RESUME PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK (RB)
PERTEMUAN 1 - 4

Oleh :

Vito Anwar (121140074)

Program Studi Teknik Informatika

Institut Teknologi Sumatera

2022

Daftar Isi

Pertemuan 1

Pengenalan dan Dasar Pemrograman Python I

A. Pengenalan Bahasa Python

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

Dalam dokumen The Zen of Python (PEP 20), terdapat filosofi inti dari bahasa pemrograman Python, yang menekankan pentingnya kesederhanaan dan readability dalam kode. Beberapa poin dalam filosofi tersebut antara lain "Beautiful is better than ugly", "Explicit is better than implicit", "Simple is better than complex", "Complex is better than complicated", dan "Readability counts". Python sangat mementingkan readability pada kode, dan untuk mengimplementasikan filosofi tersebut, Python tidak menggunakan kurung kurawal ({}) atau keyword (seperti start, begin, end) sebagai gantinya menggunakan spasi (white space) untuk memisahkan blok-blok kode.

B. 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 (\).

- Baris dan Indentasi

```
1  mobil, motor, sepeda = 1, 1, 1
2
3  kendaraan = mobil + motor + sepeda
4
5  print(kendaraan)
```

```
3
```

Python tidak menggunakan kurung kurawal sebagai grouping blok kode melainkan menggunakan spasi ataupun tab (4 spasi). kode yang berada di blok yang sama harus memiliki jumlah spasi yang sama di awal. Contoh salah :

```
1  mobil, motor, sepeda = 1, 1, 1
2
3  kendaraan = mobil + motor + sepeda
4
5     print(kendaraan)
```

```
  File "/home/main.py", line 5
    print(kendaraan)
IndentationError: unexpected indent
```

b. 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, tipe data yang ada :

Tipe Data	Jenis	Nilai
Bool	Boolean	True atau False
Int	Bilangan bulat	Seluruh bilangan bulat
float	Bilangan real / desimal	Seluruh bilangan real
string	Teks	Kumpulan karakter

Contoh deklarasi dalam program :

```
1  boolean = True
2  integer = 21
3  inifloat = 3.14
4  inistring = "hai dunia"
```

c. Operator

- Operator Aritmatika

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

Operator	Nama dan Fungsi	Contoh
+	Penjumlahan, untuk melakukan operasi pemjumlahan nilai.	a + b
-	Pengurangan, untuk melakukan operasi pengurangan nilai.	a – b
*	Perkalian, untuk melakukan operasi perkalian oleh suatu nilai.	a * b
/	Pembagian, untuk melakukan operasi pembagian oleh suatu nilai,	a / b
**	Pemangkatan, untuk melakukan operasi perhitungan pangkat bilangan.	a ** b
//	Pembagian bulat, untuk melakukan pembagian dengan hasil bilangan bulat.	a // b
%	Modulus, untuk mencari nilai dari sisa bagi suatu nilai.	a % b

Contoh dalam program :

```
1  angka1 = int(input("masukan angka ke-1 : "))
2  angka2 = int(input("masukan angka ke-2 : "))
3  print("hasil penjumlahan : ", angka1 + angka2)
4  print("hasil pengurangan : ", angka1 - angka2)
5  print("hasil perkalian : ", angka1 * angka2)
6  print("hasil pembagian : ", angka1 / angka2)
7  print("hasil pangkat : ", angka1 ** angka2)
8  print("hasil pembagian bulat : ", angka1 // angka2)
9  print("hasil modulus : ", angka1 % angka2)
```

```
masukan angka ke-1 : 7
masukan angka ke-2 : 2
hasil penjumlahan :  9
hasil pengurangan :  5
hasil perkalian :  14
hasil pembagian :  3.5
hasil pangkat :  49
hasil pembagian bulat :  3
hasil modulus :  1
```

