

**PEMROGRAMAN BERORIENTASI OBJEK  
RESUME PERTEMUAN 1-4**



Disusun Oleh:  
**Andreyan Renaldi**  
**(121141086)**

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

**2023**

# Pertemuan 1

## 1. Sejarah Bahasa Pemrograman Python

Python adalah bahasa pemrograman tingkat tinggi, dinamis, dan serbaguna yang diciptakan oleh Guido van Rossum pada tahun 1991 di Belanda. Nama "Python" diambil dari judul acara komedi BBC, "Monty Python's Flying Circus".

Tujuan utama dari penciptaan Python adalah untuk menciptakan bahasa pemrograman yang mudah dipelajari, mudah dibaca, dan mudah dipahami, serta memiliki sintaks yang bersih dan sederhana.

## 2. Dasar Pemrograman Python

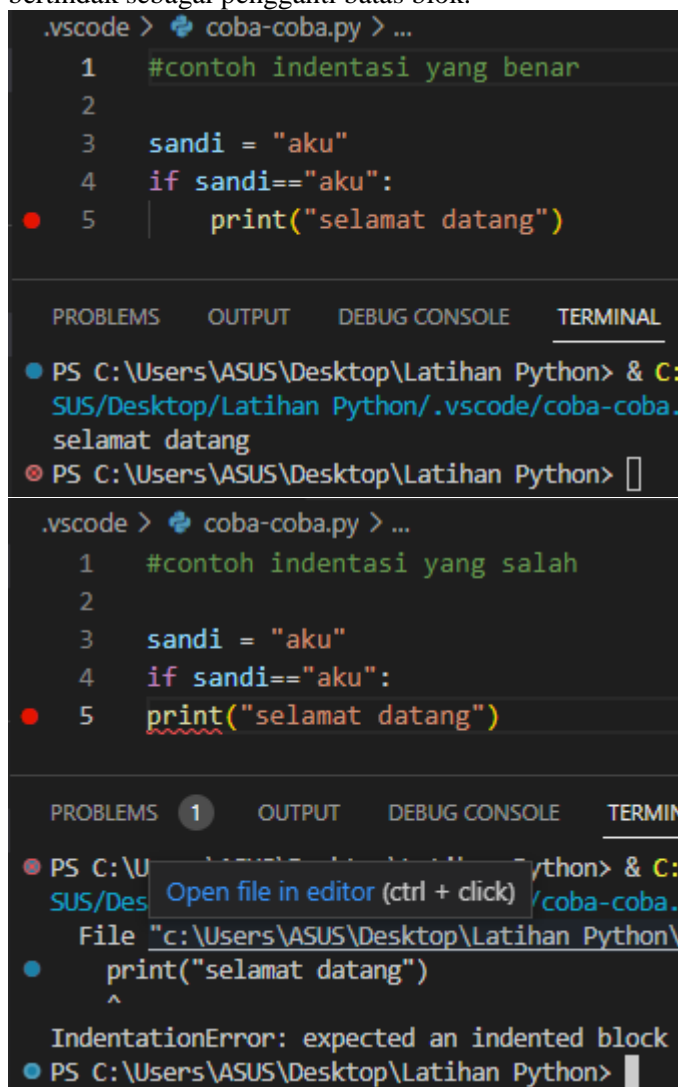
Python memiliki beberapa sintaks dasar, seperti :

### a. Statement

Semua perintah yang mampu dieksekusi Python.

### b. Baris dan Indentasi

Semua aturan indentasi seperti pemakaian spasi yang menggantikan kurung kurawal seperti di bahasa pemrograman lain, selain itu mengatur indentasi juga diharuskan di Python karena bertindak sebagai pengganti batas blok.



```
.vscode > 📄 coba-coba.py > ...
1  #contoh indentasi yang benar
2
3  sandi = "aku"
4  if sandi=="aku":
5      print("selamat datang")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python\.vscode\coba-coba.py
selamat datang
⊙ PS C:\Users\ASUS\Desktop\Latihan Python>

.vscode > 📄 coba-coba.py > ...
1  #contoh indentasi yang salah
2
3  sandi = "aku"
4  if sandi=="aku":
5  print("selamat datang")

PROBLEMS  1  OUTPUT  DEBUG CONSOLE  TERMINAL
⊙ PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python\.vscode\coba-coba.py
File "c:\Users\ASUS\Desktop\Latihan Python\.vscode\coba-coba.py", line 5, in <module>
    print("selamat datang")
    ^
IndentationError: expected an indented block
⊙ PS C:\Users\ASUS\Desktop\Latihan Python>
```

