

# RESUME PRAK PBO RB

Nama : Muhammad Ihsanuddin Faruq

NIM : 121140214

kelas siakad : PBO RD

kelas praktikum : PBO RB

## BAB 1

### 1. Pengenalan Bahasa Pemrograman Python

Bahasa pemrograman python dibuat oleh Guido Van Rossum pada tahun 1980-an akhir di Centrum Wiskunde & Informatica Belanda. Python mendukung banyak paradigma pemrograman seperti object-oriented, functional dan structured. Bahasa Python menjadi populer sekarang ini dikarenakan sintaks nya yang mudah, didukung oleh library(modul) yang berlimpah dan Python bisa dipakai untuk pemrograman desktop maupun mobile, CLI, GUI, web, otomatisasi, hacking, IoT, robotika, dan lain sebagainya.

### 2. Dasar Pemrograman Python

#### a. Sintaks Dasar

- **Statement**

Semua perintah yang bisa dieksekusi Python disebut statement. Pada Python akhir dari sebuah statement adalah baris baru (newline) tapi dimungkinkan membuat statement yang terdiri dari beberapa baris menggunakan backslash (\).

```
2
3  nanas, jeruk, mangga, melon = 1, 2, 3, 4
4
5  rujak = nanas + \
6         jeruk + \
7         mangga + \
8         melon + \
9  print(rujak)
```

- **Variabel dan Tipe Data Primitif**

Variabel merupakan lokasi penyimpanan yang berguna untuk menyimpan suatu data atau suatu nilai. Dalam mendeklarasikan suatu variabel dalam pemrograman, perlu diketahui tipe-tipe data yang berhubungan dengan variabel yang akan dideklarasikan, di bawah ini merupakan tipe data yang ada :

Tipe data	Jenis	Nilai
bool	Boolean	True atau False
int	Bilangan bulat	Seluruh bilangan bulat
float	Bilangan real	Seluruh bilangan real
string	Teks	Kumpulan karakter

contohnya seperti dibawah ini

```

3
4  Num1 = int(input ("angka pertama:"))
5  Num2 = int(input ("angka kedua: "))
6  print ("Hasil : ", Num1 + Num2)
7

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```

angka pertama:12
angka kedua: 43
Hasil : 55
PS C:\Users\USER>

```

## • Operator

### ○ Operator Aritmatika

Operator aritmatika adalah operator yang digunakan untuk melakukan operasi matematika, seperti penjumlahan, pengurangan, perkalian, pembagian, dan sebagainya. Berikut ialah operator aritmatika dalam python

Operator	Nama dan Fungsi	Contoh
+	Penjumlahan, untuk menambahkan 2 operand	$x + y$
-	Pengurangan, untuk mengurangi 2 operand	$x - y$
*	Perkalian, untuk mengalikan 2 operand	$x * y$
/	Pembagian, untuk membagi 2 operand	$x / y$
**	Perpangkatkan, membuat pangkat pada bilangan	$x ** y$
//	Pembagian bulat, untuk hasil tanpa angka di belakang koma	$x // y$
%	Modulus, menghasilkan sisa 2 bilangan	$x \% y$

contohnya seperti dibawah ini

```
3
4  Num1 = int(input ("angka pertama:"))
5  Num2 = int(input ("angka kedua: "))
6  print ("Hasil : ", Num1 % Num2)
7
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
angka pertama:23
angka kedua: 10
Hasil : 3
PS C:\Users\USER> █
```

```
3
4  Num1 = int(input ("angka pertama:"))
5  Num2 = int(input ("angka kedua: "))
6  print ("Hasil : ", Num1 // Num2)
7
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
angka pertama:20
angka kedua: 3
Hasil : 6
PS C:\Users\USER> █
```

## ● Operator Perbandingan

Operator perbandingan adalah operator yang digunakan untuk membandingkan 2 buah nilai. Hasil perbandingannya adalah True atau False tergantung kondisi.