- Operator Perbandingan

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

Operator	Nama dan Fungsi	Contoh
>	Lebih besar dari : nilai True jika nilai kiri lebih besar dari nilai kanan.	a > b
<	Lebih kecil dari : nilai True jika nilai kiri lebih kecil dari nilai kanan.	a < b
==	Sama dengan : nilai True jika nilai kiri dan nilai kanan memiliki nilai yang sama.	a == b
!=	Tidak sama dengan : nilai True jika nilai kiri dan nilai kanan tidak sama.	a != b
>=	Lebih besar atau samadengan : nilai True jika nilai kiri sama atau lebih besar dari nilai kanan.	a >= b
<=	Lebih kecil atau samadengan : nilai True jika nilai kiri sama atau lebih kecil dari nilai kanan.	a <= b

- 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.

Operator	Penjelasan	Contoh
=	Menugaskan nilai sebelah kiri akan menjadi sama dengan nilai sebelah nilai kanan.	a = b nilai a akan menjadi sama dengan nilai b
+=	Nilai sebelah kiri akan memiliki nilai dari penjumlahan nilai kiri dan nilai kanan.	a += b sama dengan a = a+b
-=	Nilai sebelah kiri akan memiliki nilai dari pengurangan nilai kiri dan nilai kanan.	a -= b sama dengan a = a-b
*=	Nilai sebelah kiri akan memiliki nilai dari perkalian nilai kiri dan nilai kanan.	a *= b sama dengan a = a*b
/=	Nilai sebelah kiri akan memiliki nilai dari pembagian nilai kiri dan nilai kanan.	a /= b sama dengan a = a/b
**=	Nilai sebelah kiri akan memiliki nilai dari perpangkatan nilai kiri pangkat nilai kanan.	a **= b sama dengan a = a**b
//=	Nilai sebelah kiri akan memiliki nilai dari pembagian bulat dari nilai kiri dan nilai kanan.	a //= b sama dengan a = a//b
%=	Nilai sebelah kiri akan memiliki nilai modulus dari nilai kiri dan nilai kanan.	a %= b sama dengan a = a%b

- Operator Logika

  Operator logika adalah operator yang digunakan untuk melakukan operasi logika.

Operator	Penjelasan	Contoh
and	Hasil True jika kedua kondisi bernilai benar.	a and b
or	Hasil True jika salah satu kondisi bernilai benar.	a or b
not	Hasilnya True jika kondisi bernilai salah (kebalikan)	not a

- 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.

Operator	Nama	Contoh
&	Bitwise AND	a & b = 0 (0000 0000)
|	Bitwise OR	a | b = 14 (0000 1110)
~	Bitwise NOT	~a = -11 (1111 0101)
^	Bitwise XOR	a ^ b = 14 (0000 1110)
>>	Bitwise right shift	a >> 2 = 2 (0000 0010)
<<	Bitwise left shift	a << 2 = 40 (0010 1000)

- Operator Identitas

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

Operator	Penjelasan	Contoh
is	True jika kedua operand identik (menunjuk ke nilai yang sama)	a is True
is not	True jika kedua operand tidak sama (tidak menunjuk ke nilai yang sama)	a is not True

- 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).

Operator	Penjelasan	Contoh
in	True jika nilai/variabel ditemukan didalam data.	6 in a
not in	True jika nilai/variabel tidak ditemukan didalam data.	6 not in a

d. Tipe Data Bentukan

- List

  Sebuah kumpulan data yang terurut, dapat diubah, dan memungkinkan ada anggota yang sama. Contoh [1, 2, 3, 4, 5] [‘apple’, ‘banana’, ‘cherry’] atau [‘xyz’,768, 2.23]

- Tuple

  Sebuah kumpulan data yang terurut, tidak dapat diubah, dan memungkinkan ada anggota yang sama. Contoh (‘xyz’, 1, 3.14)

- Set

  Sebuah kumpulan data yang tidak berurutan, tidak terindeks, dan idak memungkinkan ada anggota yang sama. Contoh { ‘apple’, ‘banana’,‘cherry’ }

- Dictionary

  Sebuah kumpulan data yang tidak berurutan, dapat diubah, tidak memungkinkan ada anggota yang sama. Contoh {‘firstName’:‘Adelia’, ‘lastName’: ‘Natasyah’}

e. Percabangan

Dalam bahasa pemrograman Python, terdapat beberapa percabangan yaitu IF, IF-ELSE, dan IF-ELSE-IF.

