

**LAPORAN PRAKTIKUM PEMOGRAMAN  
BERORIENTASI OBJEK (PBO)  
PRAKTIKUM 7**



**2411102441249  
HERVINO ISLAMI FASHA**

**FAKULTAS SAINS DAN TEKNOLOGI  
PROGRAM STUDI S1 TEKNIK INFORMATIKA  
UNIVERSITAS MUHAMMADIYAH KALIMANTAN  
TIMUR**

## LATIHAN PRAKTIKUM

### 1. Membuat Code KalkulatorLingkaran

```

● ● ●

1 import math #Import pustaka math
2
3
4 class KalkulatorLingkaran:
5     def __init__(self, radius):
6         self.__radius = 0
7         self.set_radius(radius)
8         print(f"Objek lingkaran dengan radius {self.__radius} dibuat.")
9
10    def set_radius(self, radius):
11        if radius > 0:
12            self.__radius = radius
13        else:
14            print("Error: Radius harus lebih besar dari 0")
15            self.__radius = 1 #nilai default jika input salah
16
17    def hitung_luas(self):
18        #Menggunakan konstanta pi dari pustaka math
19        luas = math.pi * (self.__radius ** 2)
20        return luas
21
22    def hitung_keliling(self):
23        #Menggunakan Konstanta pi lagi
24        keliling = 2 * math.pi * self.__radius
25        return keliling
26
27
28 #--- BAGIAN UTAMA PROGRAM ---
29 lingkaran_1 = KalkulatorLingkaran(7)
30 luas_lingkaran = lingkaran_1.hitung_luas()
31 keliling_lingkaran = lingkaran_1.hitung_keliling()
32
33 print(f"\nRadius: 7")
34 print(f"Luas Lingkaran: {luas_lingkaran:.2f}") # format 2 dibelakang koma
35 print(f"Keliling Lingkaran: {keliling_lingkaran:.2f}")

```

- Program mengimpor modul math untuk menggunakan konstanta  $\pi$  (pi).
- Membuat class berisi:
  - a `__init__`: konstruktor, menerima nilai radius dan memanggil `set_radius()`.

- b set\_radius(): memvalidasi radius. Jika radius  $\leq 0$ , muncul pesan error dan radius diatur ke 1.
  - c hitung\_luas(): menghitung luas dengan rumus  $\pi \times r^2$ .
  - d hitung\_keliling(): menghitung keliling dengan rumus  $2 \times \pi \times r$ .
- Bagian utama program
    - a Membuat objek lingkaran\_1 dengan radius 7.
    - b Memanggil hitung\_luas() dan hitung\_keliling() untuk mendapatkan hasil.
    - c Mencetak radius, luas, dan keliling dengan format dua angka di belakang koma.

Output nya adalah:

```
D:\smt 3\PBO\P7>C:/Python313/python.exe "d:/smt 3/PBO/P7/pertemuan_7.py"
Objek lingkaran dengan radius 7 dibuat.

Radius: 7
Luas Lingkaran: 153.94
Keliling Lingkaran: 43.98
```

## 2. Integrasi Pustaka DateTime

```

● ● ●

1  from datetime import datetime
2
3  class LogPesan:
4      def __init__(self, pengirim, isi_pesan):
5          self.__pengirim = pengirim
6          self.__isi_pesan = isi_pesan
7          #Secara otomatis mendapatkan waktu saat ini ketika objek dibuat
8          self.__timestamp = datetime.now()
9
10     def tampilkan_log(self):
11         waktu_terformat = self.__timestamp.strftime("%d %B %Y, Pukul %H:%M:%S")
12         print("--- Log Pesan Masuk ---")
13         print(f"Pengirim : {self.__pengirim}")
14         print(f"Waktu : {waktu_terformat}")
15         print(f"pesan : {self.__isi_pesan}")
16
17     #--- BAGIAN UTAMA PROGRAM ---
18 pesan_1 = LogPesan("Admin", "Server akan segera di restart untuk maintenance. ")
19 pesan_1.tampilkan_log()
20
21 #tunggu beberapa detik untuk pesan Lain
22 #(untuk simulasi, kita bisa tambahkan time.sleep jika diinginkan)
23 pesan_2 = LogPesan("User 1", "Pekerjaan saya sudah disimpan silahkan restart.")
24 pesan_2.tampilkan_log()