### c. Variabel dan Tipe Data Primitif

Variabel merupakan penyimpan suatu nilai, dan tipe data merupakan pasangan dari variabel untuk menentukan jenis data yang ada. Ada 4 tipe data primitif yaitu *bool* (bernilai true/false), *int* (bernilai bilangan bulat), *float* (bernilai bilangan real), dan *string* (bernilai karakter). Namun, dalam python pasangan variabel dan tipe data tidak harus dituliskan secara eksplisit karena python mendukung *dynamic typing*.

```
.vscode > 📄 coba-coba.py > ...
1  #membuat variabel dengan menulis tipe datanya
2  int1 = int(1)
3  string1 = str("1")
4  float1 = float(1.1)
5  bool1 = bool(True)
6
7  #membuat variabel tanpa menulis tipe datanya
8  int2 = 1
9  string2 = "1"
10 float2 = 1.1
11 bool2 = True
```

d. Operator

Python memiliki beberapa operator yaitu:

1) Operator Aritmatika

⇒ digunakan untuk melakukan operasi aritmatika (pembagian, penambahan, pengurangan, perkalian, dan lainnya). Contoh operator aritmatika yaitu, + (untuk menambah), - (untuk mengurangi), \* ( untuk mengalikan), / (untuk membagikan), dan lainnya.

```
.vscode > 📄 coba-coba.py > ...
1  #operator aritmatika
2
3  a = 2
4  b = 3
5
6  print("hasil a + b = ", a+b)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:
SUS/Desktop/Latihan Python/.vscode/coba-coba.
hasil a + b = 5
● PS C:\Users\ASUS\Desktop\Latihan Python> |
```

2) Operator Perbandingan

⇒ Digunakan untuk membandingkan 2 buah nilai yang akan menghasilkan kondisi True(jika kondisi benar) atau False (jika kondisi salah). Operator yang ada di perbandingan yaitu, > (perbandingan lebih besar), < (perbandingan lebih kecil), == (perbandingan sama dengan), != (perbandingan tidak sama dengan), >= (perbandingan lebih besar atau sama dengan), <= (perbandingan lebih kecil atau sama dengan).

```
.vscode > 🐞 coba-coba.py > ...
1  #operator perbandingan
2
3  a = 2
4  b = 3
5
6  print(a<b)
7  print(a>b)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python\.vscode\coba-coba.py
True
● PS C:\Users\ASUS\Desktop\Latihan Python> 
```

### 3) Operator Penugasan

⇒ digunakan untuk memberi sebuah nilai ke variabel dengan mengoperasikan variabel lain. Contoh operator yang ada di penugasan yaitu = (untuk menugaskan nilai yang ada di sebelah kanan operator agar ke sebelah kiri operator), += (menugaskan untuk menambahkan nilai yang ada di sebelah kiri operator dengan nilai di sebelah kanan operator), -= (menugaskan untuk mengurangi nilai yang ada di sebelah kiri operator dengan nilai di sebelah kanan operator), %= (menugaskan untuk melakukan sisa bagi yang berada di sebelah kanan operator (hasil bagi dari sebelah kiri operator) agar disimpan di variabel sebelah kiri operator), \*= (menugaskan untuk mengalikan nilai yang ada di sebelah kiri operator dengan nilai di sebelah kanan operator untuk disimpan di variabel sebelah kiri).

```
.vscode > 🐞 coba-coba.py > ...
1  #operator penugasan
2
3  a = 2
4  b = 3
5
6  c = a+b
7  print("Nilai dari c adalah ", c)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python\.vscode\coba-coba.py
Nilai dari c adalah 5
● PS C:\Users\ASUS\Desktop\Latihan Python> 
```

