LAPORAN PRAKTIKUM III
PRAKTIKUM STRUKTUR DATA

Materi Pertemuan-2

DISUSUN OLEH:
Yohanes Doly Jaya Laneta Tambunan
25071104585

DOSEN PENGAMPU:
Reny Fitri Yani, S.T., M.T.

ASISTEN PRAKTIKUM:
Rizkillah Ramanda Sinyo
Muhammad Abidillah

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS RIAU
PEKANBARU
2026

DAFTAR ISI

BAB I

PELAKSANAAN PRAKTIKUM

1.1 Python OOP

Python adalah bahasa berorientasi objek, yang memungkinkan Anda untuk menyusun kode menggunakan kelas dan objek untuk pengorganisasian dan penggunaan kembali yang lebih baik. Kelas dan objek adalah dua konsep inti dalam pemrograman berorientasi objek. Sebuah kelas mendefinisikan seperti apa seharusnya sebuah objek, dan sebuah objek dibuat berdasarkan kelas tersebut. Misalnya:

Class	Objects
Fruit	Apple, Banana, Mango
Car	Volvo, Audi, Toyota

1.2 Python Classes/Object

Hampir semua hal di Python adalah objek, dengan properti dan metodenya masing-masing. Sebuah Class itu seperti konstruktor objek, atau "cetak biru" untuk membuat objek.

```
class MyClass:
  x = 5
```

Sekarang kita dapat menggunakan kelas bernama MyClass untuk membuat objek:

```
p1 = MyClass()
print(p1.x)
```

Anda dapat membuat beberapa objek dari kelas yang sama:

```
class MyClass:
  x = 5

p1 = MyClass()
p2 = MyClass()
p3 = MyClass()

print(p1.x)
print(p2.x)
print(p3.x)
```

Class adalah definisi tidak boleh kosong, tetapi jika karena suatu alasan Anda memiliki class definisi tanpa konten, tambahkan pass pernyataan tersebut untuk menghindari kesalahan.

```
class Person:
  pass
```

1.3 Metode __init__ Python

Semua kelas memiliki metode bawaan yang disebut __init__() yang selalu dieksekusi ketika kelas tersebut diinisialisasi. Metode __init__() digunakan untuk menetapkan nilai pada properti objek, atau untuk melakukan operasi yang diperlukan saat objek sedang dibuat.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("Emil", 36)

print(p1.name)
print(p1.age)
```

1.4 Self Parameter

Parameter self digunakan untuk mengakses properti dan metode yang dimiliki oleh kelas tersebut.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def greet(self):
    print("Hello, my name is " +
self.name)

p1 = Person("Emil", 25)
p1.greet()
```

Self dapat mengakses properti apa pun dari kelas tersebut:

```
class Car:
  def __init__(self, brand, model, year):
    self.brand = brand
    self.model = model
    self.year = year

  def display_info(self):
    print(f"{self.year} {self.brand}
{self.model}")

car1 = Car("Toyota", "Corolla", 2020)
car1.display_info()
```

1.5 Class Properties

Properti adalah variabel yang dimiliki oleh suatu kelas. Variabel ini menyimpan data untuk setiap objek yang dibuat dari kelas tersebut. Variabel dapat mengakses properti objek menggunakan notasi titik:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("Emil", 36)

print(p1.name)
print(p1.age)
```

KIta dapat memodifikasi nilai properti pada objek:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " +
self.name)

p1 = Person("John", 36)

p1.age = 40

print(p1.age)
```

Kita dapat menghapus properti dari objek menggunakan del kata kunci:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " +
self.name)

p1 = Person("Linus", 18)

del p1.age

print(p1.name)
print(p1.age)
```

Anda dapat menambahkan properti baru ke objek yang sudah ada:

```
class Person:
  def __init__(self, name):
    self.name = name

p1 = Person("Tobias")

p1.age = 25
p1.city = "Oslo"

print(p1.name)
print(p1.age)
print(p1.city)
```

1.6 Class Methods

Metode adalah fungsi yang dimiliki oleh suatu kelas. Metode mendefinisikan perilaku objek yang dibuat dari kelas tersebut.

```
class Person:
  def __init__(self, name):
    self.name = name

  def greet(self):
    print("Hello, my name is " +
self.name)

p1 = Person("Emil")
p1.greet()
```

Metode dapat menerima parameter seperti halnya fungsi biasa menggunakan return:

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

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

calc = Calculator()
print(calc.add(5, 3))
print(calc.multiply(4, 7))
```