

LAPORAN PRAKTIKUM

STRUKTUR DATA

Pertemuan 3



Disusun oleh :

Nama : Fildza Khusaini
NIM : 25071104589
Dosen : Reny Fitri Yani, S.T., M.T
Asisten : 1. Akhlaqul Muhammad Fadwa (2307112834)
 2. Rizkillah Ramanda Sinyo (2307126144)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS RIAU

PEKANBARU

GENAP 2025/2026

Python OOP

1. Apa itu OOP?

OOP adalah singkatan dari Object Oriented Programming (Pemrograman Berorientasi Objek) yang merupakan orientasi pemrograman yang menggunakan konsep objek dan kelas dalam pemrograman. Dalam OOP, setiap bagian program direpresentasikan sebagai objek yang memiliki atribut dan method (perilaku). OOP diciptakan agar memberikan struktur yang jelas pada program, memudahkan pemeliharaan kode, mempercepat pengembangan, dan membantu menjaga agar kode tetap DRY (Don't repeat Yourself) atau tidak menulis kode yang sama berulang kali.

2. Class dan object

Class adalah blueprint atau cetak biru untuk membuat objek yang berisi atribut dan method. Sedangkan objek adalah wujud nyata dari cetak biru tersebut. Sebuah class medefinisikan seperti apa seharusnya sebuah objek, dan sebuah objek dibuat berdasarkan class tersebut. Contohnya, class Buah. Maka objectnya adalah apel, pisang,

```
#contoh membuat class

class Buah: #nama class menggunakan pascal case (Buah)
    manis = "pepaya" #manis dan asam merupakan atribut class
    asam = "jeruk"

#membuat objek
b1 = Buah() #b1 adalah variabel atau objek yang menampung class
print(b1.manis)

#output
Pisang
```

jeruk, dan sejenisnya.

- a) Kamu bisa membuat beberapa object dari kelas yang sama. Dan untuk menghapus object, kamu dapat menggunakan keyword del.

```
#menghapus objek dengan keyword del
class Band:
    def __init__(self, nama_grup):
        self.nama = nama_grup

    def perkenalan(self):
        print("Halo, kami " + self.nama)

grup1 = Band("Aespa")
```

```
del grup1
print(grup1)

#output
Traceback (most recent call last):
  File "c:\Users\USER\Praktikum-StrukDat-B-2026\pertemuan-3\classes_objects.py", line 21, in <module>
    print(grup1)
    ^^^^^^
NameError: name 'grup1' is not defined
```

- b) Jika kamu tidak ingin mengisi definisi class terlebih dahulu, gunakan pass statement untuk menghindari terjadinya error.

```
#menggunakan pass untuk menghindari eror saat definisi class kosong

class Rumah:
    pass
```

3. `__init__()` Method

Semua class memiliki method bawaan yang disebut `__init__()`, yang selalu dieksekusi Ketika class diinisialisasi. Fungsi utamanya adalah untuk menginisialisasi atribut object, menetapkan nilai awal, atau melakukan operasi yang diperlukan saat objek sedang dibuat.

```
class Buku:

    def __init__(self, nama, kode): #method __init__() digunakan untuk
        menetapkan nilai untuk nama dan kode

        self.nama = nama
        self.kode = kode

buku1 = Buku("Novel", 119)

print(buku1.nama)
print(buku1.kode)

#output
Novel
119
```

- a) Tanpa menggunakan `__init__()`, atribut untuk setiap objek harus diatur secara manual.

```
class Buku:  
    pass  
  
buku1 = Buku()  
buku1.nama = "Neo"  
buku1.kode = 127  
  
print(buku1.nama)  
print(buku1.kode)  
  
#output  
Neo  
127
```

- b) Di dalam method `__init__()`, kamu dapat mengatur nilai default untuk parameter.

```
#nilai default di __init__()  
class Biodata:  
    def __init__(self, nama, kelas = "TI B"): #TI B adalah nilai  
default untuk parameter kelas  
        self.nama = nama  
        self.kelas = kelas  
  
mhs1 = Biodata("Haechan")  
mhs2 = Biodata("Jeno", "TI A")  
  
print(mhs1.nama, mhs1.kelas)  
print(mhs2.nama, mhs2.kelas)  
  
#metode __init__ juga dapat memiliki banyak parameter  
  
#output  
Haechan TI B  
Jeno TI A
```

4. Self Parameter

Parameter `self` adalah referensi ke instance (objek) class saat ini yang digunakan untuk mengakses atribut dan method yang dimiliki oleh class. Parameter `self` harus menjadi parameter pertama dari method manapun di dalam kelas.

```

#parameter self digunakan untuk mengakses atribut class
class Buku:
    def __init__(self, nama, kode): #parameternya ada self, nama, kode
        self.nama = nama #self.nama itu atribut instance
        self.kode = kode

    def pinjam(self): #self merujuk ke objek yang memanggil method ini
        print(f"Buku {self.nama} telah dipinjam")

buku1 = Buku("Resonance", 2320) #disini, self di __init__() merujuk ke
objek buku1

buku1.pinjam()

#output
Buku Resonance telah dipinjam

