LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 7

2411102441250

Malik Sabarullah Akbar

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : itsmalik57@gmail.com

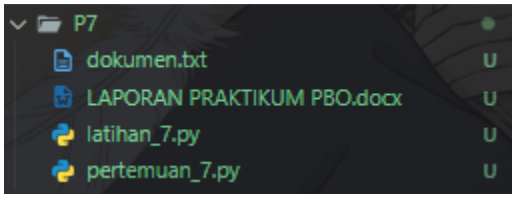
Menggunakan Pustaka Standar dalam Program OOP

Pustaka (library) dalam bahasa pemrograman adalah kumpulan kode yang sudah ditulis sebelumnya untuk menyediakan fungsionalitas tertentu, seperti fungsi atau kelas, yang dapat digunakan kembali oleh pengembang. Tujuannya adalah untuk menyederhanakan dan mempercepat proses pengembangan, karena pengembang tidak perlu menulis kode yang sama berulang kali untuk tugas-tugas umum.

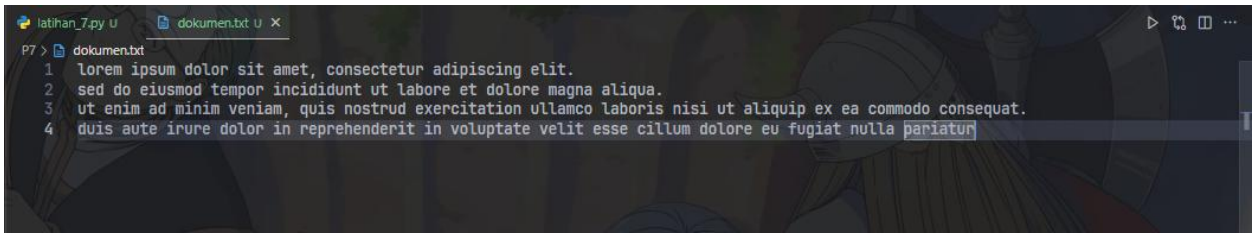
Studi Kasus - Analisis File Sederhana

Studi kasus kali ini adalah membuat sebuah alat sederhana yang dapat memberikan informasi dasar tentang sebuah file. Anda perlu menggabungkan pengetahuan tentang OOP dengan pustaka os dan datetime.