berikut ialah tabel Operator Perbandingan

Operator	Nama dan Fungsi	Contoh
>	Lebih besar dari, jika angka kiri lebih besar ketimbang yang kanan, maka akan menghasilkan jawaban <b>true</b>	$x > y$
<	Lebih kecil dari, jika angka kiri lebih kecil dari yang kanan, maka akan menghasilkan jawaban <b>true</b>	$x < y$
==	sama dengan, jika angka kanan dan kiri	$x == y$

	besarnya sama, maka akan menghasilkan	
	jawaban <b>true</b>	
<b>!=</b>	Tidak sama dengan, jika angka sebelah kiri tidak sama dengan yang kanan, maka akan menimbulkan jawaban <b>true</b>	$x \neq y$
<b>&gt;=</b>	Lebih besar atau sama dengan, jika angka kiri mempunyai nilai lebih besar atau sama dengan yang kanan, maka hasilnya akan <b>true</b>	$x \geq$
<b>&lt;=</b>	Kurang dari atau sama dengan, jika nilai sebelah kiri lebih kecil atau sama dengan yang kanan, maka hasilnya akan menjadi <b>true</b>	$x \leq y$

contohnya ialah sebagai berikut

```

3
4  Num1 = int(input ("angka pertama:"))
5  Num2 = int(input ("angka kedua: "))
6  lebih_kecil = Num1 < Num2
7  print ("Hasil : ", lebih_kecil)
8

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```

angka pertama:20
angka kedua: 23
Hasil : True
PS C:\Users\USER>

```

```

4  Num1 = int(input ("angka pertama:"))
5  Num2 = int(input ("angka kedua: "))
6  sama_dengan = Num1 == Num2
7  print ("Hasil : ", sama_dengan)
8

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```

angka pertama:12
angka kedua: 22
Hasil : False
PS C:\Users\USER>

```

## - Operator Penugasan

Operator penugasan adalah operator yang digunakan untuk memberi nilai ke variabel.  $a = 7$  adalah contoh operator penugasan yang memberi nilai 7 di kanan ke variabel  $a$  yang ada di kiri. Berikut ialah tabel mengenai Operator Penugasan

Operator	Penjelasan	Contoh
=	Menugaskan nilai yang ada di kanan menjadi operand sebelah kiri	$c = a + b$ menugaskan $a + b$ ke $c$
+=	Menambahkan operand yang di kanan dengan yang dikiri maka hasilnya ditugaskan ke operand yang kiri	$c += a$ sama dengan $c = c + a$
-=	Mengurangi operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	$c -= a$ sama dengan $c = c - a$
*=	Mengalikan operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	$c *= a$ sama dengan $c = c * a$
/=	Membagi operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	$c /= a$ sama dengan $c = c / a$
**=	Memangkatkan operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	$c **= a$ sama dengan $c = c ** a$

//=	Melakukan pembagian bulat operand di kanan terhadap operand di kiri dan hasilnya disimpan di operand yang di kiri	<b>c //= a</b> sama dengan <b>c = c // a</b>
%=	Melakukan operasi sisa bagi operand di kanan dengan operand di kiri dan hasilnya disimpan di operand yang di kiri	<b>c %= a</b> sama dengan <b>c = c % a</b>

Contohnya ialah seperti berikut

```

3
4  Num1 = int(input ("angka pertama:"))
5  Num2 = 10
6  Num2 += Num1
7  print ("Hasil : ", Num2)
8

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```

angka pertama:23
Hasil :  33
PS C:\Users\USER>

```

## • Operator Logika

Operator logika adalah operator yang digunakan untuk melakukan operasi logika. berikut ialah tabel dari Operator Logika

