PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

(RB)

Disusun oleh:

Muhammad Widyantoro Wiryawan 121140183

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2023

Modul 1

1. Pengenalan Bahasa Pemrograman Python

Python adalah bahasa pemrograman tingkat tinggi yang sering digunakan dalam pengembangan perangkat lunak, pemrosesan data, kecerdasan buatan, pengembangan aplikasi web, dan banyak lagi. Bahasa pemrograman ini dibuat pada tahun 1989 oleh Guido van Rossum, dan sejak itu menjadi salah satu bahasa pemrograman paling populer di dunia.

Python dirancang untuk mudah dipelajari dan mudah dibaca, sehingga menjadi bahasa pemrograman yang ideal bagi pemula yang ingin mempelajari dasar-dasar pemrograman. Bahasa pemrograman ini juga populer di kalangan pengembang profesional karena memiliki sintaks yang bersih dan jelas, serta dukungan yang luas untuk berbagai platform dan aplikasi.

Python memiliki beberapa fitur unik, seperti dukungan untuk pemrograman fungsional dan pemrograman berorientasi objek, serta dukungan untuk berbagai jenis tipe data dan modul yang dapat digunakan untuk memperluas kemampuan bahasa pemrograman. Python juga memiliki banyak pustaka dan kerangka kerja yang dapat digunakan untuk mempercepat proses pengembangan perangkat lunak.

2. Dasar pemrograman Python

a. Variable dan Tipe Data Primitif

Variabel adalah sebuah simbol atau nama yang digunakan untuk merepresentasikan nilai atau data tertentu dalam program komputer. Dalam pemrograman, variabel dapat digunakan untuk menyimpan data sementara atau untuk memperoleh nilai dari hasil perhitungan. Variabel dapat dianggap sebagai sebuah wadah atau kotak yang dapat berisi nilai atau data.

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

Contoh:

```
cek = True # Boolean
angka = 10 # int
desimal = 3.14 # float
kata = "ini adlah string" # string
```

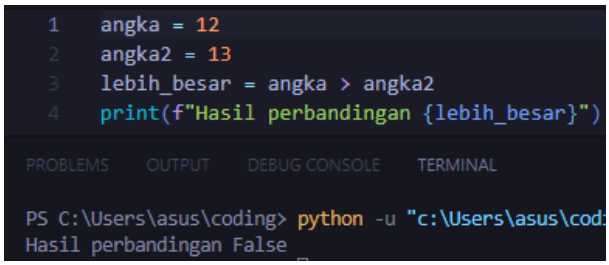
b. Operator

Operator adalah simbol atau tanda khusus yang digunakan untuk melakukan operasi aritmatika, perbandingan, logika, dan lain-lain pada nilai atau variabel. Operator digunakan untuk menggabungkan atau memanipulasi nilai dalam program Python.

- **Operator perbandingan**

Operator	Nama dan Fungsi	Contoh
>	Lebih besar dari – Hasilnya True jika nilai sebelah kiri lebih besar dari nilai sebelah kanan	x > y
<	Lebih kecil dari – Hasilnya True jika nilai sebelah kiri lebih kecil dari nilai sebelah kanan	x < y
==	Sama dengan – Hasilnya True jika nilai sebelah kiri sama dengan nilai sebelah kanan	x == y
!=	Tidak sama dengan – Hasilnya True jika nilai sebelah kiri tidak sama dengan nilai sebelah kanan	x != y
>=	Lebih besar atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih besar atau sama dengan nilai sebelah kanan	x >= y
<=	Lebih kecil atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih kecil atau sama dengan nilai sebelah kanan	x <= y

Contoh:



- **Operator penugasan**

Operator	Penjelasan	Contoh
=	Menugaskan nilai yang ada di kanan ke operand di sebelah kiri	**c = a + b** menugaskan a + b ke c
+=	Menambahkan 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**
-=	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	**c *= a** sama dengan **c = c * a**

	hasilnya ditugaskan ke operand yang di kiri	
/=	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	**c //= a** sama dengan **c = c // a**
%=	Melakukan operasi sisa bagi operand di kanan dengan operand di kiri dan hasilnya disimpan di operand yang di kiri	**c %= a** sama dengan **c = c % a**

Contoh:

```
1   angka = 12
2   angka2 = 13
3   angka2 += angka
4   print(f"Hasil penjumlahan {angka2}")

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\as
Hasil penjumlahan 25
```

