Step 1: Create the Directory Structure

First, set up the directory and file structure:

```
geometry_toolkit/
├── geometry_toolkit/
│   ├── __init__.py
│   ├── shapes.py
├── setup.py
├── LICENSE
├── README.md
└── pyproject.toml
```

Step 2: Implement the Code

geometry_toolkit/shapes.py

This file will contain the actual code for calculating areas and perimeters.

```
# geometry_toolkit/shapes.py


import math
def rectangle_area(length, width):
    return length * width
def rectangle_perimeter(length, width):
    return 2 * (length + width)
def square_area(side):
    return side * side
def square_perimeter(side):
    return 4 * side
def circle_area(radius):
    return math.pi * radius * radius
def circle_circumference(radius):
    return 2 * math.pi * radius
```

- **geometry_toolkit/__init__.py**

This file makes it easy to import functions directly from the package. Create

geometry_toolkit /__init__.py to make the functions accessible when you import the package:

geometry_toolkit/__init__.py

```
from .shapes import (
    rectangle_area,
    rectangle_perimeter,
    square_area,
    square_perimeter,
    circle_area,
    circle_circumference,
)

__all__ =[ “rectangle_area”,
    “rectangle_perimete”r,
   “square_area”,
   “square_perimeter”,
    “circle_area”,
    “circle_circumference”
]
```

__all__: This is a list defining the public interface of the module. When you use from geometry_toolkit import *, only these specified functions will be imported.

The __all__ variable is a list that specifies which names will be publicly available when the package is imported using from your_package_name import *. In this case, only rectangle_area, rectangle_perimeter,square_area, square_perimeter,circle_area,circle_circumference,

will be exposed for import.

Step 3: Create Setup Files

setup.py

This file is the main setup file that includes the package metadata.

```
# setup.py

from setuptools import setup, find_packages

setup(
    name="geometry_toolkit",
    version="0.1.0",
    author="Your Name",
    author_email="your.email@example.com",
    description="A toolkit for basic geometric calculations",
    long_description=open("README.md").read(),
    long_description_content_type="text/markdown",
    url="https://github.com/yourusername/geometry_toolkit",
    packages=find_packages(),
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    python_requires=">=3.6",
)
```

- Replace "Your Name" with your actual name.
- Replace "your.email@example.com" with your actual email.
- Update "https://github.com/yourusername/math_toolkit" with the correct URL to your GitHub repository for the package, if you plan to host it there.

- **pyproject.toml**

This file specifies the build system requirements. Create pyproject.toml to define setuptools and wheel as dependencies for building the package:

```
[build-system]
```

requires = ["setuptools", "wheel"]

build-backend = "setuptools.build_meta"

Explanation

- **requires = ["setuptools", "wheel"]**: This specifies that setuptools and wheel are needed to build the package. These are essential tools for creating Python package distributions.
- **build-backend = "setuptools.build_meta"**: This tells Python to use setuptools as the backend for building the package.

Why It's Needed

The pyproject.toml file is part of PEP 517, which defines a standard way to specify build dependencies. Having this file ensures that when users or CI/CD pipelines try to build your package, the necessary tools (in this case, setuptools and wheel) are automatically installed.

- **README.md**

This file provides an overview of the package and usage examples.

Geometry Toolkit

`geometry_toolkit` is a simple Python package for basic geometric calculations.

Installation

Install the package with pip:

pip install geometry_toolkit

- **Usage**

```
from geometry_toolkit import (
    rectangle_area,
    rectangle_perimeter,
    square_area,
    square_perimeter,
    circle_area,
    circle_circumference,
```

)

Rectangle
print(rectangle_area(5, 3))   # Output: 15
print(rectangle_perimeter(5, 3))   # Output: 16

Square
print(square_area(4))   # Output: 16
print(square_perimeter(4))   # Output: 16

Circle
print(circle_area(3))   # Output: 28.274333882308138
print(circle_circumference(3))   # Output: 18.84955592153876

`LICENSE`

Choose a license, such as MIT License. Here's a sample MIT License template:

```plaintext
MIT License

Copyright (c) 2024 Your Name

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

[... License text continues ...]
```

Step 4: Build the Package
Once you have all the files in place, use the following commands to build the package:

1.Navigate to your package directory:

```
cd geometry_toolkit
```

2. Build the package:

```
python setup.py sdist bdist_wheel
```

Ensure the dist/ directory has files like math_toolkit-0.1.0.tar.gz and math_toolkit-0.1.0-py3-none-any.whl.

This will generate a dist/ folder containing .tar.gz and .whl files.

Step 5: Publish to PyPI

If you haven't already, create an account on PyPI. Then, upload the package using Twine:
1.Install twine (if you haven't already):
pip install twine

2.Upload the package to PyPI:
twine upload dist/*

3.Enter your PyPI username and password or API token when prompted.

Step 1: Log In to PyPI

- Go to https://pypi.org/ and log in with your account.
- If you don't have an account, create one and confirm your email before proceeding.

Step 2: Go to API Tokens

- After logging in, click on your profile icon in the top-right corner.
- Select Account settings from the dropdown menu.
- Scroll down to the API tokens section.

Step 3: Create a New API Token

- Click on Add API token.
- In the Name field, enter a name for your token (e.g., geometry_toolkit).
- Under Scope, choose:
- Entire account if you want the token to work for all your projects.
- Specify a project if you want the token to work for a specific project only (e.g., geometry_toolkit).
- Click Create token.

Step 4: Copy the Token

- After creation, PyPI will show you the token only once. Copy it immediately and store it in a secure place.
- Do not share this token publicly, as it provides full access to publish and update your packages

Step 6: Install and Use the Package
After publishing, you (or anyone else) can install the package from PyPI:
pip install geometry_toolkit

To use it:

```
from geometry_toolkit import (
    rectangle_area,
    rectangle_perimeter,
    square_area,
    square_perimeter,
    circle_area,
    circle_circumference,
)

print(rectangle_area(5, 3))        # 15
print(square_area(4))              # 16
print(circle_circumference(3))     # 18.84955592153876
```

Add Python Scripts Directory to PATH (Windows only)

If twine is still not recognized, you might need to manually add the Scripts directory to your PATH:

C:\Users\<YourUsername>\AppData\Local\Programs\Python\Python<version>\Scripts