Operator	Penjelasan	Contoh
and	Hasilnya ialah <b>true</b> jika kedua operand bernilai besar	x and y
or	Hasilnya ialah <b>true</b> jika salah satu atau kedua operand bernilai besar	x or y
not	Hasilnya ialah <b>true</b> jika operandnya bernilai salah	not x

contohnya ialah sebagai berikut

```
4  #Operator logika
5  X = 10
6  Y = 20
7  Operator1 = Y and Y<15
8  print ("hasil kondisi Operator1 : ", Operator1)
9
10 Operator2 = X or Y>15
11 print ("hasil Kondisi Operator2 : ", Operator2)
12
13 Operator3 = X and Y>15 or X and Y<11
14 print ("hasil kondisi Operator3 : ", Operator3)
15 |
16
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   +

```
hasil kondisi Operator1 : False
hasil Kondisi Operator2 : 10
hasil kondisi Operator3 : True
PS C:\Users\USER>
```

## ● Operator Bitwise

Operator bitwise adalah operator yang melakukan operasi bit terhadap operand. Operator ini beroperasi bit per bit sesuai dengan namanya. Sebagai misal, angka 2 dalam bit ditulis 10 dalam notasi biner dan angka 7 ditulis 111. Pada tabel di bawah ini, misalkan x = 10 ( 0000 1010) dalam biner dan y = 4 (0000 0100) dalam biner. berikut ialah tabel mengenai Operator Bitwise

Operator	Nama	Contoh
&	Bitwise AND	x& y = 0 (0000 0000)
	Bitwise Or	x   y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0 000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

berikut ialah contoh penerapannya

```
4  num1 = int(input("angka pertama :"))
5  num2 = int(input("angka kedua : "))
6  bitwise = num1 ^ num2
7  print ("hasil : ", bitwise)
8  |
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
angka pertama :23
angka kedua : 10
hasil :  29
PS C:\Users\USER> |
```

```
4  num1 = int(input("angka pertama :"))
5  num2 = int(input("angka kedua : "))
6  bitwise = num1 ^ num2
7  print ("hasil : ", bitwise)
8  |
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
angka pertama :20
angka kedua : 25
hasil :  13
PS C:\Users\USER> |
```

- **Operator Identitas**

Operator identitas adalah operator yang memeriksa apakah dua buah nilai ( atau variabel ) berada pada lokasi memori yang sama. berikut ialah tabel mengenai Operator Identitas

Operator	Penjelasan	Contoh
is	True jika kedua operand identik (menunjuk ke objek yang sama)	x is True
is not	True jika kedua operand tidak identik (tidak merujuk ke objek yang sama)	x is <b>not</b> True

berikut ialah contohnya



```

3
4  A = 'Praktikum'
5  B = 'Pratikum'
6  C = 'PBO'
7
8  print (" A is B :", A is B)
9  print ("A is not C: ", A is not B)

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```

A is B : False
A is not C:  True
PS C:\Users\USER> 

```

## • Operator Keanggotaan

Operator keanggotaan adalah operator yang digunakan untuk memeriksa apakah suatu nilai atau variabel merupakan anggota atau ditemukan di dalam suatu data (string, list, tuple, set, dan dictionary). berikut ialah tabel mengenai Operator Keanggotaan

Operator	Penjelasan	Contoh
in	True jika nilai/variabel ditemukan di dalam data	5 in x
not in	True jika nilai/variabel tidak ada di dalam data	5 not in x

berikut ialah contohnya

```
4 A = 'Praktikum'
5 B = 'Pratikum'
6 C = 'PBO'
7
8 print (" A in B :", A in B)
9 print ("A not C: ", A not in B)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
A in B : False
A not C: True
PS C:\Users\USER> 
```

- **Tipe Data Bentuk**

ada 4 tipe data bentuk, yaitu

- **List**

Sebuah kumpulan data yang terurut, dapat diubah, dan memungkinkan ada anggota yang sama

- **Tuple**

Sebuah kumpulan data yang terurut, tidak dapat diubah, dan memungkinkan ada anggota yang sama

- **Set**

Sebuah kumpulan data yang tidak berurutan, tidak terindeks, dan tidak memungkinkan ada anggota yang sama

- **Dictionary**

Sebuah kumpulan data yang tidak berurutan, dapat diubah, tidak memungkinkan ada anggota yang sama

- **Percabangan**

- a. **Percabangan IF**