1. **Membuat file Latihan_7.py dan dokumen.txt di folder yang sama**

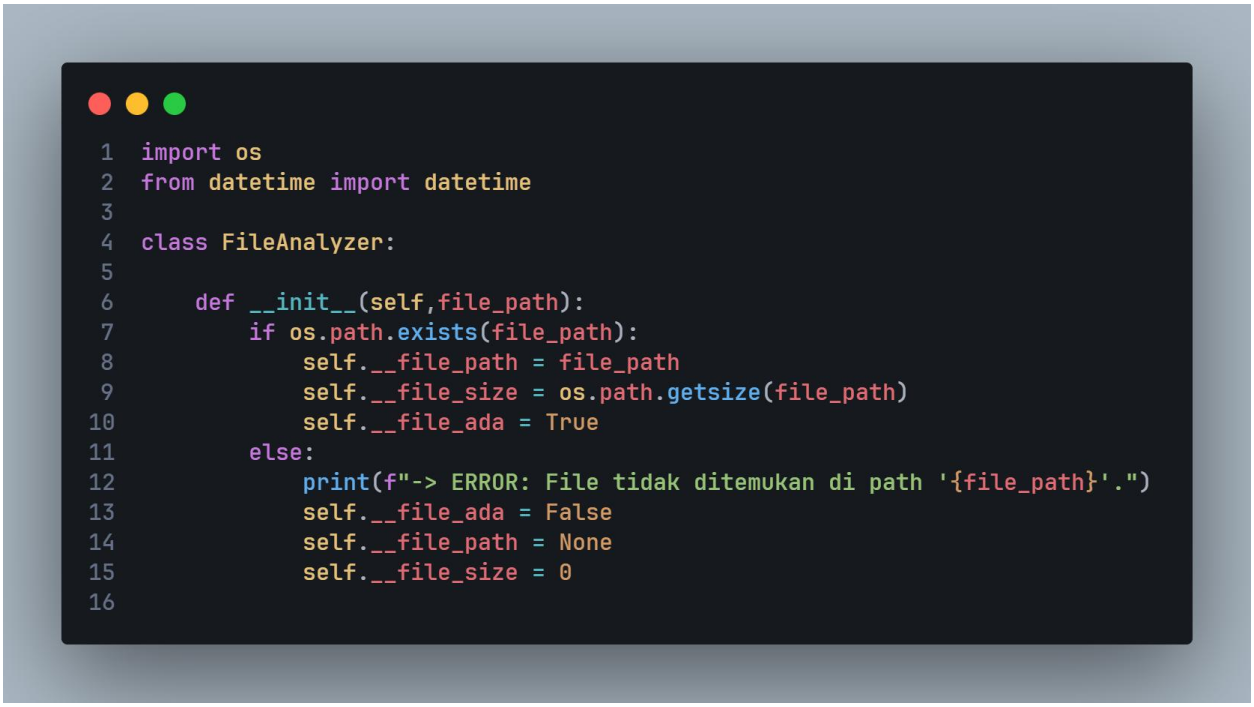


isi file dokumen.txt:



2. **Buat sebuah Class bernama FileAnalyzer.**
   a. **Constructor __init__(self, file_path) harus menerima path ke sebuah file (misalnya,"dokumen.txt").**
   b. **Di dalam constructor, gunakan pustaka os untuk memeriksa apakah file tersebut ada (os.path.exists()).**
      - **Jika file ada, simpan path-nya dan ukuran filenya (os.path.getsize()) dalam atribut privat.**
      - **Jika file tidak ada, cetak pesan error dan buat atribut penanda, misalnya**

self.__file_ada = False.



```python
import os
from datetime import datetime

class FileAnalyzer:

    def __init__(self,file_path):
        if os.path.exists(file_path):
            self.__file_path = file_path
            self.__file_size = os.path.getsize(file_path)
            self.__file_ada = True
        else:
            print(f"-> ERROR: File tidak ditemukan di path '{file_path}'.")
            self.__file_ada = False
            self.__file_path = None
            self.__file_size = 0

```

Di bagian constructor terdapat operasi perbandingan, jika file tersebut ditemukan maka akan dibuatkan 3 atribut private *__file_path = berisikan path file, __file_size = mengambil size file, __file_ada = bernilai true jika file ada*. Jika file tidak ditemukan maka menampilkan output error.

3. **get_file_size(self, unit="bytes"): Method ini harus mengembalikan ukuran file. Tambahkan logika untuk mengonversi ukuran ke Kilobytes (KB) jika parameter unit diisi "KB". (Petunjuk: 1 KB = 1024 Bytes).**

```python
def get_file_size(self, unit="bytes"):
        if not self.__file_ada:
            return "File tidak tersedia."

        if unit == "bytes":
            return f"{self.__file_size} bytes"
        elif unit == "KB":
            return f"{self.__file_size / 1024} KB"
        elif unit == "MB":
            return f"{self.__file_size / (1024 * 1024)} MB"
        else:
            return "Unit tidak dikenali. Gunakan 'bytes', 'KB', atau 'MB'."

```

Method ini digunakan untuk menampilkan output size dengan unit bytes, KB, dan MB agar mudah dibaca.

4. **get_modification_time(self): Method ini harus menggunakan os.path.getmtime() untuk mendapatkan waktu modifikasi terakhir file. Kemudian, gunakan pustaka datetime (datetime.fromtimestamp()) untuk mengubahnya menjadi format tanggal dan waktu yang dapat dibaca manusia.**

```python
def get_modification_time(self):
        if not self.__file_ada:
            return "File tidak tersedia."

        mod_time = os.path.getmtime(self.__file_path)
        return f"Waktu modifikasi terakhir: {datetime.fromtimestamp(mod_time)}"

```

Method ini digunakan untuk merubah atau memodifikasi waktu pembuatan file agar mudah di baca oleh manusia.

