

**PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK**  
**TUGAS 5 RESUME MODUL 1 SAMPAI 4**



Disusun oleh:

Inori Muira Sitanggang 121140020

Kelas RB

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**INSTITUT TEKNOLOGI SUMATERA**  
**LAMPUNG SELATAN 2023**

## MODUL 1

### 1. Pengenalan Bahasa Pemrograman

Bahasa Pemrograman Python dibuat pada akhir 1980-an oleh Guido van Rossum di Centrum Wiskunde & Informatica (CWI) di Belanda sebagai penerus bahasa ABC (yang terinspirasi oleh SETL). Bahasa pemrograman Python mampu menangani pengecualian dan berinteraksi dengan sistem operasi Amoeba yang implementasinya dimulai pada bulan Desember 1989. Python sendiri merupakan bahasa pemrograman yang banyak digunakan dalam aplikasi web, pengembangan perangkat lunak, ilmu data, dan juga pembelajaran mesin atau *machine learning*.

### 2. Dasar Pemrograman

#### #Sintaks Dasar

##### ➤ Statement

Semua perintah yang dapat dieksekusi pada Python disebut statement. statement dapat direpresentasikan pada baris baru atau dapat menggunakan backslash (\).

##### ➤ Baris dan Indentasi

Python tidak menggunakan kurung kurawal untuk mengelompokkan blok kode melainkan menggunakan spasi/tab.

#### # Variabel dan Tipe Data Primitif

Variabel berfungsi untuk menyimpan suatu nilai. Untuk mendeklarasikan variabel, maka dibutuhkan beberapa tipe data berikut:

Tipe Data	Contoh	Penjelasan
Boolean	<code>True</code> atau <code>False</code>	Menyatakan benar <code>True</code> yang bernilai <code>1</code> , atau salah <code>False</code> yang bernilai <code>0</code>
String	<code>"Ayo belajar Python"</code>	Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda <code>"</code> atau <code>'</code> )
Integer	<code>25</code> atau <code>1200</code>	Menyatakan bilangan bulat
Float	<code>3.14</code> atau <code>0.99</code>	Menyatakan bilangan yang mempunyai koma
Complex	<code>1 + 5j</code>	Menyatakan pasangan angka real dan imajiner
List	<code>['xyz', 786, 2.23]</code>	Data urutan yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah
Tuple	<code>('xyz', 786, 2.23)</code>	Data urutan yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah
Dictionary	<code>{'name': 'adi', 'id': 1}</code>	Data urutan yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai

#### # Operator

##### ➤ Operator Aritmatika

Operator aritmatika adalah operator yang digunakan untuk melakukan operasi matematika, seperti penjumlahan, pengurangan, perkalian, pembagian dan sebagainya. Contohnya:

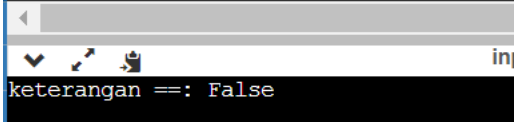
```
1 A=12
2 B=13
3
4 print("Total jumlah:", A+B)
```

Total jumlah: 25

➤ Operator perbandingan

Operator perbandingan adalah operator yang digunakan untuk membandingkan 2 buah yang hasil perbandingannya adalah True atau False, seperti lebih besar dari, lebih kecil dari, sama dengan dan sebagainya. Contohnya:

```
1 A=12
2 B=13
3
4 print("keterangan ==:", A==B)
```



keterangan ==: False

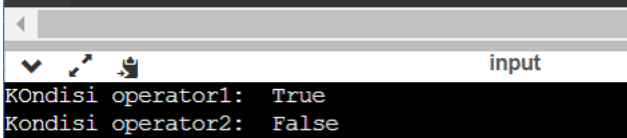
➤ Operator Penugasan

Operator penugasan adalah operator yang digunakan untuk memberi nilai ke variabel. Contohnya a=1, yang artinya operator penugasan yang memberi nilai 1 ke variabel a yang di kiri.

➤ Operator Logika

Operator logika berfungsi untuk melakukan operasi logika, seperti or, and dan not. Contohnya:

```
1 A=12
2 B=13
3
4 operator1 = A and B >=5
5 print("Kondisi operator1: ", operator1)
6
7 operator2 = A ==B
8 print("Kondisi operator2: ", operator2)
```



Kondisi operator1: True  
Kondisi operator2: False

➤ Operator Bitwise

Operator bitwise adalah operator yang melakukan operasi bit terhadap operand dan beroperasi bit per bit sesuai dengan namanya. Contohnya angka 2 dalam bit ditulis 10 dalam notasi biner.