```
print ("masukkan nilai n :", end="")
N = int(input())

if (N>0 ):
    print (str(N) + "bilangan positif")
```

codingan diatas untuk percabangan menentukan bilangan positif. jika memasukkan bilangan negatif, maka program tidak akan mengeluarkan hasil.

### b. Percabangan IF-ELSE

```
4 print ("masukkan nilai n :", end="")
5 N = int(input())
6
7 if (N>0 ):
8     print (str(N) + "bilangan positif")
9 else:
10    print (str(N) + "bilangan negatif")
```

codingan ini sama dengan contoh sebelumnya, hanya saja jika kita memasukkan angka negatif, maka program akan tetap berjalan dengan mengeluarkan hasil seperti dibawah

```
6
7 if (N>0 ):
8     print (str(N) + "bilangan positif")
9 else:
10    print (str(N) + "bilangan negatif")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
masukkan nilai n :-5
-5bilangan negatif
PS C:\Users\USER>
```

### c. Percabangan IF-ELSE-IF

```
4 print ("masukkan nilai n :", end="")
5 N = int(input())
6
7 if (N>0 ):
8     print (str(N) + "bilangan positif")
9 elif (N==0):
10    print (str(N) + "bilangan nol")
11 else:
12    print (str(N) + "bilangan negatif")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
masukkan nilai n :0
0bilangan nol
PS C:\Users\USER>
```

codingan ini berfungsi untuk melengkapi codingan yang sebelumnya, yang dimana ketika kita memasukkan angka nol, program tidak akan berjalan, maka percabangan kali ini akan membuatnya menjalankan program ketika *user* memasukkan angka nol

### d. Nested IF

Selain menggunakan code seperti gambar diatas untuk menentukan bilangan positif, negatif atau bilangan nol, kita dapat mengubah format code menjadi percabangan bersarang (nested if) dengan menempatkan percabangan di dalam percabangan.

```
3
4 print ("masukkan nilai n :", end="")
5 N = int(input())
6
7 if (N<=0):
8     if (N==0):
9         print (str(N) + "bilangan nol")
10    else:
11        print (str(N) + "bilangan negatif")
12
13 else:
14     print (str(N) + "bilangan positif")
15
```

- **Perulangan**

di dalam python terdapat 2 perulangan, sebagai berikut

1. **Perulangan For**

pada perulangan For, dapat digunakan pada list, string, dan tuple contohnya ialah sebagai berikut

- **Perulangan For pada list**

```
3
4 #contoh perulangan pada list
5 namaMahasiswa = ["Aqyra", "Budi", "Qurfa", "Qisha"]
6 for i in namaMahasiswa:
7     print(i)
```

PROBLEMS OUTPUT TERMINAL ... Python + - [] ...

Aqyra  
Budi  
Qurfa  
Qisha  
PS C:\Users\USER>

- **Perulangan For pada string**

```
3
4 #contoh perulangan pada tuple
5 for i in "Aqyra":
6     print(i)
7
```

PROBLEMS OUTPUT TERMINAL ... Python +

A  
q  
y  
r  
a  
PS C:\Users\USER>

- **Perulangan For pada tuple**

```
3
4 #contoh perulangan pada tuple
5 namaMahasiswa = ("Aqyra", "Budi", "Qurfa", "Qisha")
6 for i in namaMahasiswa:
7     print(i)
```

PROBLEMS OUTPUT TERMINAL ... Python + - [] ..

Aqyra  
Budi  
Qurfa  
Qisha  
PS C:\Users\USER>

## 2. Perulangan While

Dengan menggunakan while maka dapat dilakukan perulangan selama kondisi tertentu terpenuhi. sintaks umum pada perulangan contohnya seperti menghitung huruf a dalam suatu kalimat

