Python Package Development & Publishing – Lab Guide

1.1 Create the Project Folder

What is this step for?

Before writing any code, we need to **create a structured folder** to organize our package.

Command:

```
cd ~/Documents
mkdir stringutils
cd stringutils
```

Explanation:

- `cd ~/Documents` → Moves into the `Documents` directory.
- `mkdir stringutils` → Creates a new folder named `stringutils`.
- `cd stringutils` → Navigates inside the newly created folder.

1.2 Create the Required Folders and Files

Command:

```
mkdir -p src/stringutils tests
touch src/stringutils/__init__.py
touch src/stringutils/text_operations.py
touch pyproject.toml LICENSE README.md
touch tests/test_text_operations.py
```

Explanation:

- `mkdir -p src/stringutils tests` → Creates the main **package folder (`src/stringutils/`)** and **test folder (`tests/`)**.
- `touch` → Creates new empty files that we will edit later.
- `__init__.py` → This file is required to **mark a directory as a package** in Python.

After this step, your folder structure should look like this:

stringutils/

│__ src/

│ │__ stringutils/

│ │__ __init__.py

│ │__ text_operations.py

│__ tests/

│ │__ test_text_operations.py

│__ pyproject.toml

│__ LICENSE

│__ README.md

Step 2: Write the Package Code

2.1 Define String Manipulation Functions in `text_operations.py`

Command: Open `src/stringutils/text_operations.py` and add:

```
def to_uppercase(text):
    """Converts a string to uppercase."""
    return text.upper()

def to_lowercase(text):
    """Converts a string to lowercase."""
    return text.lower()

def capitalize_first_letter(text):
    """Capitalizes only the first letter of the string."""
    return text.capitalize()

def capitalize_each_word(text):
    """Capitalizes the first letter of every word in the string."""
    return text.title()
```

Explanation:

- This module provides basic string manipulation functions.
- Functions include **converting text to uppercase, lowercase, and capitalizing words**.

2.2 Create `README.md`

Command: Open `README.md` and add:

StringUtils
A simple Python package for basic string manipulation.

Usage

from stringutils.text_operations import to_uppercase, capitalize_each_word

print(to_uppercase("hello"))

print(capitalize_each_word("hello world"))

Explanation:

- The `README.md` provides **documentation** for the package.

- It includes **installation instructions and usage examples**.

2.3 Define `pyproject.toml`

Command: Open `pyproject.toml` and add:

```toml

[project]

name = "stringutils-lab"

version = "0.0.1"

authors = [

  { name="Sophie Tsanang", email="sophietsanang@gmail.com" },
```

]

description = "A simple Python package for basic string manipulation"

readme = "README.md"

requires-python = ">=3.8"

classifiers = [

"Programming Language :: Python :: 3",

"License :: OSI Approved :: MIT License",

"Operating System :: OS Independent",

]

[build-system]

requires = ["setuptools", "wheel"]

build-backend = "setuptools.build_meta"

Explanation:

- Defines **metadata** about the package, such as the name, author, and version.
- `"setuptools"` and `"wheel"` are required for building the package.

Step 3: Set Up a Virtual Environment

Command:

- **Mac/Linux**:

```
python3 -m venv stringenv

source stringenv/bin/activate
```

🔹 **Windows**:

```
python -m venv stringenv

stringenv\Scripts\activate
```

Explanation:

- Creates a **virtual environment (`stringenv`)** that isolates package dependencies.
- Activating ensures that we install libraries in an isolated environment.

Step 4: Build and Upload the Package

4.1 Build the Package

```
python3 -m build
```

Explanation:

- Creates a **`dist/` folder** with distributable files (`.tar.gz` and `.whl`).

4.2 Upload to Test PyPI

```
python3 -m twine upload --repository testpypi dist/*
```

Explanation:

- Uploads the package to **Test PyPI**.
- You'll need to enter **your Test PyPI API token**.

Step 5: Install and Test the Package

5.1 Install from Test PyPI

- Get link from Test PyPI

Write Test Cases in `test_text_operations.py`

```
import sys
import os
sys.path.insert(0,
os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'src')))
from stringutils.text_operations import to_lowercase, to_uppercase,
capitalize_first_letter

def test_to_lowercase():
    assert to_lowercase("HELLO") == "hello"

def test_to_uppercase():
    assert to_uppercase("world") == "WORLD"

def test_capitalize_first_letter():
    assert capitalize_first_letter("python") == "Python"

print("All tests passed successfully!")
```

5.3 Run the Tests

- On VS Code

Step 6: Deactivate the Virtual Environment

```
deactivate
```

Explanation:

- This returns your terminal to the normal Python environment.