➤ Operator Identitas

Operator identitas berfungsi untuk memeriksa apakah 2 buah nilai atau variabel berada pada lokasi memori yang sama atau tidak dengan menggunakan is dan is not.

➤ Operator Keanggotaan

Operator keanggotaan berfungsi untuk memeriksa apakah suatu nilai atau variabel merupakan anggota yang ditemukan di dalam suatu data (string, list, tuple, set, dan dictionary) atau tidak menggunakan in dan not in.

## # Tipe Data Bentukan

- List : Kumpulan data terurut, dapat diubah dan elemennya bisa sama

```

1 list = ["o", "p", "q", "r"]
2 print(list)

```

['o', 'p', 'q', 'r']

- Tuple : Kumpulan data terurut, tidak dapat diubah dan elemennya bisa sama

```

1 tuple = ["o", "p", "q", "r"]
2 print(tuple)

```

['o', 'p', 'q', 'r']

- Set : Kumpulan data tidak terurut, tidak terindeks dan elemennya tidak ada yang sama

```

1 set = ["o", "p", "q", "r"]
2 print(set)

```

['o', 'p', 'q', 'r']

- Dictionary : Kumpulan data tidak terurut, tidak terindeks dan elemennya bisa sama, serta memiliki key dan nilai.

```

1 dict = {1:"o", 2:"p", 3:"q", 4:"r"}
2 print(dict)
3
4 print(dict[3])

```

{1: 'o', 2: 'p', 3: 'q', 4: 'r'}

q

## #Percabangan

- Percabangan IF Percabangan ini hanya menggunakan satu kondisi saja, yaitu kondisi IF. Contohnya:

```

1 a=int(input("Inputkan nilai a:"))
2
3 if a<10:
4     print(a, "= bisa saja negatif")

```

input

Inputkan nilai a:3

3 = bisa saja negatif

- Percabangan IF-ELSE Percabangan ini berfungsi untuk memberi kondisi untuk 2 pernyataan saja. Contohnya:

```

1 a=int(input("Inputkan nilai a:"))
2
3 if a<10:
4     print(a, "= bisa saja negatif")
5 else :
6     print(a, "= bisa saja tidak")

```

input

Inputkan nilai a:10  
10 = bisa saja tidak

- Percabangan IF-ELSE-IF Percabangan ini berfungsi untuk memberi kondisi untuk lebih dari 2 pernyataan.
- Nested IF Percabangan ini disebut percabangan bersarang karena di dalam suatu percabangan terdapat percabangan yang lain di dalamnya.

### #Perulangan

- Perulangan For

Perulangan for merupakan perulangan yang batasannya telah didefinisikan terlebih dahulu dan biasanya digunakan untuk iterasi pada list, tuple, atau string. Salah satu contohnya:

```

1 biodata = ["Inori", "121140020", "ITERA"]
2
3 for i in biodata:
4     print(i)

```

input

Inori  
121140020  
ITERA

- Perulangan While

Berbeda dengan perulangan for, perulangan while merupakan perulangan yang akan dieksekusi ketika kondisi tertentu terpenuhi. Contohnya:

```

1 a = 3
2 while a!=10:
3     a=int(input("Inputkan nilai a: "))

```

input

Inputkan nilai a: 3  
Inputkan nilai a: 3  
Inputkan nilai a: 3  
Inputkan nilai a: 8  
Inputkan nilai a: 9  
Inputkan nilai a: 10

...Program finished with exit code 0  
Press ENTER to exit console.

### #Fungsi

Fungsi atau method biasanya menggunakan sintaks def dan berfungsi untuk mengantisipasi penulisan blok kode yang berulang.

## MODUL 2

1. Kelas Kelas atau class merupakan sebuah blueprint dari suatu objek yang dibuat. Dengan menggunakan class, maka dapat mendesain suatu objek secara bebas. Namun class tidak dapat langsung digunakan. Solusinya adalah dengan mengimplementasi menjadi sebuah objek terlebih dahulu, seperti kelas Mobil, kelas Manusia dan sebagainya. Contohnya:

```
import random

class Robot:
    jumlah_turn = 0
    Anta = 0
    Alpha = 0
    Leca = 0

    def __init__(self, nama, health, damage):
        self.nama = nama
        self.health = health
        self.damage = damage

    def lakukan_aksi(self, enemy):
        if self.nama == "Antares":
            damage = self.damage
            Robot.Anta += 1
            if Robot.Anta%3 == 0:
                damage *= 1.5
            enemy.terima_aksi(damage)
```