- **Operator aritmatik**

Operator	Nama dan fungsi	Contoh
+	Penjumlahan, menjumlahkan 2 buah operand	x + y
-	Pengurangan, mengurangkan 2 buah operand	x – y
*****	Perkalian, mengalikan 2 buah operand	x * y
/	Pembagian, membagi 2 buah operand	x / y
****	Pemangkatan, memangkatkan bilangan	x **y
//	Pembagian bulat, menghasilkan hasil bagi tanpa koma	x // y
%	Modulus, menghasilkan sisa pembagian 2 bilangan	x % y

Contoh:

```
1   angka = 12
2   angka2 = 13
3   print(f"Hasil penjumlahan {angka} + {angka2} adalah {angka + angka2}")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\python\coolyeah_PBO\lat\temp
Hasil penjumlahan 12 + 13 adalah 25
```

- **Operator logika**

Operator	Penjelasan	Contoh
and	Hasilnya adalah True jika kedua operandnya bernilai benar	x and y
not	Hasilnya adalah True jika salah satu atau kedua operandnya bernilai benar	x or y
or	Hasilnya adalah True jika operandnya bernilai salah (kebalikan nilai)	not x

Contoh:

```
1   cek = 4
2   tes = 7
3   operator = tes and tes < cek
4   print(f"Hasil logika: {operator}")
5
6   operator2 = cek or cek > tes
7   print(f"Hasil logika: {operator2}")
8
9   operator3 = tes and tes > cek or tes and cek > tes
10  print(f"Hasil logika: {operator3}")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\pythor
Hasil logika: False
Hasil logika: 4
Hasil logika: True
```

- **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 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

Contoh:

```
1   num1 = 5
2   num2 = 9
3   bitwise = num1 | num2
4   print(f"Bitwise: {bitwise}")

PROBLEMS   OUTPUT   DEBUG CONSOLE   TER

PS C:\Users\asus\coding> python -u "c:\
Bitwise: 13
```

- **Operator identitas**

Operator	Penjelasan	Contoh
is	True jika kedua operand identic (menunjuk ke objek yang sama)	x is True
is not	True jika kedua operand tidak identik (tidak merujuk ke objek yang sama)	x is **not** True

Contoh:

```
1   num1 = 5
2   num2 = 9
3   num3 = 5
4   print("num1 is num2:", num1 is num2)
5   print("num1 is num3:", num1 is num3)
6   print("num2 is not num3:", num2 is not num3)

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding
num1 is num2: False
num1 is num3: True
num2 is not num3: True
```

- **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

Contoh:

```
a = "aku"
b = "aku lagi makan"
c = "saya"
print('a in b:', a in b)
print('c not in b:', c not in b)
```

```
PS C:\Users\asus\coding> python -u "c:\Use
a in b: True
c not in b: True
```

c. Tipe data bentukan

Ada 4, dengan perbedaan penggunaan:

- List
  Sebuah kumpulan data yang terurut, dapat diubah, dan memungkinkan ada anggota yang sama
  Contoh:

```
list = [1, 2, 3, 4, 5]
list2 = ["pisang", "apel", "jeruk"]
```

- Tuple
  Sebuah kumpulan data yang terurut, tidak dapat diubah, dan memungkinkan ada anggota yang sama
  Contoh:

```
ex = (1, 2, "belion")
```

- Set
  Sebuah kumpulan data yang tidak berurutan, tidak terindeks, dan tidak memungkinkan ada anggota yang sama
  Contoh:

```
x = {"anggur", "leci", "pepaya"}
```

- Dictionary
  Sebuah kumpulan data yang tidak berurutan, dapat diubah, tidak memungkinkan ada anggota yang sama
  Contoh:

```
dictionary = {'nama_depan' : 'widyantoro', 'nama_belakang' : 'wiryawan'}
```

d. Percabangan

- **if**

```
n = int(input("Masukan bilangan: "))
if n < 0:
    print(n, "bilangan negatif")
```

```
PS C:\Users\asus\coding> python -u "c:\Users\as
Masukan bilangan: -1
-1 bilangan negatif
```

Kode di atas hanya bisa menentukan bilangan negatif saja, tanpa menentukan bilangan lainnya yaitu positif dan nol.