### 4) Operator Logika

⇒ Digunakan untuk melakukan operasi logika. Operasi yang ada yaitu and (kondisi true jika kedua variabelnya bernilai benar), or (kondisi true jika salah satu variabelnya bernilai benar), dan not (kondisi true jika operand nya bernilai salah).

```
coba-coba.py X 121140186_Andreyan Renald

.vscode > coba-coba.py > ...
1  #operator logika
2
3  a = 2
4  b = 3
5
6  opsi1 = a > b and a < 5
7  opsi2 = a < b and a < 5
8
9  print("Opsi 1 bernilai ", opsi1)
10 print("Opsi 2 bernilai ", opsi2)
11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python/.vscode/coba-coba.py
Opsi 1 bernilai False
● Opsi 2 bernilai True
● PS C:\Users\ASUS\Desktop\Latihan Python> 
```

##### 5) Operator Identitas

⇒ digunakan untuk memeriksa 2 nilai/variabel berada di satu memori yang sama. Operasi yang ada yaitu, is(bernilai true jika kedua nilai dari variabel identik(sama), dan is not(bernilai true jika kedua nilai tidak identik).

```
.vscode > coba-coba.py > ...
1  #operator identitas
2
3  a = "Aku"
4  b = "Kamu"
5  c = "Aku"
6
7  print(a is b)
8  print(a is c)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python/.vscode/coba-coba.py
False
● True
● PS C:\Users\ASUS\Desktop\Latihan Python> 
```

6) Operator Keanggotaan

⇒ digunakan untuk memeriksa nilai atau variabel terdapat di salah satu anggota data(list,set,tuple,string,dict). Operasi yang ada yaitu in( bernilai true jika ditemukan di dalam data), dan not in(bernilai true, jika nilai tidak ada di dalam data).

```
.vscode > 🚀 coba-coba.py > ...
1  #operator identitas
2
3  List = [1,2,3,4,5,6,7,8,9]
4
5  print(1 in List)
6  print(10 in List)
7
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python/.vscode/coba-coba.py
True
● False
● PS C:\Users\ASUS\Desktop\Latihan Python> 
```

e. Tipe Data Bentukan

Ada 4 Tipe data bentukan yaitu, List (kumpulan data yang dapat diubah dan memungkinkan ada anggota yang sama), Tuple (kumpulan data yang tidak dapat diubah dan memungkinkan ada anggota yang sama), Set (kumpulan data yang tidak terindeks dan tidak ada anggota yang sama), dan Dictionary (kumpulan data yang mendefinisikan data lain yang tidak terindeks dan tidak memungkinkan ada yang sama).

```
.vscode > 🚀 coba-coba.py > ...
1  #Tipe Data Bentukan
2
3  List = [1,2,3,4,5]
4  Tuple = (1,2,3,4,5)
5  Dictionary = {"Aku" : "Andre", "Kamu" : "Renal"}
6  Set = {"Aku", "Kamu", "Dia"}
```

f. Kondisi Percabangan

Python memiliki 4 tipe percabangan yaitu:

1) Percabangan IF

```
.vscode > 🐍 coba-coba.py > ...
1  #percabangan IF
2
3  sandi = "Aku"
4
5  if(sandi == "Aku"):
6      print("Berhasil Login!")
7
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python\.vscode\coba-coba.py
Berhasil Login!
● PS C:\Users\ASUS\Desktop\Latihan Python> 
```

2) Percabangan IF-ELSE

```
.vscode > 🐍 coba-coba.py > ...
1  #percabangan IF-ELSE
2
3  sandi = "Kamu"
4
5  if(sandi == "Aku"):
6      print("Berhasil Login!")
7  else:
8      print("Gagal Login!")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Users\ASUS\Desktop\Latihan Python\.vscode\coba-coba.py
Gagal Login!
● PS C:\Users\ASUS\Desktop\Latihan Python> 
```

3) Percabangan IF-ELIF

```
.vscode > 🐍 coba-coba.py > ...
1  #percabangan IF-ELIF
2
3  bilangan = 0
4
5  if(bilangan > 0):
6      print("Bilangan Positif")
7  elif(bilangan < 0):
8      print("Bilangan Negatif")
9  else:
10     print("Bilangan Nol")
...
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
PS C:\Users\ASUS\Desktop\Latihan Python> & C:
SUS/Desktop/Latihan Python/.vscode/coba-coba.
Bilangan Nol
PS C:\Users\ASUS\Desktop\Latihan Python> 
```

