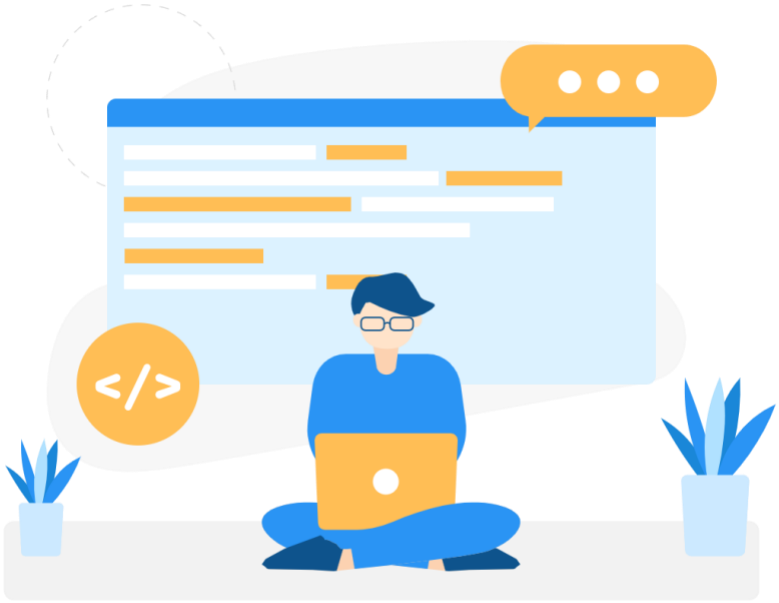
# BAB 7

# Python Library



## 7.1 Tujuan

1. Dapat menjelaskan konsep penggunaan library external pada python
2. Dapat mengimplementasikan library external seperti pathlib dan pytest pada pemrograman bahasa python

## 7.2 Pengantar

Python adalah salah satu bahasa pemrograman yang sangat populer di dunia. Popularitasnya tidak hanya karena sintaks yang sederhana dan mudah dipelajari, tetapi juga karena dukungan ekosistemnya yang luas. Python menyediakan banyak library bawaan (built-in library) seperti math, os, random, datetime, dan lain sebagainya. Library-library ini sudah mencakup berbagai kebutuhan dasar

pemrograman. Misalnya, kita bisa menggunakan math untuk operasi matematika, os untuk berinteraksi dengan sistem operasi, dan random untuk menghasilkan angka acak.

Namun, dalam praktiknya, kebutuhan pengembangan perangkat lunak tidak selalu dapat dipenuhi oleh library bawaan. Seiring berkembangnya aplikasi, kita sering memerlukan fitur tambahan seperti pemrosesan data berskala besar, komunikasi jaringan, pengolahan gambar, bahkan machine learning. Pada titik inilah library eksternal berperan penting.

### 7.2.1 Konsep Library External

Library eksternal adalah paket kode Python yang dibuat oleh komunitas atau pengembang lain di luar distribusi standar Python. Artinya, library ini tidak langsung tersedia saat kita pertama kali menginstal Python, melainkan perlu diunduh dan dipasang secara terpisah.

Dengan adanya library eksternal, seorang pengembang tidak perlu membangun semua fungsionalitas dari nol. Katakanlah kita ingin mengambil data dari internet. Jika menggunakan cara manual, kita harus membuka koneksi socket, mengirim request HTTP, membaca response, dan menguraikannya. Itu pekerjaan rumit. Tetapi dengan library eksternal seperti requests, kita cukup menulis satu baris:

```
1. import requests
2. response = requests.get("https://example.com")
3. print(response.text)
```

Untuk menggunakan library eksternal, Python menyediakan package manager bernama pip (Python Package Installer). pip memungkinkan kita menginstal, memperbarui, dan menghapus library eksternal dengan perintah sederhana di terminal atau command prompt. Hampir semua library eksternal Python didistribusikan melalui Python Package Index (PyPI), yaitu repositori resmi yang menampung lebih dari 400 ribu paket Python. Situs resminya dapat diakses di https://pypi.org

Contoh instalasi library eksternal:

```
1. pip install requests
2. pip install numpy
3. pip install pytest
```

Beberapa library eksternal yang sering digunakan antara lain:

- requests : untuk komunikasi HTTP, misalnya mengambil data dari API.
- numpy : untuk perhitungan numerik dan operasi pada array multidimensi.
- pandas : untuk analisis data, manipulasi tabel, dan pengolahan dataset.
- matplotlib : untuk membuat grafik dan visualisasi data.
- scikit-learn : untuk machine learning.

- pytest : untuk pengujian otomatis.

### 7.2.2 pathlib

pathlib adalah library di Python yang digunakan untuk mempermudah manipulasi path (lokasi file dan direktori). Sebelum pathlib diperkenalkan, Python sudah memiliki os.path untuk tujuan serupa, namun pathlib menawarkan pendekatan berbasis objek yang lebih intuitif dan modern.
Fungsi utama pathlib:

- Membuat path ke file atau folder.
- Mengecek apakah file/folder ada.
- Membaca isi file.
- Menulis teks ke dalam file.
- Membuat folder baru.

### 7.2.3 pytest

pytest adalah library eksternal untuk melakukan pengujian otomatis di Python. Library ini sangat populer di kalangan pengembang karena mudah digunakan, ringkas, dan fleksibel. Dalam pemrograman, kita sering membuat fungsi-fungsi untuk menyelesaikan masalah tertentu. Namun, bagaimana memastikan fungsi itu selalu bekerja sesuai harapan? Dengan pengujian otomatis, kita bisa mendeteksi bug sejak dini.