5. **analyze(self): Method ini akan menjadi method utama. Ia akan memanggil dua method di atas dan mencetak laporan lengkap tentang file tersebut (nama file,**

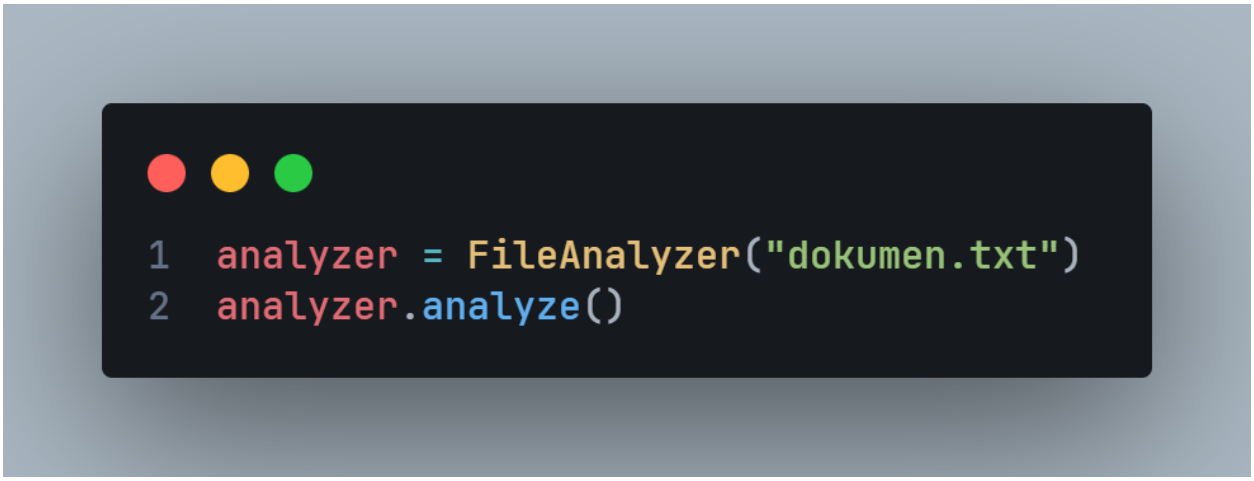
apakah ada, ukuran dalam KB, dan waktu modifikasi terakhir) dengan format yang rapi. Jika file tidak ada, method ini hanya mencetak pesan bahwa file tidak dapat dianalisis.

```
def analyze(self):
        if self.__file_ada:
            print(f"File Path: {self.__file_path}")
            print(f"File Exists: {self.__file_ada}")
            print(f"File Size: {self.get_file_size("KB")}")
            print(f"Last Modified: {self.get_modification_time()}")

```

Sebuah method yang digunakan untuk menampilkan output dengan lengkap terkait file yang ingin di analis.

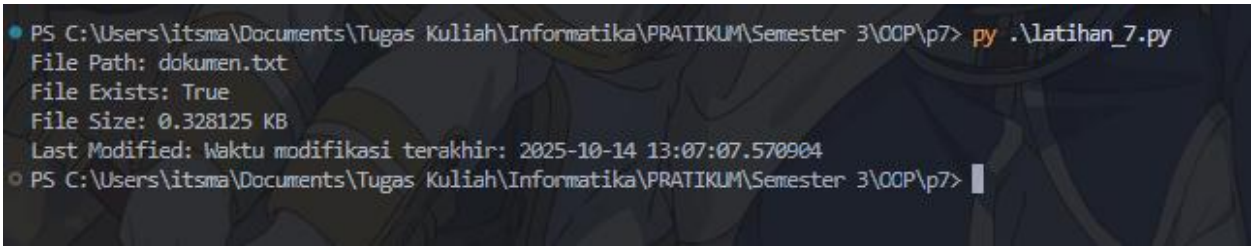
6. **Pembuatan object dan hasil output**



```
analyzer = FileAnalyzer("dokumen.txt")
analyzer.analyze()
```

Membuat object analyzer dan file target-nya dokumen.txt.

Output:



Kesimpulan

Pada pertemuan ke 7 ini, saya telah mempelajari cara penggunaan library datetime dan os pada python. Mengapa menggunakan library lebih baik dari pada menulis serangkaian fungsi terpisah di luar kelas?

Karena penggunaan library lebih mudah digunakan, seperti penggunaan datetime yang hanya melakukan import dan memanggil method yang dibutuhkan, hal tersebut lebih menghemat waktu dan tenaga pada project atau aplikasi yang besar. Jika library yang ingin kita gunakan tidak disediakan oleh python, maka kita bisa menginstall-nya di terminal menggunakan pip (package installer python).