- **if-else**

```
1  n = int(input("Masukan bilangan: "))
2  if n < 0:
3      print(n, "bilangan negatif")
4  else:
5      print(n, "bilangan positif")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\as
Masukan bilangan: 2
2 bilangan positif
```

Penambahan line 4-5 akan mengeluarkan keluaran untuk bilangan positif. Namun, kode di atas tidak dapat mengeluarkan keluaran bila pengguna memasukkan bilangan nol.

- **if-else-if**

```
1  n = int(input("Masukan bilangan: "))
2  if n < 0:
3      print(n, "bilangan negatif")
4  elif n == 0:
5      print(n, "bilangan nol")
6  else:
7      print(n, "bilangan positif")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\as
Masukan bilangan: 0
0 bilangan nol
```

Menambahkan line sebelum else dengan kondisi (n == 0) untuk mengeluarkan output “bilangan nol” ketika n bernilai 0.

- **Nested if**

Nested if adalah kondisi if yang ada di dalam kondisi if lainnya. Dalam nested if, blok kondisi if yang terdapat di dalam blok kondisi if lainnya akan dieksekusi hanya jika kondisi pada blok kondisi if luar terpenuhi.

```
1  n = int(input("Masukan bilangan: "))
2  if n < 0:
3      if n < -5:
4          print(n, "lebih kecil dari -5")
5      else:
6          print(n, "lebih besar dari -5")
7  elif n == 0:
8      print(n, "bilangan nol")
9  else:
10     print(n, "bilangan positif")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\
Masukan bilangan: -6
-6 lebih kecil dari -5
```

e. Perulangan

- **Perulangan for**

  Pada perulangan for biasa digunakkan untuk iterasi pada urutan berupa list, tuple, atau string.

  Contoh:

```
# contoh perulangan for pada list
x = ["ucup", "gehrman", "klein"]
for i in x:
    print(i)

# contoh perulangan for
print('contoh lain perulangan for')
for i in range(5):
    print(i)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\asus\coding> python -u "c:\Users\a
ucup
gehrman
klein
contoh lain perulangan for
0
1
2
3
4
```

- **Perulangan while**

  Perulangan while adalah struktur perulangan pada pemrograman yang akan terus dilakukan selama kondisi yang diberikan bernilai benar atau true.

  Contoh:

```
n = int(input("masukan angka: "))
while n != 0:
    n = int(input("masukan angka: "))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\asus\coding> python -u "c:\Users\asu
masukan angka: 1
masukan angka: 1
masukan angka: 2
masukan angka: 3
masukan angka: 4
masukan angka: 0
PS C:\Users\asus\coding>
```

f. Fungsi

Fungsi adalah blok kode yang terorganisir dan dapat digunakan kembali yang dirancang untuk melakukan tindakan tertentu. Fungsi digunakan untuk menghindari pengulangan kode, menyediakan modularitas, meningkatkan keterbacaan kode, dan memungkinkan pengujian dan debugging yang lebih mudah.

Contoh:

```python
def lolos(n):
    if n > 60:
        print('selamat anda lolos')
    else:
        print('maaf anda belum lolos')

n = int(input("masukan nilai anda: "))
lolos(n)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\
masukan nilai anda: 65
selamat anda lolos
```

Modul 2

1. Kelas

Kelas adalah sebuah blueprint atau cetakan untuk membuat objek-objek tertentu yang memiliki sifat-sifat dan perilaku yang sama. Kelas digunakan untuk mengelompokkan fungsi-fungsi dan 10ariable-variabel ke dalam satu kesatuan yang disebut objek.

Contoh:

```python
class Hero:
    def __init__(self, inputName, InputHealth, inputAttack, inputArmor):
        self.name = inputName
        self.health = InputHealth
        self.attack = inputAttack
        self.armor = inputArmor

hero = Hero("Eizer", 100, 23, 50)
print(hero.name)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\python\pbo_KT\oop#2.py"
Eizer
```

Dalam kelas, kita dapat mendefinisikan atribut-atribut (variabel) dan metode-metode (fungsi) yang berkaitan dengan objek tersebut. Setelah sebuah kelas didefinisikan, kita dapat membuat objek-objek dari kelas tersebut dengan menggunakan operator new.

a. Atribut

Atribut merujuk pada variabel yang disimpan di dalam suatu objek. Atribut ini bisa diberikan nilai saat objek dibuat dan nilainya dapat berubah selama program dijalankan. Atribut juga dapat diakses dan dimodifikasi melalui metode yang ada pada kelas objek tersebut. Dalam hal ini, atribut dapat digunakan sebagai karakteristik atau informasi tentang objek, yang berguna dalam proses pemrograman.