## 7.3 Kegiatan Praktikum

### 7.3.1 Kegiatan Praktikum 1 : Implementasi pathlib

1. Buat sebuah file program baru, kemudian tuliskan kode berikut ini

```
1. from pathlib import Path
2.
3. class FileManager:
4.     def __init__(self, base_dir: str):
5.         self.base_dir = Path(base_dir)
6.         self.base_dir.mkdir(exist_ok=True)
7.
8.     def create_file(self, filename: str, content: str):
9.         file_path = self.base_dir / filename
10.        file_path.write_text(content, encoding="utf-8")
11.        return file_path
12.
13.    def append_to_file(self, filename: str, content: str):
14.        file_path = self.base_dir / filename
15.        with file_path.open("a", encoding="utf-8") as f:
16.            f.write(content)
17.
18.    def read_file(self, filename: str) -> str:
```

```
        file_path = self.base_dir / filename
        return file_path.read_text(encoding="utf-8")

    def list_files(self):
        return [f.name for f in self.base_dir.iterdir() if f.is_file()]

    def move_to_archive(self, filename: str):
        archive_dir = self.base_dir / "arsip"
        archive_dir.mkdir(exist_ok=True)
        file_path = self.base_dir / filename
        file_path.rename(archive_dir / file_path.name)


if __name__ == "__main__":
    manager = FileManager("praktikum_data")

    manager.create_file("catatan.txt", "Ini adalah catatan pertama.\n")
    manager.append_to_file("catatan.txt", "Baris kedua ditambahkan.\n")
    manager.append_to_file("catatan.txt", "Baris ketiga ditambahkan.\n")

    print("Isi file catatan.txt:")
    print(manager.read_file("catatan.txt"))

    print("\nDaftar file dalam folder:")
    for filename in manager.list_files():
        print("-", filename)

    manager.move_to_archive("catatan.txt")
    print("\nFile catatan.txt dipindahkan ke folder arsip.")
```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan 1 ini

### 7.3.2 Kegiatan Praktikum 2 : Implementasi pytest

1. Buka terminal pada VSCODE kemudian ketikkan kode berikut untuk membuat virtual environment

```
# membuat virtual environment
python -m venv venv

# mengaktifkan virtual environment di Windows
venv\Scripts\activate

```

2. Instalasi pytest menggunakan pip

```
pip install pytest

```

3. Buat file bernama calculator.py, isikan dengan kode berikut

```
class Calculator:
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

    def multiply(self, a, b):
        return a * b

    def divide(self, a, b):
        if b == 0:
            raise ValueError("Tidak bisa membagi dengan nol")
        return a / b
```

4. Buat file bernama test_calculator.py, pastikan file ini berada dalam 1 folder yang sama. isikan dengan kode berikut.

```
import pytest
from calculator import Calculator

def test_add():
    calc = Calculator()
    assert calc.add(2, 3) == 5

def test_subtract():
    calc = Calculator()
    assert calc.subtract(5, 3) == 2

def test_multiply():
    calc = Calculator()
    assert calc.multiply(4, 3) == 12

def test_divide():
    calc = Calculator()
```

```
    assert calc.divide(10, 2) == 5

def test_divide_by_zero():
    calc = Calculator()
    with pytest.raises(ValueError, match="Tidak bisa membag
i dengan nol"):
        calc.divide(5, 0)
```

5. Buka terminal kemudian jalankan perintah berikut

```
pytest

```

6. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan 2 ini

### 7.3.3 Kegiatan Praktikum 3 : Implementasi pytest bagian 2

1. Buat folder baru bernama string, buat virtual environment seperti pada Kegiatan 2.
2. Buat file bernama string_processor.py. Isikan dengan kode berikut

```
class StringProcessor:
    def reverse(self, s: str) -> str:
        return s[::-1]

    def is_palindrome(self, s: str) -> bool:
        cleaned = s.replace(" ", "").lower()
        return cleaned == cleaned[::-1]

    def capitalize_words(self, s: str) -> str:
        return " ".join(word.capitalize() for word in s.spl
it())

    def is_valid_email(self, email: str) -> bool:
        return "@" in email and "." in email.split("@")[-1]
```

3. Buat file bernama test_ string_processor.py, isikan dengan kode berikut

```
import pytest
from string_processor import StringProcessor

def test_reverse():
    sp = StringProcessor()
    assert sp.reverse("python") == "nohtyp"

def test_is_palindrome():
```

```
    sp = StringProcessor()
    assert sp.is_palindrome("kasur rusak") is True
    assert sp.is_palindrome("python") is False

def test_capitalize_words():
    sp = StringProcessor()
    assert sp.capitalize_words("belajar python pytest") ==
  "Belajar Python Pytest"

def test_is_valid_email():
    sp = StringProcessor()
    assert sp.is_valid_email("user@example.com") is True
    assert sp.is_valid_email("user@domain") is False
```

4. Buka terminal kemudian jalankan perintah berikut

```
pytest

```

5. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan 3 ini

## 7.4 Tugas

1. Apa perbedaan utama antara library bawaan Python dan library eksternal?
2. Mengapa kita tidak sebaiknya menulis semua kode dari nol?
3. Apa manfaat utama menulis pengujian dengan pytest?
4. Menurut Anda, apakah semua fungsi perlu diuji dengan pytest? Mengapa?