- Percabangan IF

  Percabangan ini hanya memiliki satu kondisi, seberpi contoh dibawah ini :

```
1  n = int(input("masukan nilai n : "))
2
3  if n > 0 :
4     print("bilangan positif!")
```

```
masukan nilai n : 6
bilangan positif!
```

- Percabangan IF-ELSE

  Percabangan ini memiliki 2 kondisi, yaitu true dan false, keduanya akan menjalankan aksi yang berbeda, seperti contoh dibawah ini :

```
1  n = int(input("masukan nilai n : "))
2
3  if n > 0 :
4     print("bilangan positif!")
5  else :
6     print("negatif atau nol !")
```

```
masukan nilai n : 0
negatif atau nol !
```

- Percabangan IF-ELSE-IF

  Percangangan ini memiliki lebih dari 2 kondisi seperti contoh dibawah ini :

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

if n > 0 :
   print("bilangan positif!")
elif n < 0 :
   print("bilangan negatif!")
else :
   print("bilangan nol !")
```

```
masukan nilai n : -8
bilangan negatif!
```

- Nested IF

  Percabangan ini adalah percabangan bersarang atau percabangan didalam percabangan, seperti contoh dibawah ini :

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

if n > 0 :
   print("bilangan positif!")
   if n%2 == 0 :
      print("genap")
```

```
masukan nilai n : 6
bilangan positif!
genap
```

f. Perulangan

Dalam python terdapat dua jenis perulangan , yaitu perulangan for dan perulangan while. Dalam implementasi nya perulangan digunakkan jika ingin mengulang sesuatu sebanyak n kali.

- Perulangan for

  Berikut ini adalah contoh sederhana perulangan for :

```
n = int(input("masukan batas : "))

for i in range (1, n, 1):
   print(i)

```

```
masukan batas : 4
1
2
3
```

- Perulangan While

Berikut ini adalah contoh sederhana perulangan while :

```
1  n = int(input("masukan batas : "))
2
3  i = 0
4
5  while i < n :
6      print(i+1)
7      i += 1
```

```
masukan batas : 2
1
2
```

g. Fungsi

Dengan fungsi kita bisa melakukan perintah dengan sebuah kumbulan blok kode, tanpa harus menulisnya secara berulang-ulang. Contoh :

```
1  def perkalian(a, b):
2      hasil = a * b
3      print("perkalian", a, "dengan", b, "hasilnya adalah :", hasil)
4
5  perkalian(8, 5)
```

```
input
perkalian 8 dengan 5 hasilnya adalah : 40
```

Jadi untuk menjalankan blok kode yang ada didalam fungsi, kita hanya perlu memanggil nama fungsinya, dan jika fungsi memasukan nilai, maka kita harus memberikan nilainya.

Pertemuan 2

Objek dan Kelas Dalam Python (Konstruktor, Setter, Dan Getter)

A. 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. Untuk membuat kelas, gunakan kata kunci class diikuti oleh nama kelas tersebut dan tanda titik dua. Contoh:

```
1  Class Motor :
2      def __init__ (self, warna, tahun):
3          self.warna = warna
4          self.tahun = tahun
5
6  motor = Motor("Hitam", 2020)
```

Program diatas merupakan cara untuk membuat sebuah kelas. Dapat dilihat diatas terdapat __init__(), method tersebut adalah konstruktor untuk membuat kelas motor. Kita juga bisa membuat kelas kosong dengan menambahkan kata kunci *pass*.

```
1  Class Motor :
2      pass
3
4  motor = Motor()
```

motor adalah objek untuk kelas Motor.