Contoh:

```python
class Hero:
    def __init__(self, inputName, InputHealth, inputAttack, inputArmor):
        self.name = inputName
        self.health = InputHealth
        self.attack = inputAttack
        self.armor = inputArmor

hero = Hero("Eizer", 100, 23, 50)
print("Nama hero:",hero.name)
print("Health hero:",hero.health)
print("Attack hero:",hero.attack)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\python\pbo_KT\oop#2.py"
Nama hero: Eizer
Health hero: 100
Attack hero: 23
```

b. Method

Method adalah sebuah fungsi yang didefinisikan di dalam sebuah kelas dan bertugas untuk memanipulasi objek dari kelas tersebut. Method bisa dianggap sebagai tindakan atau perilaku dari suatu objek. Method juga bisa digunakan untuk mengakses dan memanipulasi atribut dari suatu objek atau kelas.

Contoh:

```python
class Hero:
    def __init__(self, inputName, inputArmor, inputAttack, InputHealth):
        self.name = inputName
        self.armor = inputArmor
        self.health = InputHealth
        self.attack = inputAttack

    def method(self):
        print('nama hero adalah ' + self.name)

hero = Hero("Jajang", 120, 15, 200)
hero.method()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\python\coolyeah_PBO\lat\tempCo
nama hero adalah Jajang
```

2. Objek

Objek adalah sebuah instance atau realisasi dari sebuah kelas (class). Objek memiliki atribut yang mewakili sifat-sifat atau karakteristiknya, serta metode yang merepresentasikan perilaku atau aksi yang dapat dilakukan oleh objek tersebut.

Contoh:

```python
ashborn = Hero("Ashborn", 100, 10)
```

3. Magic method

Magic method adalah metode khusus dalam Python yang memungkinkan kita untuk menentukan perilaku objek yang sudah ada, seperti operator aritmatika atau perbandingan, dengan cara mendefinisikan metode yang memiliki nama khusus yang diawali dan diakhiri dengan double underscore, misalnya “__add__” atau “__eq__”.

Magic method memungkinkan kita untuk menyesuaikan bahasa Python dengan kebutuhan kita dalam penggunaan objek. Beberapa contoh magic method yang sering digunakan adalah “__init__” untuk menginisialisasi objek saat pertama kali dibuat, “__str__” untuk mengubah representasi objek menjadi string, dan “__eq__” untuk membandingkan kesamaan antara dua objek.
Contoh:

```python
class Mangga:
    def __init__(self, jenis, jumlah):
        self.jenis = jenis
        self.jumlah = jumlah

    def __str__(self):
        return f"Mangga {self.jenis}, jumlah {self.jumlah}"

belanja = Mangga("Arum manis", 10)
print(belanja)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\python\coolyeah
Mangga Arum manis, jumlah 10
```

4. Konstruktor

Konstruktor atau constructor adalah sebuah method khusus yang akan dieksekusi secara otomatis saat sebuah objek dibuat. Konstruktor biasanya digunakan untuk menginisialisasi nilai awal pada atribut objek agar objek tersebut sudah bisa digunakan dengan baik. Dalam bahasa pemrograman Python, konstruktor ditandai dengan method init() yang diikuti dengan parameter self dan parameter lain yang dibutuhkan untuk menginisialisasi objek.
Contoh:

```python
class Hero:
    # ini adalah konstruktor
    def __init__(self, inputName, InputHealth, inputAttack, inputArmor):
        self.name = inputName
        self.health = InputHealth
        self.attack = inputAttack
        self.armor = inputArmor

    # ini adalah method
    def ex(self):
        print('ini adalah method')
```

5. Destructor

Destructor dalam bahasa pemrograman adalah sebuah method khusus yang dipanggil saat sebuah objek dihapus dari memori. Destructor biasanya digunakan untuk membersihkan sumber daya yang digunakan oleh objek tersebut. Di Python, destructor direpresentasikan dengan method “__del__”.
Contoh:

```python
class MyClass:
    def __init__(self, n):
        self.n = n

    def __del__(self):
        print(f"objek {self.n} telah dihapus")

obj = MyClass(9)  # Constructor dipanggil
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\p
objek 9 telah dihapus
```