```

- a) Self bisa diganti menggunakan kata lain untuk membuatnya, tetapi harus menjadi parameter pertama dari method apapun di dalam class tersebut.

```

#mengganti self dengan kata ini dan abc
class Tiket:
    def __init__(ini, nama, tujuan):
        ini.nama = nama
        ini.tujuan = tujuan

    def pesan(abc):
        print(f"Anda memesan tiket dengan tujuan {abc.tujuan}")

tiket1 = Tiket("Jisung", "Pekanbaru")
tiket1.pesan()

#output
Anda memesan tiket dengan tujuan Pekanbaru

```

- b) Selain mengakses atribut dengan menggunakan self, kamu juga bisa memanggil method lain di dalam class menggunakan self.

```

class Band:
    def __init__(self, nama_grup):
        self.nama = nama_grup

    def perkenalan(self):
        return "Halo, kami " + self.nama

    def sapa(self):
        sapaan = self.perkenalan() #memanggil method lain di dalam
        class menggunakan self
        print(sapaan + "! Salam kenal! " )

grup1 = Band("Aespa")
grup1.sapa()

#output
Halo, kami Aespa! Salam kenal!

```

5. Class Properties

Properti atau atribut adalah variabel di dalam class yang menyimpan data atau property objek. Properti dapat diakses dengan menggunakan notasi titik.

```

class Donat:
    def __init__(self, varian, ukuran):
        self.varian = varian
        self.ukuran = ukuran

donat1 = Donat("donat coklat", "L")

print(donat1.varian)

#output
donat coklat

```

- a) Kamu bisa mengubah nilai properti pada objek.

```
#output  
Pilihan pertama: hitam  
Pilihan terakhir: putih
```

```
#mengubah nilai properti pada objek  
class Sepatu:  
    def __init__(self, warna, ukuran):  
        self.warna = warna  
        self.ukuran = ukuran  
  
    def beli(self):  
        print(f"Sepatu berwarna {self.warna} dengan ukuran  
{self.ukuran}")  
  
s1 = Sepatu("hitam", 40)  
print("Pilihan pertama:", s1.warna)  
  
s1.warna = "putih"  
print("Pilihan terakhir:", s1.warna)  
  
print(tiket1.nama)  
print(tiket1.tujuan) #akan menimbulkan error
```

- b) Untuk menghapus properti, gunakan keyword del
- c) Property class vs property object

Properti class adalah atribut yang dimiliki class dan dibagikan ke seluruh instance. Atribut ini ditulis langsung di dalam class, tetapi di luar method. Sedangkan properti object adalah atribut yang didefinisikan di dalam `__init__()` dan dimiliki setiap instance secara terpisah.

```
class Hewan:  
    jenis = "peliharaan" #properti class  
  
class Hewan:  
    jenis = "peliharaan"  
  
    def __init__(self, nama):  
        self.nama = nama  
  
    h1 = Hewan("kucing")  
    h2 = Hewan("kelinci")  
  
    Hewan.jenis = "liar"  
  
    print(h1.nama, h1.jenis)  
    print(h2.nama, h2.jenis)  
  
#output  
kucing liar  
kelinci liar
```

d) Memodifikasi properti class menggunakan notasi titik.

e) Menambahkan property baru ke objek yang sudah ada.

```
class Bunga:  
    def __init__(self, nama):  
        self.nama = nama  
  
    b1 = Bunga("mawar")  
  
    b1.warna = "merah"  
    b1.jumlah = 5  
  
    print(b1.jumlah, b1.nama, b1.warna)  
  
#output  
5 mawar merah
```

6. Class methods

Method adalah fungsi yang dimiliki oleh sebuah class. Method mendefinisikan perilaku objek yang dibuat dari class tersebut.

```
#contoh
class Band:
    def __init__(self, nama_grup):
        self.nama = nama_grup

    def sapa(self): #ini adalah method
        print("Halo! To the world, yeogin NCT " + self.nama)

grup1 = Band("Dream")
grup1.sapa()

#output
Halo! To the world, yeogin NCT Dream
```

- a) Method dapat menerima parameter, sama halnya dengan fungsi biasa

```
#contoh
class Hitung:
    def tambah(self, a, b):
        return a + b

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

hasil = Hitung()
print(hasil.tambah(2, 3))
print(hasil.kali(4, 5))

#output
5
20
```