Pada class terdapat \_\_init\_\_ method yang berperan sebagai suatu konstruktor untuk membuat sebuah objek.

a. Atribut/Property Dalam suatu kelas terdapat 2 jenis atribut, yaitu atribut objek dan atribut kelas. Atribut objek merupakan atribut yang dimiliki oleh masing-masing objek atau biasanya berada di dalam sebuah fungsi. Sedangkan atribut kelas adalah atribut yang dideklarasikan di dalam kelas namun tidak di dalam fungsi yang ada di kelas tersebut.

b. Method Method atau disebut juga sebagai fungsi di dalam sebuah kelas. Method dapat diibaratkan sebagai aktivitas/proses yang dapat dilakukan oleh sebuah objek. Misalkan manusia dapat berjalan, berjalan dan sebagainya. Contoh dari Atribut dan Method di dalam kelas:

### 2. Objek

Objek berfungsi sebagai perwakilan suatu kelas saat dipanggil ke main. Cara merepresentasikan objek ini adalah seperti berikut:

```
class Robot:
    jumlah_turn = 0
    Anta = 0
    Alpha = 0
    Leca = 0

    def __init__(self, nama, health, damage):
        self.nama = nama
        self.health = health
        self.damage = damage
```

### 3. Magic Method

Magic method adalah metode yang diawali dan diakhiri dengan double underscore (dunder). Method ini tidak dipanggil secara langsung, tapi dipanggil sistem secara internal ketika melakukan sesuatu. Contohnya saat melakukan penjumlahan, maka operator `__add__` yang dioperasikan.

### 4. Konstruktor

Konstruktor adalah method yang “pasti” dijalankan secara otomatis pada saat sebuah objek dibuat untuk mewakili kelas tersebut. Selain operasi method dasar, konstruktor dapat menerima argumen yang diberikan ketika objek dibuat.

### 5. Destruktor

Destruktor adalah fungsi yang dipanggil ketika user menghapus objek. Fungsi ini bekerja secara otomatis, jadi tidak perlu dilakukan pemanggilan.

### 6. Setter dan Getter

Setter dan getter digunakan untuk melakukan enkapsulasi agar tidak terjadi perubahan data secara tidak sengaja. Setter adalah method yang digunakan untuk menetapkan nilai suatu atribut, sedangkan getter digunakan untuk mengambil nilai.

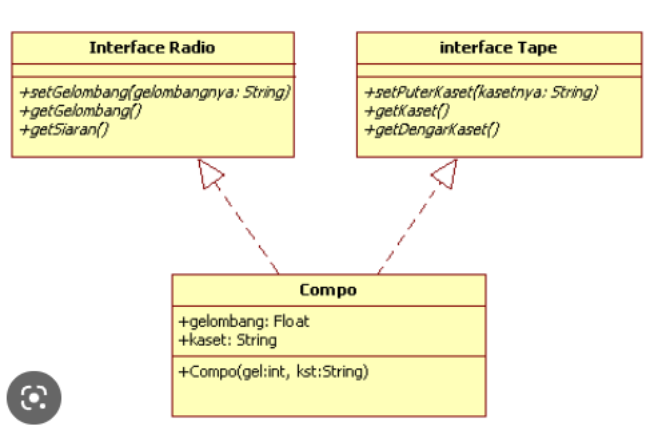
### 7. Decorator

Decorator yang biasanya ditandai dengan simbol (`@`) adalah alat yang memungkinkan programmer untuk mengubah perilaku fungsi atau kelas dengan cara membungkus fungsi lain untuk memperluas perilaku dari fungsi yang dibungkus, seperti `@property`, `@classmethod` dan lain sebagainya. Salah satu contohnya adalah Decorator Property:

## MODUL 3

### 1. Abstraksi

Abstraksi adalah konsep OOP dimana model yang dibuat hanya memperlihatkan atribut yang esensial dan menyembunyikan detail-detail yang tidak penting dari user. gunanya untuk mengurangi kompleksitas, atau dapat dikatakan bahwa Abstraksi adalah cara untuk menyembunyikan informasi yang tidak dibutuhkan.



### 2. Enkapsulasi

Enkapsulasi adalah sebuah metode yang digunakan untuk mengatur struktur kelas dengan cara menyembunyikan alur kerja dari kelas tersebut. Yang dimaksud dengan struktur kelas tersebut adalah property dan method. Untuk membatasi hak akses terhadap property dan method dalam suatu kelas, terdapat 3 jenis access modifier yang terdapat dalam python, yaitu public access, protected access, dan private access.

