))
```

```
      1
     1 2
    1 2 3
   1 2 3 4
  1 2 3 4 5
 1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
```

How to create a wheel file:

1. First, let's clean up and create a fresh project: ```bash cd ~/basic-module rm -rf * .* # Be careful with this command! ```
2. Create the project structure: ```bash mkdir -p src/basic_module/datastructure ```
3. Create the pattern.py file: ```bash cat > src/basic_module/datastructure/pattern.py << 'EOF' def main(): N=9 j=0 for i in range(0,N): for j in range(0,i): print(j ,end=" ") print("") for i in range(0,N): for j in range(0,i): print(i ,end=" ") print("") print() for i in range(N,0,-1): for j in range(0,i): print(i ,end=" ") print("") print("") for i in range(1, N + 1): print(" " * (N - i), end=" ") print(f"{i} " * i) print(" ".join(map(str,range(1,10)))) for i in range(1, N + 1): print(" " * (N - i)+" ".join(map(str,(1,i)))) if __name__ == "__main__": main() EOF
4. Create __init__.py files: ```bash touch src/basic_module/__init__.py touch src/basic_module/datastructure/__init__.py ```
5. Create pyproject.toml: ```bash 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 ```
6. Create README.md: ```bash echo "# Basic Module" > README.md ```
7. Build and install: ```bash # Make sure we're not in a virtualenv deactivate # Remove old installations pip uninstall -y basic-module # Build new wheel python3 -m build –wheel

1. # Install the wheel pip install dist/basic_module-0.1.0-py3-none-any.whl --force-reinstall ```
8. Now go in to ubuntu terminal home echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc chmod +x ~/.local/bin/basic-module which basic-module basic-module

The key changes are: 1. Changed import in project.scripts to use `basic_module.datastructure.pattern:main` 2. Properly structured the package under src/basic_module/ 3. Made sure the package installation is done outside of virtualenvIf you're still in a virtualenv, first exit it with `deactivate` command, then try theinstallation and running steps again. This should resolve the module not founderror