a. Atribut/Properti

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. Atribut kelas merupakan sifat yang dimiliki oleh sebuah kelas dan juga akan dimiliki oleh setiap objek. Sedangkan atribut objek adalah sebuah atribut dari masing - masing objek. Berikut adalah contoh pendeklarasian nya :

```
1  class Motor :
2      jumlah_roda = 2
3
4      def __init__ (self, warna, tahun):
5          self.warna = warna
6          self.tahun = tahun
7
8  motor = Motor("Hitam", 2020)
9
10 print(motor.warna)
11 print(motor.tahun)
12 print(motor.jumlah_roda)
```

```
Hitam
2020
2
```

Pada contoh diatas jumlah_roda merupakan atribut kelas dan untuk atribut objek adalah warna dan tahun.

b. 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. Misalkan objek motor dapat berjalan_maju, berhenti.

```
class Motor :
    jumlah_roda = 2

    def __init__ (self, warna, tahun):
        self.warna = warna
        self.tahun = tahun

    def berjalan_maju(self):
        print("motor berjalan maju")

motor = Motor("Hitam", 2020)

motor.berjalan_maju()
```

```
motor berjalan maju
```

Pada contoh diatas yang merupakan method adalah berjalan_maju, jadi kita bisa memanggilnya dan aksi didalamnya akan dijalankan.

B. 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 ().

C. Magic Method

Magic method adalah metode yang diawali dan diakhiri dengan double underscore (dunder). Method ini tidak dipanggil secara langsung, tapi dipanggil sistem secarainternal ketika melakukan sesuatu seperti menggunakan operator tambah (__add__), membuat objek (__init__), dan lain-lain.

Tujuan utama dari penggunaan magic method adalah untuk mengubah sifat (behavior) bawaan dari suatu objek. Untuk melihat apa saja magic method bawaan python bisa menggunakan sintaks dir(int).

Magic method tidak terbatas pada kelas int saja, tetapi ada di setiap jenis objek dan variabel. Salah satu contoh magic method yaitu __add__(). Method ini ditambahkan agar user dapat melakukan operasi penambahan (+) secara langsung pada objek, tanpa mengakses atribut isi objek.

D. Konstruktor

Konstruktor adalah method yang “pasti” dijalankan secara otomatis pada saat sebuah objek dibuat untuk mewakili kelas tersebut. Seperti dengan method lain, konstruktor dapat melakukan operasi seperti melakukan print. Selain operasi method dasar, konstruktor dapat menerima argumen yang diberikan ketika objek dibuat. Argumen ini akan diproses di dalam kelas nantinya. Contoh dalam program :

```
class Motor :
    #ini constructor
    def __init__ (self, warna, tahun):
        self.warna = warna
        self.tahun = tahun

    def berjalan_maju(self):
        print("motor warna", self.warna, "berjalan maju")

motor = Motor("Hitam", 2020)

motor.berjalan_maju()
```

input

```
motor warna Hitam berjalan maju
```

E. Destruktor

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

F. Setter and 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. Contoh dalam program :

```
class Motor :

    def __init__ (self, tahun = 0):
        self.tahun = tahun

    #getter method
    def get_tahun(self):
        return self.tahun

    #setter method
    def set_tahun(self, n):
        self.tahun = n

motor = Motor()

#setting tahun using setter
motor.set_tahun(2021)
#retrieving tahun using getter
print(motor.get_tahun())
```

```
2021
```

G. 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 pada program :

```
class Motor :

    def __init__ (self, tahun = 0):
        self.__tahun = tahun

    @property
    def tahun(self):
        print("fungsi getter tahun dipanggil")
        return self.__tahun

    @tahun.setter
    def tahun(self, n):
        print("fungsi setter tahun dipanggil")
        self.__tahun = n

motor = Motor()


print(motor.tahun)
motor.tahun += 2022
print(motor.tahun)
```

```
0
fungsi getter tahun dipanggil
fungsi setter tahun dipanggil
fungsi getter tahun dipanggil
2022
```

Pertemuan 3

Abstraksi dan Enkapsulasi (Visibilitas Fungsi dan Variabel, Relasi Antar Kelas)