6. Setter dan getter

Setter dan Getter adalah metode yang digunakan dalam OOP untuk mengatur dan mendapatkan nilai variabel dalam kelas. Setter digunakan untuk menetapkan nilai ke variabel kelas, sedangkan Getter digunakan untuk mengambil nilai dari variabel kelas.
Contoh:

```python
class Orang:
    def __init__(self, name = "ucup"):
        self.__name = name

    def setName(self, name):
        self.__name = name

    def getName(self):
        return self.__name

ucup = Orang()
ucup.setName("Bambang")
print(ucup.getName())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus
Bambang
```

7. Decorator

Decorator adalah sebuah fitur di Python yang digunakan untuk memodifikasi fungsi atau metode pada saat deklarasi dengan menambahkan fungsionalitas tambahan pada fungsi atau metode tersebut. Decorator dapat memodifikasi fungsi tanpa harus mengubah kode fungsi asli dan digunakan untuk mengembangkan dan memodifikasi kode dengan mudah.
Contoh:

```python
class Orang:
    def __init__(self, name = "ucup"):
        self.__name = name

    @property
    def setName(self):
        print("method setName dipanggil")
        return self.__name

    @setName.setter
    def setName(self, name):
        print('setter setName dipanggil')
        self.__name = name

ucup = Orang()
ucup.setName = "Ashborn"
print(ucup.setName)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\co
setter setName dipanggil
method setName dipanggil
Ashborn
```

Modul 3

1. Abstraksi

Abstraksi pada proses penyederhanaan kompleksitas sistem dengan cara memperhatikan hanya aspek-aspek penting dari objek dan menyembunyikan detail implementasinya dari pengguna. Hal ini dilakukan untuk mempermudah penggunaan dan pengembangan program.

Contoh:

```python
class Karyawan:
    def __init__(self, nama, gaji):
        self.nama = nama
        self.gaji = gaji

    def setGaji(self, plus):
        self.gaji += plus
        return self.gaji

    def info(self):
        return f"Nama karyawan adalah {self.nama} dengan gaji {self.gaji:,}"

awan = Karyawan("Awan", 5000000)
awan.setGaji(1000000)
print(awan.info())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\python\coolyeah_PBO\lat\tempCodeRu
Nama karyawan adalah Awan dengan gaji 6,000,000
```

2. Enkapsulasi

- **Public**

  Variabel public dalam pemrograman adalah variabel yang dapat diakses dan diubah oleh seluruh program atau objek lain yang memiliki akses ke variabel tersebut. Variabel public biasanya didefinisikan di luar fungsi atau kelas dan dapat diakses dengan menggunakan objek atau nama variabelnya secara langsung.

  Contoh:

```python
class Karyawan:
    def __init__(self, nama, gaji):
        self.nama = nama
        self.gaji = gaji

    def info(self):
        return f"Nama karyawan adalah {self.nama} dengan gaji {self.gaji:,}"

awan = Karyawan("Awan", 5000000)
print(awan.info())
awan.gaji = 7000000
print(awan.info())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\asus\coding> python -u "c:\Users\asus\coding\python\coolyeah_PBO\lat\tempCodeRu
Nama karyawan adalah Awan dengan gaji 5,000,000
Nama karyawan adalah Awan dengan gaji 7,000,000
```

- **Protected**

  Protected variable adalah variabel yang hanya dapat diakses dari dalam kelas dan juga kelas turunannya. Dalam Python, variabel protected didefinisikan dengan menambahkan underscore tunggal di depan nama variabel. Sebagai contoh:

```python
class Karyawan:
    def __init__(self, nama, gaji):
        self._nama = nama
        self._gaji = gaji

    def info(self):
        return f"Nama karyawan adalah {self._nama} dengan gaji {self._gaji:,}"
```

- **Private**

  Private variable adalah jenis variabel dalam OOP yang hanya dapat diakses dari dalam kelas yang sama. Untuk menandai variabel sebagai private dalam Python, nama variabel harus diawali dengan underscore ("__"). Sebagai contoh:

```python
class Karyawan:
    def __init__(self, nama, gaji):
        self.__nama = nama
        self.__gaji = gaji

    def info(self):
        return f"Nama karyawan adalah {self.__nama} dengan gaji {self.__gaji:,}"
```