```
4 #contoh perulangan while menghitung huruf a
5 count = int(0)
6
7 kalimat = (input())
8
9 while (kalimat != '!'):
10
11     if kalimat == 'a':
12         count+=1
13     kalimat = (input())
14
15 print ("jumlah huruf a ialah", count)
16
```

- **Fungsi**

fungsi ialah suatu perintah yang dapat mengeksekusi suatu kode tanpa harus menuliskannya berulang kali, contohnya ialah sebagai berikut

```
4 def kelulusan(nama, nilai):
5     if nilai > 50:
6         return f"{nama} lulus dengan nilai {nilai}"
7     else :
8         return f"{nama} anda mengulang dengan nilai {nilai}"
9
10 list_mahasiswa = [
11     ["Udin", 85], ["Aqyra", 90], ["Seto", 40]
12 ]
13
14 for i in list_mahasiswa:
15     print (kelulusan(i[0], i[1]))
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [X] ...

Udin lulus dengan nilai 85  
Aqyra lulus dengan nilai 90  
Seto anda mengulang dengan nilai 40  
PS C:\Users\USER>

## BAB 2

### 1. Kelas

Kelas atau class pada python bisa kita katakan sebagai sebuah blueprint (cetakan) dari objek (atau instance) yang ingin kita buat. Dengan menggunakan kelas kita dapat mendesain objek secara bebas. Kelas berisi dan mendefinisikan atribut/properti dan metode untuk objeknya nanti. Satu buah kelas dapat membuat banyak objek dan kelas sendiri tidak bisa langsung digunakan, harus diimplementasikan menjadi sebuah objek dulu, dapat disebut Instansiasi. seperti contoh dibawah ini

```
4 class Orang:
5     jumlah_kaki = 2
6     jumlah_tangan = 2
7
8     def __init__(self, nama, pekerjaan, tahun_lahir):
9         self.nama = nama
10        self.pekerjaan = pekerjaan
11        self.tahun_lahir = tahun_lahir
12
13 orang1 = Orang("Aqyra", "mahasiswa", 2001)
14 orang2 = Orang("Vania", "Ibu rumah tangga", 2002)
15 print(orang1.nama)
16 print(orang1.pekerjaan)
17 print(orang2.nama)
18 print(orang2.pekerjaan)
19 print(orang1.jumlah_kaki)
20 print(orang2.jumlah_kaki)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [X] ...

Aqyra  
mahasiswa  
Vania  
Ibu rumah tangga  
2  
2  
PS C:\Users\USER> [ ]

- **Atribut**

Dalam suatu kelas, umumnya dideklarasikan sebuah variabel yang akan disebut sebagai sebuah atribut. Karena dideklarasikan di dalam sebuah kelas, maka setiap objek yang dibentuk dari kelas tersebut juga akan memiliki atribut yang dimiliki oleh kelas tersebut. Terdapat 2 jenis atribut, yaitu atribut kelas dan atribut objek.

```
1 class Orang:
2     jumlah_kaki = 2
3     jumlah_tangan = 2
4
5     def __init__(self, nama, pekerjaan, tahun_lahir):
6         self.nama = nama
7         self.pekerjaan = pekerjaan
8         self.tahun_lahir = tahun_lahir
9
10 orang1 = Orang("Enrico", "Mahasiswa", 2000)
11 orang2 = Orang("Andi", "Direktur", 1980)
12 print(orang1.nama)
13 print(orang1.pekerjaan)
14 print(orang2.nama)
15 print(orang2.pekerjaan)
16 print(orang1.jumlah_kaki)
17 print(orang2.jumlah_kaki)
```

- **Method**

Selain atribut, elemen lain yang terdapat dalam suatu kelas adalah method. Method adalah suatu fungsi yang terdapat di dalam kelas. Sama halnya seperti atribut, semua objek yang dibuat menggunakan kelas yang sama akan memiliki method yang sama pula. Method dapat diibaratkan sebagai sebuah aktivitas/proses yang dapat dilakukan oleh sebuah objek.