A. 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. User mengetahui apa yang objek lakukan, tapi tidak tau mekanisme yang terjadi di belakang layar bagaimana. Contohnya ketika mengendarai mobil, user mengetahui bagaimana menyalakan mobil, menjalankan, menghentikan, dll, tetapi tidak mengetahui mekanisme apa yang terjadi pada mobil ketika mendapat perintah di atas. Ketika mendefinisikan kelas, sebenarnya sedang membuat Abstrak dari suatu Objek. Kelas merupakan bentuk abstrak atau cetak biru (blue print) dari suatu objek nyata. Wujud nyata suatu kelas dinamakan instance (Objek). Contoh : Str adalah kelas, sedangkan teks “Python” adalah objek.

B. Enkapsulasi (Encapsulation)

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. Dengan konsep ini, kita dapat “menyembunyikan” property dan method dari suatu kelas agar hanya property dan method tertentu saja yang dapat diakses dari luar kelas. Dengan menghalangi akses dari luar kelas tersebut, elemen penting yang terdapat dalam kelas dapat lebih terjaga, dan menghindari kesalahan jika elemen tersebut diubah secara tidak sengaja.

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

a. Public Acces Modifier

Pada umumnya ketika kita mendeklarasikan suatu variabel atau method, maka itulah public access modifier. Setiap class, variable dan method yang dibuat secara default merupakan public.

Contoh Program :

```
1  class Motor:
2      def __init__(self, warna, tahun):
3          self.warna = warna
4          self.tahun = tahun
5
6      def cek_data(self):
7          print("warna motor :", self.warna)
8          print("keluaran tahun :", self.tahun)
```

b. Protected Acces Modifier

Jika suatu variabel dan method dideklarasikan secara protected, maka variable dan method tersebut hanya dapat diakses oleh kelas turunan darinya. Cara mendeklarasikannya dengan menambahkan 1 underscore (_) sebelum variable atau method.

Contoh Program :

```
1  class Motor:
2      _roda = 2
3
4      def __init__(self, warna, tahun):
5          self._warna = warna
6          self._tahun = tahun
7
8      def _cek_data(self):
9          print("warna motor :", self._warna)
10         print("keluaran tahun :", self._tahun)
```

c. Private Acces Modifier

Jika suatu variabel dan method dideklarasikan secara private, maka variable dan method tersebut hanya dapat diakses di dalam kelas itu sendiri, private access merupakan yang paling aman. Dalam mendeklarasikannya, hanya perlu menambahkan double underscore ( ) sebelum nama variable dan methodnya.

Contoh Program :

```
1  class Motor:
2      __roda = 2
3
4      def __init__(self, warna, tahun):
5          self.__warna = warna
6          self.__tahun = tahun
7
8      def _cek_data(self):
9          print("warna motor :", self.__warna)
10         print("keluaran tahun :", self.__tahun)
```

d. Setter and Getter

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. Dikarenakan property dengan access modifier private hanya dapat diakses dari dalam kelas tersebut, dengan metode inilah kita dapat mengaksesnya dari luar kelas. Contoh program ada di halaman 12.

Pertemuan 4

Pewarisan dan Polimorfisme (Overloading, Overriding, Dynamic Cast)

A. 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 "Kendaraan" dan terdapat class "Mobil" yang merupakan turunan dari class "Kendaraan". ini berarti class "Mobi;" memiliki atribut dan method yang sama dengan class "Kendaraan". Dan objek "Mobil" dapat menggantikan objek "Kendaraan" dalam aplikasi.

