Creating your own Python library is a great way to organize reusable code and even share it with others. Here's a step-by-step guide to get you started:

🧱 1. Set Up Your Project Structure

Create a folder for your library:

Copy

```
my_library/
├── my_library/
│   ├── __init__.py
│   └── core.py
├── tests/
│   └── test_core.py
├── README.md
├── setup.py
└── pyproject.toml
```

- `__init__.py`: Makes the folder a package.
- `core.py`: Your main module with functions/classes.
- `tests/`: Unit tests for your library.
- `setup.py` and `pyproject.toml`: For packaging and distribution.

🧠 2. Write Your Code

In `core.py`, define your functions or classes:

Python Copy

```python
# core.py
def greet(name):
    return f"Hello, {name}!"
```

Then expose it in `__init__.py`:

Python Copy

```python
# __init__.py
from .core import greet
```

Now users can do:

Python Copy

```python
from my_library import greet
```

🧪 3. Add Tests

Use `unittest` or `pytest`:

Python Copy

```python
# test_core.py
from my_library import greet

def test_greet():
    assert greet("Aravind") == "Hello, Aravind!"
```

📦 4. Package It

In `setup.py`:

Python Copy

```python
from setuptools import setup, find_packages

setup(
    name="my_library",
    version="0.1",
    packages=find_packages(),
    install_requires=[],
)
```

In `pyproject.toml` (modern packaging):

Toml Copy

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

🚀 5. Install Locally or Publish

- Install locally:

  Bash Copy

  ```bash
  pip install .
  ```

- Publish to PyPI:
  1. Build: `python setup.py sdist bdist_wheel`
  2. Upload: `twine upload dist/*`