Module 1: Introduction to Python Libraries (3 Hours)

1.1 What is a Python Library? - Overview of libraries vs. scripts, use cases, and advantages.

1.2 Python Packaging Basics - Python module structure, importing and using libraries.

1.3 Tools for Building Libraries - Overview of setuptools and pip, introduction to pyproject.toml.

Module 2: Creating a Python Library (6 Hours)

2.1 Structuring a Python Package

- Creating __init__.py, Structuring directories for single and multi-module packages.
- **Mapping to Multimedia**: Structure a package for an **image processing library** that includes resizing, format conversion, and basic filters.

2.2 Writing Functions and Classes

- Building reusable functions, Designing modular and scalable classes.
- **Mapping to Multimedia**: Create functions like resize_image(), convert_format() in a class that handles image operations (e.g., resizing and converting formats for images).

2.3 Adding Configuration Files

- Creating setup.py, Using pyproject.toml and setup.cfg, Adding dependencies.
- **Mapping to Multimedia**: Add dependencies for libraries such as **Pillow** and **OpenCV** to the configuration files, which will be used for image manipulation tasks.

Module 3: Documentation and Testing (6 Hours)

3.1 Writing Documentation

- Writing docstrings (Google, NumPy, and Sphinx styles), Generating HTML documentation with Sphinx.
- **Mapping to Multimedia**: Document the image processing functions in your library (e.g., resize_image()), including examples of usage (how to resize an image) and expected input/output (image dimensions and format).

3.2 Writing Unit Tests

- Using unittest and pytest, Creating test cases for your library.

- **Mapping to Multimedia**: Write tests for image manipulation functions (e.g., check if the resizing function works correctly for different image sizes or formats).

3.3 Automating Tests

- Setting up continuous integration (GitHub Actions or Travis CI), Automating test runs and linting.
- **Mapping to Multimedia**: Set up automated tests for the image processing library to ensure that new commits do not break functionality (e.g., image resizing, format conversion).

Module 4: Version Control with Git and GitHub (6 Hours)
4.1 Git Basics - Initializing a Git repository, staging, committing, and branching.
4.2 Advanced Git Features - Using tags for versioning, resolving merge conflicts, writing meaningful commit messages.
4.3 Publishing on GitHub - Creating a GitHub repository, pushing local code to GitHub, collaborating using pull requests and issues.

Module 5: Publishing the Library (5 Hours)
5.1 Packaging for PyPI - Creating MANIFEST.in for additional files, building distributable packages with setup tools, and testing your package locally.
5.2 Publishing on PyPI - Uploading to TestPyPI, releasing on PyPI using twine.
5.3 Maintaining Your Library - Updating and versioning, handling user feedback and issues.

Module 6: Best Practices for Open-Source Development (4 Hours)
6.1 Writing Clean Code - Using linters like flake8, adopting PEP 8 guidelines.
6.2 Creating an Open-Source Repository - Adding README.md with usage examples, including LICENSE and CONTRIBUTING.md.
6.3 Managing Open-Source Contributions - Reviewing pull requests, using GitHub Discussions and Issues effectively.

Module 7: Capstone Project
7.1 Building a Python Library from Scratch - Create a library (e.g., a utility tool, data parser, or API wrapper), document, test, and package it.
7.2 Publishing on GitHub and PyPI - Push the project to GitHub, publish it to PyPI, and test installation.