```
1 class Orang:
2     jumlah_kaki = 2
3     jumlah_tangan = 2
4
5     def __init__(self, nama, pekerjaan, tahun_lahir):
6         self.nama = nama
7         self.pekerjaan = pekerjaan
8         self.tahun_lahir = tahun_lahir
9
10    def berjalan_kedepan(self):
11        print("Melangkah")
12        print("Posisi berpindah")
13
14 orang1 = Orang("Enrico", "Mahasiswa", 2000)
15 orang1.berjalan_kedepan()
```

## 2. Objek

Objek adalah sesuatu yang “mewakili” kelas. Objek disini berfungsi sebagai pengganti pemanggilan sebuah kelas, maka sebuah objek hanya dapat mewakili sebuah kelas saja. Untuk membuat sebuah objek yang mewakili sebuah kelas, kita dapat membuatnya dengan cara memanggil nama dari kelas yang diinginkan, ditambah dengan tanda kurung ().

## 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 seperti menggunakan operator tambah (\_\_add\_\_), membuat objek (\_\_init\_\_), dan lain-lain.

## 4. Konstruktor

Konstruktor adalah method yang “pasti” dijalankan secara otomatis pada saat sebuah objek dibuat untuk mewakili kelas tersebut.

```
class kelas_yang_dipanggil:

    #ini constructor
    def __init__(self,ini_argumen,ini_argumen_juga):
        self.nama = ini_argumen
        self.pekerjaan = ini_argumen_juga

    #ini buat panggil nama
    def panggilnama (self):
        print("Orang ini bernama", self.nama)
    #ini buat panggil kerjaan
    def panggilpekerjaan (self):
        print("Orang ini bekerja sebagai",self.pekerjaan)

ini_objek = kelas_yang_dipanggil("Zhongli","Tukang Gali Kubur")

ini_objek.panggilnama()
ini_objek.panggilpekerjaan()
```

## 5. Destruktor

Destruktor adalah fungsi yang dipanggil ketika user menghapus objek. Fungsi ini bekerja secara otomatis, jadi tidak perlu dilakukan pemanggilan. Tujuan adanya destruktur adalah melakukan final cleaning up atau “bersih-bersih” terakhir sebelum sebuah objek benar-benar dihapus dari memori.



```

1 class Angka:
2     def __init__(self, angka):
3         self.angka = angka
4
5     def __del__(self):
6         print ("objek {self.angka} dihapus")
7
8 N=Angka(1)
9 P=Angka(2)
10

```

## 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 khususnya atribut private dan protected (akan diajarkan lebih jelas pada materi minggu selanjutnya), sedangkan getter digunakan untuk mengambil nilai.

```

1 class siswa:
2     def __init__(self, umur = 0):
3         self._umur = umur
4
5     # getter method
6     def get_umur(self):
7         return self._umur
8
9     # setter method
10    def set_umur(self, x):
11        self._umur=x
12
13    raj = siswa()
14
15    # setting the umur using setter
16    raj.set_umur(19)
17    # retrieving umur using getter
18    print(raj.get_umur())
19    print(raj._umur)

```

## 7. Decorator

Selain menggunakan fungsi setter dan getter tambahan seperti pada contoh sebelumnya, dalam Python kita juga dapat memanfaatkan property decorator untuk mendapatkan hasil serupa. Bedanya, melalui property decorator ini kita tidak perlu membuat fungsi lagi dengan nama yang berbeda-beda (cukup menggunakan 1 buah nama/variabel). Contoh penggunaan decorator:

```
class Siswa:
    def __init__(self, nama, umur = 15):
        self.__nama = nama
        self.__umur = umur