- Public Access Modifier Objek berjenis public baik atribut maupun metode dapat diakses dari dalam dan luar kelas tersebut. Penulisan atribut public seperti pada biasanya, yaitu tanpa menggunakan underscore. Contohnya:

```
class AkunBank:
    list_pelanggan={}
    def __init__(self, nomor, nama, saldo):
        self.__no_pelanggan=nomor
        self.__nama_pelanggan=nama
        self.__jumlah_saldo=saldo
        AkunBank.list_pelanggan.update({self.__no_pelanggan:self.__nama_pelanggan})
        self.menu=0
```

- Protected Access Modifier Objek berjenis protected baik atribut maupun metode hanya dapat diakses dari dalam kelas dan turunan kelasnya. Untuk membuat atribut atau metode objek berjenis protected dapat dilakukan dengan cara menambahkan awalan garis bawah tunggal ( `_` ) ke dalam nama tersebut. Contohnya:



```
def __init__(self,nama,status,harga_asli):
    self.__nama=nama
    self.__status=status
    self.__harga_asli=harga_asli
```

- c. Private Access Modifier Atribut maupun metode yang berjenis private hanya dapat diakses dari dalam kelasnya saja. Untuk membuat atribut atau metode objek berjenis private dapat dilakukan dengan cara menambahkan awalan garis bawah ganda ( \_\_ ) ke dalam nama. Terdapat cara untuk mengakses atribut private dari luar kelas yaitu: nama\_objek\_instance.\_\_NamaKelas\_\_nama\_atribut.

### 3. Object

- Membuat Instance Object.  
Untuk membuat instance dari kelas yang telah dibuat dapat dilakukan dengan menggunakan nama dari class kemudian argumen diterima oleh metode init.
- Mengakses Atribut Object.  
Dari objek yang telah dibuat dapat dilakukan pengaksesan atribut dari objek dengan menggunakan operator dot(titik).
- Menambah, Menghapus dan Mengubah Atribut Object.  
Objek yang sudah dibuat dapat dimodifikasi seperti ditambah, dihapus, ataupun diubah atributnya.
- Cara memodifikasi atribut dari suatu objek
  - getattr(obj, name, [default]) – Mengakses atribut dari objek.
  - hasattr(obj, name) – Memeriksa apakah suatu objek memiliki atribut tertentu.
  - setattr(obj, name, value) – Mengatur nilai atribut. Jika ternyata atribut tidak ada, maka atribut tersebut akan dibuat.
  - delattr(obj, name) – Menghapus atribut dari suatu objek.

## MODUL 4

### 1. Inheritance (Pewarisan)

Inheritance adalah salah satu konsep dasar dari Object Oriented Programming (OOP). Pada inheritance, kita dapat menurunkan kelas dari kelas lain untuk hirarki kelas yang saling berbagi atribut dan metode. Contoh:

```
main.py
1 class Parent:
2     pass
3
4 class Child(Parent):
5     pass
```


- Inheritance Identik  
Inheritance identik merupakan pewarisan yang menambahkan constructor pada class child sehingga class child memiliki constructor-nya sendiri tanpa menghilangkan constructor pada class parent-nya.
- Menambah Karakteristik pada Child Class

Pada child class dapat ditambahkan beberapa fitur tambahan baik atribut maupun method sehingga child class tidak identik dengan parent class.

## 2. Polymorphism

Polymorphism berarti banyak (poly) dan bentuk (morphism), dalam Pemrograman Berbasis Objek konsep ini memungkinkan digunakannya suatu interface yang sama untuk memerintah objek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda. Contohnya:


```
1 class Raksasa():
2     def type(self):
3         print("Berat")
4     def warna(self):
5         print("mejikuhibiniu")
6
7 class Megalodon(Raksasa):
8     def type(self):
9         print("HO ho ho")
10    def warna(self):
11        print("telur asin")
12
13 def func(objek):
14     objek.type()
15     objek.warna()
16
17 Raksasa1=Raksasa()
18 Megalodon1=Megalodon()
19 func(Raksasa1)
20 print()
21 func(Megalodon1)
22
23 Megalodon1=Megalodon()
24 Megalodon1.type()
```