- b) Method dapat mengakses dan memodifikasi properti objek menggunakan self

```
#contoh method yang mengakses properti objek
class Tiket:
    def __init__(self, nama, tujuan):
        self.nama = nama
        self.tujuan = tujuan

    def pesan(self):
        return f"Anda telah memesan tiket atas nama {self.nama} dengan tujuan {self.tujuan}"

tiket1 = Tiket("Chenle", "Pekanbaru")
print(tiket1.pesan())

#output
Anda telah memesan tiket atas nama Chenle dengan tujuan Pekanbaru
```

c) Method yang memodifikasi property

```
#contoh method yang memodifikasi properti suatu objek
class Bunga:
    def __init__(self, nama, jumlah):
        self.nama = nama
        self.jumlah = jumlah

    def petik(self):
        self.jumlah += 1
        print(f"Petik bunga sebanyak {self.jumlah} tangkai")

b1 = Bunga("mawar", 5)
b1.petik()
b1.petik()

#output
Petik bunga sebanyak 6 tangkai
Petik bunga sebanyak 7 tangkai
```

d) __str__() method

Method ini adalah metode khusus yang menentukan apa yang dikembalikan saat objek dicetak.

```
#contoh jika tanpa method __str__()
class Nilai:
    def __init__(self, nama, skor):
        self.nama = nama
        self.skor = skor

a1 = Nilai("Kayla", 99)
print(a1)

#outputnya bukan isi objeknya, tetapi alamat memori.
<__main__.Nilai object at 0x000001F88B795E20>
```

```
#contoh jika menggunakan method __str__()
class Nilai:
    def __init__(self, nama, skor):
        self.nama = nama
        self.skor = skor

    def __str__(self):
        return f"{self.nama} ({self.skor})"

a1 = Nilai("Kayla", 99)
print(a1)

#output
Kayla (99)
```

e) Delete methods

untuk menghapus methods dari sebuah kelas, gunakan keyword del

```
class Bunga:
    def __init__(self, nama):
        self.nama = nama

    def petik(self):
        print(f"petik bunga {self.nama}")

b1 = Bunga("tulip")

del Bunga.petik

b1.petik() #ini akan memunculkan eror

#output
Traceback (most recent call last):
  File "c:\Users\USER\Praktikum-StrukDat-B-2026\pertemuan-3\tempCodeRunnerFile.py", line 12, in <module>
    b1.petik() #ini akan memunculkan eror
    ^^^^^^^^
AttributeError: 'Bunga' object has no attribute 'petik'
```

f) Sebuah class dapat memiliki beberapa method yang dapat bekerja bersama-sama.

```
class DramaList:
    def __init__(self, nama):
        self.nama = nama
        self.dramas = []

    def tambah_drama(self, drama):
        self.dramas.append(drama)
        print(f"Tambah drama baru: {drama}")

    def hapus_drama(self, drama):
        if drama in self.dramas:
            self.dramas.remove(drama)
            print(f"Hapus drama: {drama}")

    def tampilkan(self):
        print(f"List Drama '{self.nama}':")
        for drama in self.dramas:
            print(f"-{drama}")

d1 = DramaList("Belum ditonton")
d1.tambah_drama("Our Universe")
d1.tambah_drama("Taxi Drive 2")
d1.tampilkan()

#output
Tambah drama baru: Our Universe
Tambah drama baru: Taxi Drive 2
List Drama 'Belum ditonton':
-Our Universe
-Taxi Drive 2
```

Latihan Pertemuan 3

```
"""
buatlah sebuah class dengan
-minimal 3 atribut / property
-2 method
lalu buat 3 object dr class tsb
lalu ubahlah salah satu atribut dari objek tersebut
"""

class Absen:
    def __init__(self, nama, kelas, status = "hadir"):
        self.nama = nama
        self.kelas = kelas
        self.status = status

    def izin(self):
        self.status = "izin"
        print(f"{self.nama} {self.kelas} {self.status}")

    def sakit(self):
        self.status = "sakit"
        print(f"{self.nama} {self.kelas} {self.status}")

    def ubah_status(self, status_baru):
        self.status = status_baru

absen1 = Absen("fildza", "TI B")
print(absen1.nama, absen1.kelas, absen1.status)

absen2 = Absen("rania", "TI B")
absen2.izin()

absen3 = Absen("najwa", "TI B")
absen3.sakit()

#mengubah atribut status dari objek absen2
print(f"sebelum status diubah {absen2.status}")
absen2.ubah_status("alpa")
print(f"status sekarang {absen2.status}")

#output
fildza TI B hadir
rania TI B izin
najwa TI B sakit
sebelum status diubah izin
status sekarang alpa
```

Kode program di atas merupakan implementasi konsep class dalam python untuk mengelola data kehadiran mahasiswa. Terdapat tiga property yang dibuat di dalam `__init__()`, yaitu `self.nama`, `self.kelas`, dan `self.status` yang memiliki nilai default “hadir”. Class `Absen` ini memiliki tiga method, yaitu method `izin()` untuk mengubah status kehadiran menjadi izin, method `alpa()` untuk mengubah status kehadiran menjadi alpa, dan method `ubah_status()` untuk mengubah status lama menjadi status baru. Kemudian, dibuat tiga objek dari class tersebut, yaitu `absen1`, `absen2`, dan `absen3`.