    @property
    def umur(self):
        print("Fungsi getter umur dipanggil")
        return self.__umur

    @umur.setter
    def umur(self, x):
        print("Fungsi setter umur dipanggil")
        self.__umur = x

Bambang = Siswa("Bambang")

# Akan error karena bersifat private
#print(Bambang.__umur)

# Gunakan fungsi property decorator
print(Bambang.umur)
Bambang.umur += 5
print(Bambang.umur)
```

## BAB 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.

### 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.

Abstraksi adalah cara untuk menyembunyikan informasi yang tidak dibutuhkan, sedangkan enkapsulasi adalah cara menyembunyikan atribut suatu entitas serta metode untuk melindungi informasi dari luar.

- **Public Access Modifier**

```
class Mahasiswa:
    global_variable_jumlah = 0

    def __init__(self, nama, semester):
        self.nama = nama
        self.semester = semester
        Mahasiswa.global_variable_jumlah = Mahasiswa.global_variable_jumlah + 1

    def printjumlah(self):
        print("total mahasiswa adalah ", Mahasiswa.global_variable_jumlah, " Orang")

    def printprofil(self):
        print("Nama : ", self.nama)
        print("Semester : ", self.semester)
        print()
```

- **Protected Access Modifier**

```

class Mobil:
    #protected variable
    _merk = None
    _warna = None

    #constructor
    def __init__(self, merk, warna):
        self._merk = merk
        self._warna = warna

    #protected method
    def _tampilMobil(self):
        print("Merk Mobil : ", self._merk)
        print("Warna Mobil : ", self._warna)

```

- **Private Access Modifie**

```

1 class Mobil:
2     #private variable
3     __merk = None
4     __warna = None
5
6     #constructor
7     def __init__(self, merk, warna):
8         self.__merk = merk
9         self.__warna = warna
10
11     #private method
12     def _tampilMobil(self):
13         print("Merk Mobil : ", self.__merk)
14         print("Warna Mobil : ", self.__warna)
15

```

- **Setter dan Getter**

Seperti yang sudah dijelaskan pada minggu yang lalu, setter adalah sebuah method yang digunakan untuk mengatur sebuah property yang ada di dalam suatu kelas/objek. Sedangkan getter adalah sebuah method yang digunakan untuk mengambil nilai dari suatu property.

**Tanpa Decorator**

```

1  class siswa:
2      def __init__(self, umur = 0):
3          self._umur = umur
4
5      # getter method
6      def get_umur(self):
7          return self._umur
8
9      # setter method
10     def set_umur(self, x):
11         self._umur=x
12
13     raj = siswa()
14
15     # setting the umur using setter
16     raj.set_umur(19)
17     # retrieving umur using getter
18     print(raj.get_umur())
19     print(raj._umur)

```

## Dengan Decorator

```

class Siswa:
    def __init__(self, nama, umur = 15):
        self.__nama = nama
        self.__umur = umur

    @property
    def umur(self):
        print("Fungsi getter umur dipanggil")
        return self.__umur

    @umur.setter
    def umur(self, x):
        print("Fungsi setter umur dipanggil")
        self.__umur = x

Bambang = Siswa("Bambang")