```

- Import datetime – untuk mendapatkan waktu saat pesan dibuat.
- Class LogPesan
  - a \_\_init\_\_(): menyimpan nama pengirim, isi pesan, dan otomatis mencatat waktu sekarang (datetime.now()).
  - b tampilkan\_log(): menampilkan informasi log dengan format waktu yang mudah dibaca, seperti “14 Oktober 2025, Pukul 22:10:30”.
- Bagian Utama Program
  - a Membuat dua objek log:
    - pesan\_1 dari Admin dengan pesan “Server akan segera di restart untuk maintenance.”
    - pesan\_2 dari User 1 dengan pesan “Pekerjaan saya sudah disimpan silahkan restart.”
  - b Masing-masing pesan ditampilkan dengan waktu saat dibuat.

Maka outputnya:

```
D:\smt 3\PB0\P7>C:/Python313/python.exe "d:/smt 3/PB0/P7/pertemuan_7.py"
--- Log Pesan Masuk ---
Pengirim : Admin
Waktu : 14 October 2025, Pukul 22:23:25
Pesan : Server akan segera di restart untuk maintenance.

--- Log Pesan Masuk ---
Pengirim : User 1
Waktu : 14 October 2025, Pukul 22:23:25
Pesan : Pekerjaan saya sudah disimpan silahkan restart.
```

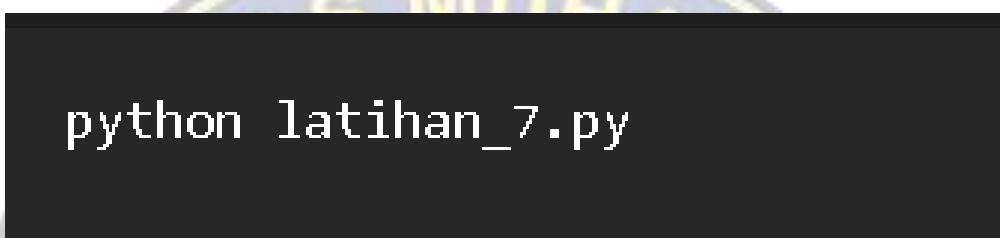


## TUGAS STUDI KASUS: ANALISIS FILE SEDERHANA

### 1. Membuat File

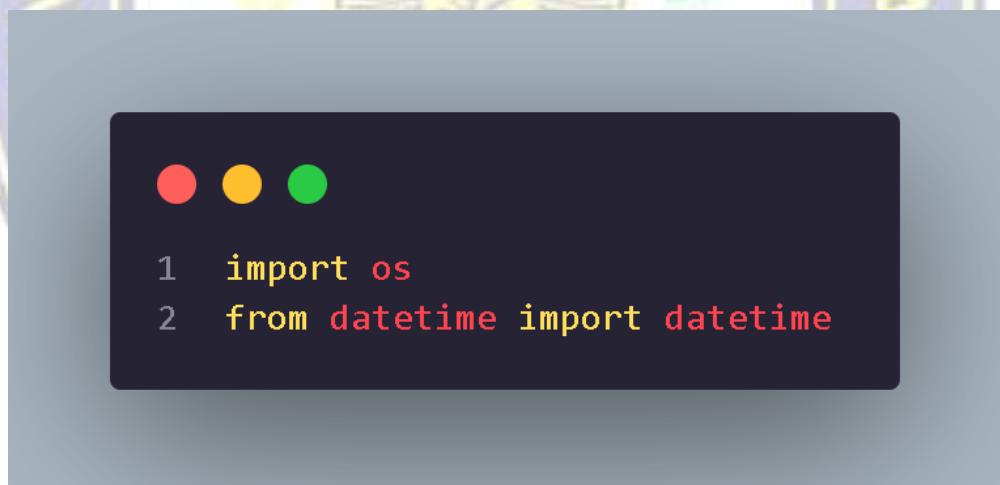


- Buat file bernama Latihan\_7.py dan Dokument.txt



- Isi dokument.txt dengan python Latihan\_7.py

### 2. Import OS dan DateTime



- OS digunakan untuk memeriksa keberadaan file, ukuran, dan waktu modifikasi.
- DateTime digunakan untuk megubah waktu modifikasi ke format tanggal dan jam yang mudah dibaca.