Syntaks dasar inheritance :

```
class Kendaraan:
    pass

class Motor(Kendaraan):
    pass
```

Contoh Program :

```
class Kendaraan:
    def __init__(self, kmh, tahun):
        self.kmh = kmh
        self.tahun = tahun

    def cetak_data(self):
        print("kendaraan ini keluaran tahun ", self.tahun)
        print("memiliki kecepatan ", self.kmh, "kmh")

class Motor(Kendaraan):
    pass

motor = Motor(125, "2017")
motor.cetak_data()
```

input

```
kendaraan ini keluaran tahun  2017
memiliki kecepatan  125 kmh
```

B. 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. Polymorphism merupakan kemampuan suatu method untuk bekerja dengan lebih dari satu tipe argumen. Pada bahasa lain (khususnya C++), konsep ini sering disebut dengan method overloading. Pada dasarnya, Python tidak menangani hal ini secara khusus. Hal ini disebabkan karena Python merupakan suatu bahasa pemrograman yang bersifat duck typing(dynamic typing).

Contoh Program :

```
class motor():
    def type(self):
        print("kendaraan")
    def color(self):
        print("hitam")

class laptop():
    def type(self):
        print("elektronik")
    def color(self):
        print("silver")

def func(obj):
    obj.type()
    obj.color()

obj_motor = motor()
obj_laptop = laptop()
func(obj_motor)
func(obj_laptop)
```

C. 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*.

Contoh Program :

```
class Kendaraan():
    def cetak(self):
        print("ini kendaraan")

class Motor(Kendaraan):
    def cetak(self):
        print("ini motor")

motor = Motor()
motor.cetak()
```

```
ini motor
```

D. 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”. Secara umum overloading memiliki beberapa signature, yaitu jumlah argumen, tipe argumen, tipe keluaran dan urutan argumen. Akan tetapi, seperti yang telah dijelaskan python tidak menangani overloading secara khusus. Karena python adalah bahasa pemrograman yang bersifat duck typing (dynamic typing), overloading secara tidak langsung dapat diterapkan.

Contoh Program :

```
class Robot:
    def __init__(self, nama):
        self.nama = nama
    def suara(self):
        print("roger..!!")

class Alien:
    def __init__(self, name):
        self.name = name
    def suara(self):
        print("i can roger too....!!")

def suara(obj):
    obj.suara()

robocop = Robot("robocop")
lien = Alien("anuaki")
suara(robocop)
suara(lien)
```

```
roger..!!
i can roger too....!!
```

E. Multiple Inheritance

Python mendukung pewarisan ke banyak kelas. Kelas dapat mewarisi dari banyak orang tua.

Syntax dasar multiple inheritance adalah sebagai berikut :

```
class Motor:
    pass

class Mobil:
    pass

class kendaraan(Motor, Mobil):
    pass
```

F. 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.

Contoh Program :

```
class A:
    def method(self):
        print("A.method() dipanggil")

class B:
    def method(self):
        print("B.method() dipanggil")

class C(A, B):
    pass

class D(B, A):
    pass

c = C()
c.method()

d=D()
d.method()
```

```
A.method() dipanggil
B.method() dipanggil
```

G. 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.

Contoh program :

```
angka_int = 123
angka_float = 1.123

angka_baru = angka_int + angka_float

print("datatype dari angka_int :", type(angka_int))
print("datatype dari angka_float :", type(angka_float))

print("Value dari angka_baru :", angka_baru)
print("datatype dari angka_baru :", type(angka_baru))
```

```
datatype dari angka_int : <class 'int'>
datatype dari angka_float : <class 'float'>
Value dari angka_baru : 124.123
datatype dari angka_baru : <class 'float'>
```

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.

Contoh Program :

```
1  angka_int = 123
2  angka_string = "123"
3
4  print("datatype dari angka_int :", type(angka_int))
5  print("datatype dari angka_float :", type(angka_string))
6
7
8  angka_string = int(angka_string)
9
10 angka_baru = angka_int + angka_string
11
12 print("Value dari angka_baru :", angka_baru)
13 print("datatype dari angka_baru :", type(angka_baru))
```

input

```
datatype dari angka_int : <class 'int'>
datatype dari angka_float : <class 'str'>
Value dari angka_baru : 246
datatype dari angka_baru : <class 'int'>
```

H. Casting

a. Downcasting

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

b. Upcasting

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

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).