4) Nested IF

```
.vscode > 🐍 coba-coba.py > ...
1  #percabangan Nested IF
2
3  bilangan = 2
4
5  if(bilangan != 0):
6      if(bilangan>0):
7          print("Bilangan Positif")
8      else:
9          print("Bilangan Negatif")
10 else:
11     print("Bilangan Nol")
...
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
PS C:\Users\ASUS\Desktop\Latihan Python> & C:
SUS/Desktop/Latihan Python/.vscode/coba-coba.
Bilangan Positif
PS C:\Users\ASUS\Desktop\Latihan Python> 
```



g. Kondisi Perulangan

Python memiliki 2 perulangan yaitu:

1) Perulangan For

```
.vscode > 📄 coba-coba.py > ...
1  #perulangan For
2
3  for i in range(1, 6, 1):
4      print("Hello!")
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:
SUS/Desktop/Latihan Python/.vscode/coba-coba.
Hello!
● Hello!
● Hello!
Hello!
● Hello!
○ PS C:\Users\ASUS\Desktop\Latihan Python> |
```

2) Perulangan While

```
.vscode > 📄 coba-coba.py > ...
1  #perulangan while
2
3  i = 1
4  while(i<6):
5      print("Hello!")
6      i = i+1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:
SUS/Desktop/Latihan Python/.vscode/coba-coba.
Hello!
● Hello!
● Hello!
Hello!
● Hello!
● PS C:\Users\ASUS\Desktop\Latihan Python> |
```

h. Fungsi Python

Python bisa mendefinisikan sebuah fungsi(def) yang berguna menyimpan sejumlah blok kode agar bisa dieksekusi berulang tanpa menulis ulang kode tersebut.

```
.vscode > coba-coba.py > faktorial
1  #fungsi python
2
3  def faktorial(a):
4      sum = 1
5      b = a
6      while(a>0):
7          sum = sum*a
8          a = a-1
9      print("faktorial dari ", b, " adalah ", sum)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

- PS C:\Users\ASUS\Desktop\Latihan Python> & C:/Users/ASUS/AppD...
- faktorial dari 4 adalah 24
- PS C:\Users\ASUS\Desktop\Latihan Python> █

## Pertemuan 2

### 1. Kelas

Class pada Python adalah sebuah cetak biru atau blueprint yang mendefinisikan karakteristik dan perilaku dari sebuah objek. Di dalam class, terdapat atribut yang merepresentasikan data dan method yang merepresentasikan perilaku dari objek tersebut.

```
.vscode > coba-coba.py > Manusia > __init__
1  #fungsi python
2
3  class Manusia:
4      def __init__(self, nama, usia):
5          self.nama = nama
6          self.usia = usia
7
8      def info(self):
9          print(f"{self.nama} berusia {self.usia} tahun")
10
11  human1 = Manusia("Andreyan", "18")
12  human1.info()
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
● PS C:\Users\ASUS\Desktop\Latihan Python> & C:/Users/ASUS/AppData/Local/
SUS/Desktop/Latihan Python/.vscode/coba-coba.py"
● Andreyan berusia 18 tahun
● PS C:\Users\ASUS\Desktop\Latihan Python> |
```

### 2. Objek

Objek merupakan sebuah pengganti dari pemanggilan kelas. Fungsi dari objek adalah mempersingkat pemanggilan kelas(pengganti). Contoh penulisan objek adalah sebagai berikut.

```
11  human1 = Manusia("Andreyan", "18")
12  human1.info()
```

### 3. Magic Method

Magic method merupakan sebuah metode python yang merubah suatu sifat bawaan objek, ada banyak magic method yang terdapat di python yang seluruhnya dapat diakses dari *dir(int)*. Contoh penerapan pemakaian magic method adalah sebagai berikut.

```
.vscode > 📄 coba-coba.py > ...
1  #Magic Method
2
3  class RS:
4      def __init__(self):
5          self.list_pasien = ["Andre", "Rey", "Renal", "Aldi"]
6
7      def info(self):
8          print(f"Daftar nama pasien : {self.list_pasien}")
9
10     def __len__(self):
11         return len(self.list_pasien)
12
13     ps = RS()
14     ps.info()
15     print(f"Banyak pasien : {len(ps)}")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

- PS C:\Users\ASUS\Desktop\Latihan Python> & C:/Users/ASUS/AppData/Local/Programs...
- a.py"
- Daftar nama pasien : ['Andre', 'Rey', 'Renal', 'Aldi']
- Banyak pasien : 4
- PS C:\Users\ASUS\Desktop\Latihan Python> |

Penggunaan magic method yang digunakan adalah magic method `__len__` yang berguna untuk menentukan panjang dari sebuah list.