# Akan error karena bersifat private
#print(Bambang.__umur)

# Gunakan fungsi property decorator
print(Bambang.umur)
Bambang.umur += 5
print(Bambang.umur)

```

## BAB 4

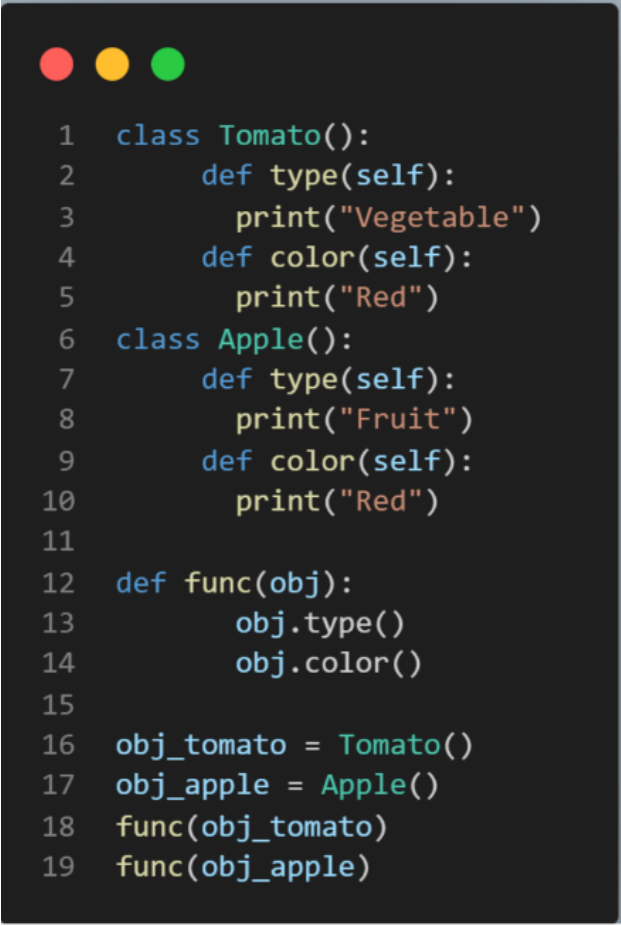
### 1. Inheritance (Pewarisan)

Inheritance adalah salah satu konsep dasar dari Pemrograman Berbasis Objek (OOP). Pada inheritance, kita dapat menurunkan kelas dari kelas lain untuk hirarki kelas yang saling berbagi atribut dan metode. Contohnya terdapat base class "Animal" dan terdapat class "Horse" yang merupakan turunan dari class "Animal".

```
1 v class Manusia:
2 v     def __init__(self, nama, umur):
3         self.nama = nama
4         self.umur = umur
5
6 v     def cetakProfile(self):
7         print(f"{self.nama} berumur {self.umur}")
8
9 v class Mahasiswa(Manusia):
10     pass
11
12 mhs1 = Mahasiswa("Joko", 20)
13 mhs1.cetakProfile()
```

### 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.



```

1  class Tomato():
2      def type(self):
3          print("Vegetable")
4      def color(self):
5          print("Red")
6  class Apple():
7      def type(self):
8          print("Fruit")
9      def color(self):
10         print("Red")
11
12  def func(obj):
13      obj.type()
14      obj.color()
15
16  obj_tomato = Tomato()
17  obj_apple = Apple()
18  func(obj_tomato)
19  func(obj_apple)

```

### 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. Dengan begitu maka method yang ada parent class tidak berlaku dan yang akan dijalankan adalah method yang terdapat di child class.

```

1  class bangunDatar():
2      def tampil(self):
3          print("Ini bangun datar")
4
5  class Persegi(bangunDatar):
6      def tampil(self):
7          print("Ini persegi")
8
9  P1 = Persegi()
10 P1.tampil()

```

#### **4. Overloading**

Metode overloading mengizinkan sebuah class untuk memiliki sekumpulan fungsi dengan nama yang sama dan argumen yang berbeda. Akan tetapi, Python tidak mengizinkan pendeklarasian fungsi (baik pada class ataupun tidak) dengan nama yang sama. Hal itu menyebabkan implementasi overloading pada python menjadi “tricky”.

#### **5. 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.

#### **6. Dynamic Cast**

Dynamic cast atau type conversion adalah proses mengubah nilai dari satu tipe data ke tipe data lainnya seperti dari string ke int atau sebaliknya. Ada 2 tipe konversi yaitu:

##### **a. Implisit**

Python secara otomatis mengkonversikan tipe data ke tipe data lainnya tanpa ada campur tangan pengguna.

##### **b. 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.