## 3. Override/Overriding

Pada konsep OOP di Python kita dapat menimpa suatu metode yang ada pada parent class dengan mendefinisikan kembali method dengan nama yang sama pada child class.

```
1 class Raksasa():
2     def type(self):
3         print("Berat")
4
5 class Megalodon(Raksasa):
6     def type(self):
7         print("HO ho ho")
8
9 Megalodon1=Megalodon()
10 Megalodon1.type()
```



## 4. Overloading

Metode overloading mengizinkan sebuah class untuk memiliki sekumpulan fungsi dengan nama yang sama dan argumen yang berbeda. Secara umum overloading memiliki beberapa signature, yaitu jumlah argumen, tipe argumen, tipe keluaran dan urutan argumen.

```
1 class Raksasa():
2     def type(self):
3         print("Berat")
4
5 class Megalodon(Raksasa):
6     def type(self):
7         print("HO ho ho")
8
9 def func(objek):
10     objek.type()
11
12 Raksasa1=Raksasa()
13 Megalodon1=Megalodon()
14 func(Raksasa1)
15 func(Megalodon1)
```

Berat  
HO ho ho

## 5. Multiple Inheritance

Python mendukung pewarisan ke banyak kelas. Kelas dapat mewarisi dari banyak orang tua. Bentuk syntax multiple inheritance adalah sebagai berikut.

```
1 class Atap:
2     pass
3
4 class Kayu:
5     pass
6
7 class Batu:
8     pass
9
10 class Rumah(Atap, Kayu, Batu):
11     pass
```

## 6. Method Resolution Order di Python

MRO adalah urutan pencarian metode dalam hirarki class. Hal ini terutama berguna dalam multiple inheritance. Urutan MRO dalam python yaitu bawah-atas dan kiri-kanan. Artinya, method dicari pertama kali di kelas objek. jika tidak ditemukan, pencarian berlanjut ke super class. Jika terdapat banyak superclass (multiple inheritance), pencarian dilakukan di kelas yang paling kiri dan dilanjutkan ke kelas sebelah kanan.

```

1 class Atap:
2     pass
3
4 class Kayu:
5     pass
6
7 class Batu:
8     pass
9
10 class Gubuk(Atap, Kayu):
11     pass
12
13 class Gedung(Atap, Batu)
14     pass
15
16 class Rumah(Atap, Kayu, Batu):
17     pass

```

## 7. Dynamic Cast

- Implisit Python secara otomatis mengkonversikan tipe data ke tipe data lainnya tanpa ada campur tangan pengguna
- Eksplisit Pengguna mengubah tipe data sebuah objek ke tipe data lainnya dengan fungsi yang sudah ada dalam python seperti `int()`, `float()`, dan `str()`. dapat berisiko terjadinya kehilangan data.

## 8. Casting

- Downcasting Parent class mengakses atribut yang ada pada kelas bawah (child class)

```

1 class Mhs:
2     def __init__(self, nama, NIM):
3         self.nama=nama
4         self.NIM=NIM
5
6     def biodata(self):
7         print("Nama: ", self.nama)
8         print("NIM: ", self.NIM)
9         print("Umur: ", self.Umur)
10
11 class Diri(Mhs):
12     def __init__(self, nama, NIM, Umur):
13         super().__init__(nama, NIM)
14         self.Umur=Umur
15
16 Inori=Diri("Inori", 121140020, 19)
17 Inori.biodata()

```

input

```

Nama: Inori
NIM: 121140020
Umur: 19

```

- Upcasting Child class mengakses atribut yang ada pada kelas atas (parent class).

```
1 class Mhs:
2     Domisili = "Lampung"
3     def __init__(self, nama, NIM):
4         self.nama=nama
5         self.NIM=NIM
6
7 class Diri(Mhs):
8     def __init__(self, nama, NIM, Umur):
9         super().__init__(nama, NIM)
10        self.Umur=Umur
11
12    def biodata(self):
13        print("DOM: ", super().Domisili)
14        print("Nama: ", self.nama)
15        print("NIM: ", self.NIM)
16        print("Umur: ", self.Umur)
17
18 Inori=Diri("Inori", 121140020, 19)
19 Inori.biodata()
```

input

DOM: Lampung  
Nama: Inori  
NIM: 121140020  
Umur: 19

c. Type casting Konversi tipe kelas agar memiliki sifat/perilaku tertentu yang secara default tidak dimiliki kelas tersebut. Karena Python merupakan bahasa pemrograman berorientasi objek, maka semua variabel atau instansi di Python pada dasarnya merupakan objek (kelas) yang sifat/perilakunya dapat dimanipulasi jika dibutuhkan (umumnya melalui magic method).