#### 4. Konstruktor dan Destruktor

Konstruktor merupakan method pasti yang akan dijalankan saat sebuah objek memanggil kelas. Konstruktor biasa dipanggil dengan magic method (`__init__`). Destruktor merupakan method yang dipanggil otomatis oleh sistem ketika pengguna memakai magic method (`__del__`). Contoh penerapan konstruktor dan destruktur terdapat di bawah ini.

```

.vscode > 🐼 coba-coba.py > ...
1  #konstruktor dan destruktur
2
3  class Manusia:
4      def __init__(self, nama, usia):
5          self.nama = nama
6          self.usia = usia
7
8      def info(self):
9          print(f"{self.nama} berusia {self.usia} tahun")
10
11     def __del__(self):
12         self.usia = "tidak diketahui"
13         print(f"nama {self.nama} usia {self.usia}")
14
15
16  human1 = Manusia("Andreyan", "18")
17  human1.info()

```

```

.vscode > 🐼 coba-coba.py > ...
1  #konstruktor dan destruktur
2
3  class Manusia:
4      def __init__(self, nama, usia):
5          self.nama = nama
6          self.usia = usia
7
8      def info(self):
9          print(f"{self.nama} berusia {self.usia} tahun")
10
11     def __del__(self):
12         self.usia = "tidak diketahui"
13         print(f"nama {self.nama} usia {self.usia}")
14
15
16  human1 = Manusia("Andreyan", "18")
17  human1.info()

```

Destruktor akan otomatis terpanggil tanpa dipanggil oleh pengguna.

## 5. Setter dan Getter

Setter dan Getter berfungsi sebagai enkapsulasi agar data tidak dapat diubah secara sembarangan. Setter merupakan method untuk menetapkan suatu nilai, dan getter dilakukan untuk mengambil suatu nilai. Contoh penerapannya adalah sebagai berikut.

```

class Mahasiswa:
    #atribut global
    prodi = "Teknik Informatika"

    def __init__(self, nama, nim, umur, alamat):
        #atribut publik
        self.nama = nama
        self.nim = nim

        #atribut protected
        self._umur = umur

        #atribut private
        self.__alamat = alamat

    def get_umur(self):
        return self._umur

    #fungsi protected
    def set_umur(self, x):
        self._umur = x

```

from tugas 3

## 6. Decorator

Decorator merupakan sebuah property yang membuat fungsi baru dengan membuat 1 buah nama diatas setter dan getter. Contoh penerapannya adalah sebagai berikut.

```

16     @umur.getter
17     def get_umur(self):
18         return self._umur
19
20     @umur.setter
21     #fungsi protected
22     def set_umur(self, x):
23         self._umur = x

```

## Pertemuan 3

### 1. Abstraksi

Abstraksi merupakan sebuah konsep dimana menyembunyikan detail yang user sekiranya tidak perlu tahu namun masih memperlihatkan atribut yang esensial. User mengerti apa yang dilakukan objek namun mekanisme yang terjadi didalam objek disembunyikan.

### 2. Enkasulapsi

Enkapsulasi merupakan sebuah konsep dimana mengatur struktur dari kelas dengan cara menyembunyikan alur dari kelas. Salah satu caranya yaitu membuat property tertentu yang dapat diakses dari luar kelas. Terdapat 3 access modifier untuk mengatur hak akses terhadap property yaitu Public Access Modifier, Protected Access Modifier, dan Private access Modifier.

Pembeda dari ketiga hal tersebut adalah sebagai berikut

- Penulisan syntax Public tidak memakai atribut tambahan apapun, jika di protected pengguna harus menambahkan underscore(\_) sebelum nama method, jika di private pengguna harus menambahkan doubleunderscore(\_\_) sebelum nama method.
- Public dapat diakses dari mana saja, Protected hanya dapat diakses oleh kelas turunan darinya, dan Private hanya diakses di dalam kelas itu sendiri.

