**Creating Python Library**

Creating a Python library and using it in another program involves several steps. Below is a guide to help you create a library, package it, and use it in another program.  
  
---  
  
### \*\*Step 1: Create Your Python Library\*\*  
A Python library is essentially a collection of Python modules (files with `.py` extension) that you can reuse in other projects.  
  
1. \*\*Create a Directory for Your Library\*\*  
 Create a folder for your library. For example, let's call it `mylibrary`.  
  
 ```  
 mylibrary/  
 ├── mymodule.py  
 ├── \_\_init\_\_.py  
 ```  
  
2. \*\*Write Your Code\*\*  
 Add your Python code to a file inside the library folder. For example, create a file called `mymodule.py`:  
  
 ```python  
 # mymodule.py  
 def add(a, b):  
 return a + b  
  
 def subtract(a, b):  
 return a - b  
 ```  
  
 The `\_\_init\_\_.py` file can be empty or used to initialize the library.  
  
---  
  
### \*\*Step 2: Package Your Library\*\*  
To make your library reusable and installable, you need to package it.  
  
1. \*\*Create a `setup.py` File\*\*  
 In the root directory of your library, create a `setup.py` file:  
  
 ```python  
 from setuptools import setup, find\_packages  
  
 setup(  
 name="mylibrary", # Name of your library  
 version="0.1.0", # Version of your library  
 packages=find\_packages(), # Automatically find modules  
 description="A simple example Python library",  
 author="Your Name",  
 author\_email="your.email@example.com",  
 url="https://github.com/yourusername/mylibrary", # Optional  
 classifiers=[  
 "Programming Language :: Python :: 3",  
 "License :: OSI Approved :: MIT License",  
 "Operating System :: OS Independent",  
 ],  
 python\_requires=">=3.6",  
 )  
 ```  
  
2. \*\*Create a `README.md` File\*\*  
 Add a `README.md` file to describe your library. This is useful for documentation.  
  
3. \*\*Build the Package\*\*  
 Install the required tools and build your package:  
  
 ```bash  
 pip install setuptools wheel  
 python setup.py sdist bdist\_wheel  
 ```  
  
 This will create a `dist/` directory containing your package files (e.g., `mylibrary-0.1.0.tar.gz` and `mylibrary-0.1.0-py3-none-any.whl`).  
  
4. \*\*Upload to PyPI (Optional)\*\*  
 If you want to share your library publicly, you can upload it to [PyPI](https://pypi.org/).  
  
 Install `twine`:  
  
 ```bash  
 pip install twine  
 ```  
  
 Upload your package:  
  
 ```bash  
 twine upload dist/\*  
 ```  
  
---  
  
### \*\*Step 3: Install and Use Your Library\*\*  
Once your library is packaged, you can install it and use it in another program.  
  
1. \*\*Install the Library\*\*  
 If the library is local, install it using `pip`:  
  
 ```bash  
 pip install /path/to/mylibrary  
 ```  
  
 If the library is on PyPI, install it directly:  
  
 ```bash  
 pip install mylibrary  
 ```  
  
2. \*\*Use the Library in Another Program\*\*  
 Create a new Python script and import your library:  
  
 ```python  
 # main\_program.py  
 from mylibrary.mymodule import add, subtract  
  
 print(add(5, 3)) # Output: 8  
 print(subtract(5, 3)) # Output: 2  
 ```  
  
 Run the program:  
  
 ```bash  
 python main\_program.py  
 ```  
  
---  
  
### \*\*Example Directory Structure\*\*  
Here’s what the final directory structure might look like:  
  
```  
mylibrary/  
├── mylibrary/  
│ ├── \_\_init\_\_.py  
│ ├── mymodule.py  
├── setup.py  
├── README.md  
```  
  
---  
  
### \*\*Summary\*\*  
1. Create a folder with your library code.  
2. Add a `setup.py` file to package the library.  
3. Build the package using `setuptools`.  
4. Install the library locally or upload it to PyPI.  
5. Use the library in other programs by importing it.