- **Setter dan Getter**
  Setter dan getter adalah metode atau fungsi dalam pemrograman objek yang digunakan untuk mengakses dan mengubah nilai variabel (atribut) dalam sebuah objek. Setter digunakan untuk mengubah nilai suatu atribut, sedangkan getter digunakan untuk mengambil nilai suatu atribut.
  Contoh:

Modul 4

1. Inheritance

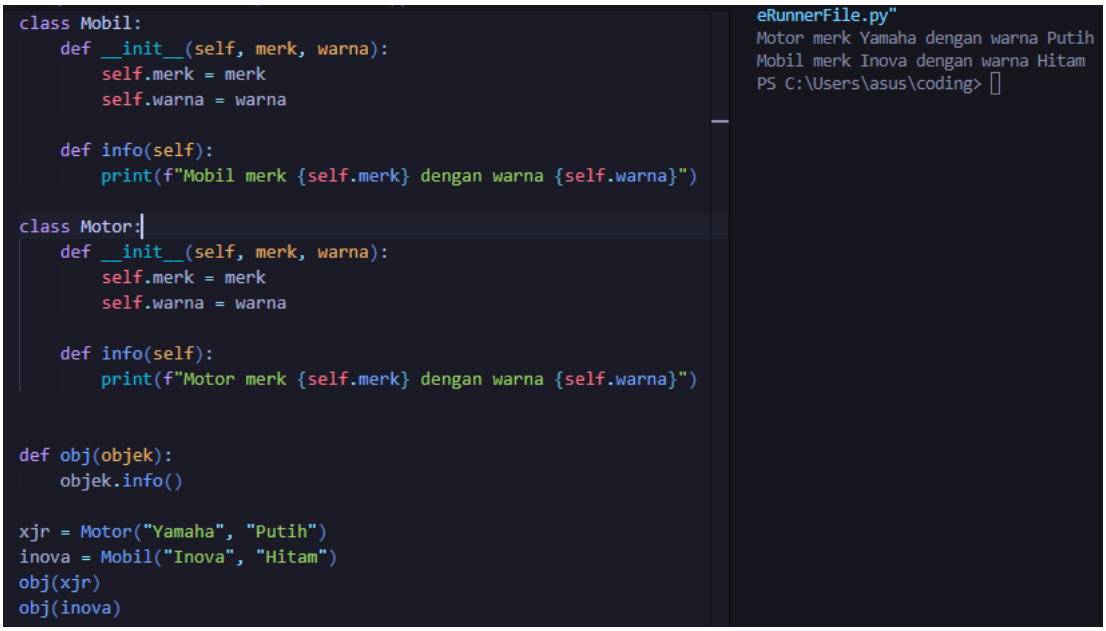
Inheritance adalah konsep dalam OOP (Object-Oriented Programming) di mana sebuah kelas baru dapat dibuat dengan mewarisi properti dan metode dari kelas yang sudah ada. Kelas yang diwarisi disebut sebagai kelas induk (parent class atau superclass), sementara kelas yang mewarisi disebut kelas anak (child class atau subclass).
Contoh:

```
class Kendaraan:
    def __init__(self, jenis, warna):
        self.jenis = jenis
        self.warna = warna

    def info(self):
        print(f"Kendaraan jenis {self.jenis} dengan warna {self.warna}")

class Mobil(Kendaraan):
    def __init__(self, jenis, warna, merek):
        super().__init__(jenis, warna)
        self.merek = merek

    def info(self):
        print(f"Mobil merek {self.merek} dengan jenis {self.jenis} dan warna {self.warna}")
```

2. Polymorphism

Polymorphism adalah kemampuan untuk memerlakukan objek dengan cara yang sama, meskipun objek tersebut berasal dari kelas yang berbeda. Dalam OOP, hal ini dicapai melalui penggunaan metode dengan nama yang sama dalam kelas yang berbeda.