Contoh pemakaian ketiga hal tersebut akan dituliskan dibawah ini.

```
# penerapan 3 jenis modifier
class Nasabah:
    def __init__(self, nama, umur, norek, tabungan):
        self.nama = nama # atribut public
        self.umur = umur # atribut public
        self.__norek = norek # atribut private
        self._tabungan = tabungan # atribut protected

    #contoh pemakaian private attribute
    def get_norek(self):
        return self.__norek

Nasabah1= Nasabah("Tok Dalang", 13, 12890003, 100000)
# cara akses private attribute
print(Nasabah1.get_norek())
# mencoba akses private attribute dari luar kelas(akan terjadi error)
print(Nasabah1.__norek)
```

Jika memaksakan mengakses private attribute akan mengeluarkan error sebagai berikut.

```
print(Nasabah1.__norek)
AttributeError: 'Nasabah' object has no attribute '__norek'
```

Hasil jika mengakses fungsi get (method yang berisikan return dari private attribute).

```
12890003
```

Setter dan Getter sangat berperan dalam pengaksesan private dan protected attribute baik dengan decorator atau tanpa decorator, contoh `get_norek` diatas merupakan implementasi dari getter tanpa decorator (lebih lengkap di pertemuan 2).

### 3. Objek di Python

Untuk mengakses sebuah kelas di python dapat menggunakan objek. Contoh deklarasi dari objek yaitu sebagai berikut.

```
16 human1 = Manusia("Andreyan", "18")
```

Nasabah1 sebagai nama objek, nasabah sebagai nama kelas yang akan dipanggil dan atribut yang ada di dalam kurung merupakan permintaan dari kelas `__init__` yang ada di kelas. Cara mengakses atribut dari kelas dapat menggunakan sintaks sebagai berikut.

```
human1.info()
```

Hasilnya sebagai berikut.

```
PS C:\Users\ASUS\Desktop\Latihan Python>
a.py"
Andreyan berusia 18 tahun
```



## Pertemuan 4

### 1. Pewarisan (Inheritance)

Inheritance merupakan sebuah konsep yang menuju kepada penurunan kelas. Terdapat hierarchy dari kelas kelas. Kelas turunan dari kelas sebelumnya bisa memakai segala atribut dan metode dari kelas yang menurunkan. Kelas yang menurunkan disebut Parent, dan kelas yang diturunkan disebut Child. Contoh penerapan inheritance adalah sebagai berikut:

```
1 class Komputer:
2     def __init__(self, nama, jenis, harga, merk):
3         self.nama = nama
4         self.jenis = jenis
5         self.harga = harga
6         self.merk = merk
7
8 class Processor(Komputer):
9     def __init__(self, nama, jenis, harga, merk, jumlah_core, kecepatan_processor):
10        super().__init__(nama, jenis, harga, merk)
11        self.jumlah_core = jumlah_core
12        self.kecepatan_processor = kecepatan_processor
13
14    def tampil(self):
15        print(f"Processor {self.jenis} produksi {self.nama}")
```

Macam macam inheritance yaitu Inheritance Identik dan Inheritance Multiple, akan dijelaskan dibawah ini.

#### a. Inheritance Identik

Inheritance identik merupakan inheritance yang menambahkan sebuah konstruktor di child class. Metode ini memiliki kata kunci **super** di class child nya. Namun child class masih bisa ditambahkan sejumlah atribut tambahan (jika diperlukan). Namun identiknya akan hilang.

```
1 class Komputer:
2     def __init__(self, nama, jenis, harga, merk):
3         self.nama = nama
4         self.jenis = jenis
5         self.harga = harga
6         self.merk = merk
7
8 class Processor(Komputer):
9     def __init__(self, nama, jenis, harga, merk, jumlah_core, kecepatan_processor):
10        super().__init__(nama, jenis, harga, merk)
11        self.jumlah_core = jumlah_core
12        self.kecepatan_processor = kecepatan_processor
13
14    def tampil(self):
15        print(f"Processor {self.jenis} produksi {self.nama}")
```