### 3. Membuat Class FileAnalyzer

```

1  class FileAnalyzer:
2      def __init__(self, file_path):
3          self.__file_path = file_path
4          if os.path.exists(file_path): # mengecek apakah file ada.
5              self.__file_ada = True
6              self.__file_size = os.path.getsize(file_path) # mengambil ukuran file (byte).
7          else:
8              print(f"Error: File '{file_path}' tidak ditemukan.")
9              self.__file_ada = False
10
11     def get_file_size(self, unit="bytes"):
12         if not self.__file_ada:
13             return None
14         if unit.lower() == "kb":
15             return round(self.__file_size / 1024, 2)
16         return self.__file_size
17
18     def get_modification_time(self):
19         if not self.__file_ada:
20             return None
21         timestamp = os.path.getmtime(self.__file_path) #mengambil waktu modifikasi terakhir.
22         return datetime.fromtimestamp(timestamp).strftime("%Y-%m-%d %H:%M:%S") #mengubah waktu menjadi format terbaik.
23
24     def analyze(self):
25         if not self.__file_ada:
26             print(f"File '{self.__file_path}' tidak dapat dianalisis karena tidak ditemukan.")
27             return
28
29         print("== Laporan Analisis File ==")
30         print(f"Nama File: {self.__file_path}")
31         print(f"Ukuran: {self.get_file_size('KB')} KB")
32         print(f"Waktu Modifikasi Terakhir: {self.get_modification_time()}")
33         print("=====")

```

- \_\_init\_\_ (constructor)
  - a. Menerima parameter `file_path` (lokasi file).
  - b. Mengecek apakah file ada dengan `os.path.exists()`.
  - c. Jika ada, menyimpan ukuran file dengan `os.path.getsize()`.
  - d. Jika tidak ada, menampilkan pesan error dan menandai file tidak ditemukan.
- Get\_file\_size()
  - a. Mengembalikan ukuran file dalam **bytes** atau **KB**.
  - b. Jika file tidak ditemukan, fungsi mengembalikan `None`.
- get\_modification\_time()
  - a. Mengambil waktu modifikasi terakhir file dengan `os.path.getmtime()`.
  - b. Mengubah waktu tersebut ke format yang mudah dibaca menggunakan `datetime.fromtimestamp()`.
- Analyze()
  - a. Menampilkan laporan lengkap berisi:
    - Nama file
    - Ukuran file (dalam KB)
    - Waktu modifikasi terakhir

- b. Jika file tidak ada, menampilkan pesan bahwa file tidak bisa dianalisis.

#### 4. Bagian Utama Program



```
● ● ●

1 # Bagian utama program
2 if __name__ == "__main__":
3     file1 = FileAnalyzer("dokument.txt")
4     file1.analyze() #mencetak hasil analisis dalam format rapi.
5
6     print() # spasi antar output
7
8     file2 = FileAnalyzer("file_Hilang.txt")
9     file2.analyze()
10
```

- Membuat objek file1 untuk dokument.txt dan menampilkan hasil analisis.
- Membuat objek file2 untuk file\_khayalan.txt guna menguji penanganan error.

#### 5. Output

Run code di terminal:

```
D:\smt 3\PB0\P7>C:/Python313/python.exe "d:/smt 3/PB0/P7/latihan_7.py"
== Laporan Analisis File ==
Nama File: dokument.txt
Ukuran: 0.02 KB
Waktu Modifikasi Terakhir: 2025-10-14 22:11:11
=====

Error: File 'file_Hilang.txt' tidak ditemukan.
File 'file_Hilang.txt' tidak dapat dianalisis karena tidak ditemukan.
```

## REFLEKSI

Menyatukan pustaka standar ke dalam sebuah kelas membuat tata kode menjadi lebih teratur dan lebih gampang untuk dikelola. Setiap objek dari kelas dapat menyimpan informasi dan tingkah laku masing-masing, sehingga lebih efektif dan luwes dibandingkan dengan fungsi yang berdiri sendiri. Metode ini juga mempermudah perbaikan dan pemeliharaan aplikasi di kemudian hari.