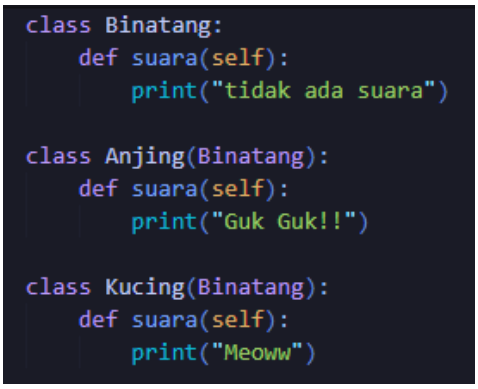
Contoh:



3. Override

Override adalah suatu teknik di mana sebuah metode yang sudah ada pada kelas induk (parent class) didefinisikan kembali pada kelas turunan (child class) dengan tujuan untuk menimpa implementasi metode di kelas induk dengan implementasi yang berbeda pada kelas turunan. Dengan override, kita bisa membuat perilaku yang berbeda untuk metode yang sama pada kelas-kelas yang berbeda.

Contoh:



4. Overloading

Overloading dalam bahasa pemrograman adalah kemampuan untuk mendefinisikan beberapa metode atau fungsi dengan nama yang sama, tetapi memiliki jumlah atau jenis parameter yang berbeda.

Contoh:

```python
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return Point(x, y)

p1 = Point(2, 3)
p2 = Point(-1, 2)
p3 = p1 + p2
print(p3.x, p3.y)
```

5. Multiple Inheritance

Multiple Inheritance adalah konsep di mana sebuah kelas dapat diwarisi dari lebih dari satu kelas induk atau superclass. Ini memungkinkan kelas anak untuk memiliki sifat dari lebih dari satu kelas induk dan memungkinkan penggunaan kembali kode yang lebih fleksibel.

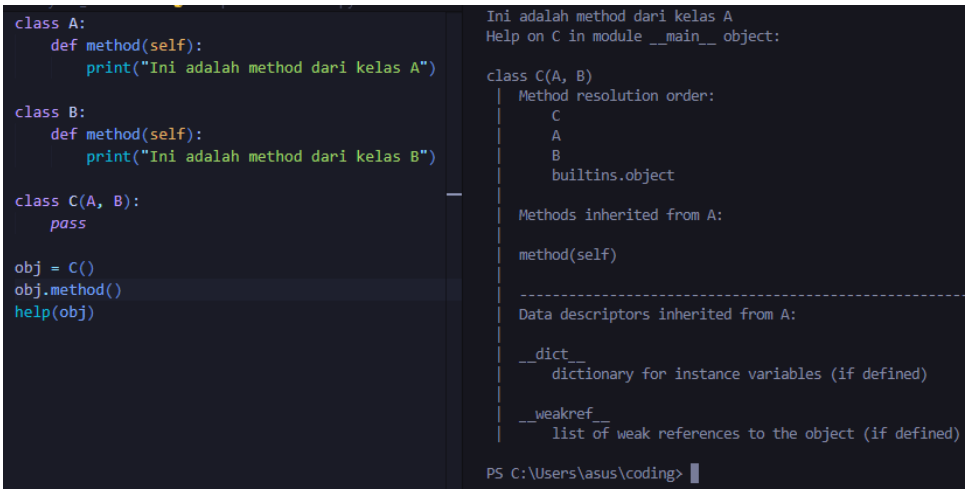
Contoh:

```python
class A:
    def method_A(self):
        print("Ini adalah method dari kelas A")

class B:
    def method_B(self):
        print("Ini adalah method dari kelas B")

class C(A, B):
    pass
```

6. Method Resolution Order

Method Resolution Order (MRO) adalah urutan pencarian metode dalam pewarisan banyak kelas atau multiple inheritance pada bahasa pemrograman Python. MRO menentukan urutan kelas mana yang akan dicari terlebih dahulu saat sebuah metode dipanggil pada objek yang berasal dari kelas yang memiliki banyak superclass. Untuk mengetahui urutannya gunakan perintah help().

Contoh:


class A:
def method(self):
print("Ini adalah method dari kelas A")
class B:
def method(self):
print("Ini adalah method dari kelas B")
class C(A, B):
pass
obj = C()
obj.method()
help(obj)
Ini adalah method dari kelas A
Help on C in module __main__ object:
class C(A, B)
Method resolution order:
C
A
B
builtins.object
Methods inherited from A:
method(self)
Data descriptors inherited from A:
__dict__
dictionary for instance variables (if defined)
__weakref__
list of weak references to the object (if defined)
PS C:\Users\asus\coding>