#### b. Inheritance Multiple

Python memungkinkan kelas parent menurunkan banyak kelas dan menurunkan kelas turunan ke kelas turunan lain(Bertingkat). Contoh dari Penurunan ke banyak kelas adalah sebagai berikut:

```
1  class Komputer:
2      def __init__(self, nama, jenis, harga, merk):
3          self.nama = nama
4          self.jenis = jenis
5          self.harga = harga
6          self.merk = merk
7
8  class Processor(Komputer):
9      def __init__(self, nama, jenis, harga, merk, jumlah_core, kecepatan_processor):
10         super().__init__(nama, jenis, harga, merk)
11         self.jumlah_core = jumlah_core
12         self.kecepatan_processor = kecepatan_processor
13
14     def tampil(self):
15         print(f"Processor {self.jenis} produksi {self.nama}")
16
17 class RAM(Komputer):
18     def __init__(self, nama, jenis, harga, merk, kapasitas_ram):
19         super().__init__(nama, jenis, harga, merk)
20         self.kapasitas_ram = kapasitas_ram
21
22     def tampil(self):
23         print(f"RAM {self.jenis} produksi {self.nama}")
24
25 class HDD(Komputer):
26     def __init__(self, nama, jenis, harga, merk, kapasitas_hdd, rpm):
27         super().__init__(nama, jenis, harga, merk)
28         self.kapasitas_hdd = kapasitas_hdd
29         self.rpm = rpm
30
31     def tampil(self):
32         print(f"SATA HDD {self.jenis} produksi {self.nama}")
```

## 2. Polymorphism

Polymorphism adalah konsep memerintah objek yang secara prinsip sama, namun secara proses berbeda. Python tidak menangani hal ini karena python bersifat duck typing.

### 3. Overriding

Overriding merupakan ambil alih/menimpa metode yang ada di parent class dengan mendefinisikan metode yang sama di child class. Dengan itu maka metode yang ada di parent sudah digantikan oleh metode di child class. Contoh penerapannya adalah sebagai berikut.

```
.vscode > coba-coba.py > ...
1  #overriding
2
3  class Hero:
4      def cetak(self):
5          print("I am hero")
6
7  class Hero2(Hero):
8      def cetak(self):
9          print("I am groot")
10
11  Hr = Hero2()
12  Hr.cetak()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Python39\python.exe C:\Users\ASUS\Desktop\Latihan Python> a.py
I am groot
PS C:\Users\ASUS\Desktop\Latihan Python>
```

### 4. Casting

Casting memiliki beberapa jenis yaitu:

#### a. Downcasting

Downcasting adalah parent class mengakses atribut yang ada di kelas childnya untuk dibawa kemudian ke dalam class parent dan dipakai. Contoh penerapannya adalah sebagai berikut.

```
.vscode > coba-coba.py > ...
7  class Hero2(Hero):
8      def __init__(self, nama):
9          super().__init__()
10         self.nama = nama
11
12         def cetak(self):
13             print("I am groot")
14
15  Hr = Hero2("Super Dede")
16  Hr.tampil()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Python39\python.exe C:\Users\ASUS\Desktop\Latihan Python> a.py
I am Super Dede
PS C:\Users\ASUS\Desktop\Latihan Python>
```

b. Upcasting

Upcasting yaitu kelas child mengakses atribut yang ada di kelas parent penerapannya yaitu sebagai berikut :

```
.vscode > 📄 coba-coba.py > ...
1  #overriding
2
3  class Hero:
4      nama = "Super Dede"
5      def __init__(self, nama):
6          self.nama = nama
7
8  class Hero2(Hero):
9      def __init__(self, nama, usia):
10         super().__init__(nama)
11         self.usia = usia
12
13         def cetak(self):
14             print(f"I am {super().nama}")
15
16  Hr = Hero2("Udin", 13)
17  Hr.cetak()
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
PS C:\Users\ASUS\Desktop\Latihan Python> & C:\Programs/Python/Python311-32/python.exe "c:/Users/ASUS/Desktop/Latihan Python/.vscode/coba-coba.py"
I am Super Dede
PS C:\Users\ASUS\Desktop\Latihan Python> |
```

c. Typecasting

Typecasting yaitu konversi tipe kelas agar berperilaku selayaknya default tidak dimiliki oleh kelas tersebut. Contoh penggunaannya yaitu sebagai berikut: