

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



Oleh :

Dhio Eko Permana 121140086

**PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA**

2023

DAFTAR ISI

Pertemuan 1	3
1. Sejarah Bahasa Pemrograman Python	3
2. Dasar Pemrograman Python.....	3
Pertemuan 2	13
1. Kelas	13
2. Objek.....	14
3. Magic Method.....	14
4. Konstruktor dan Desktruktor	15
5. Setter and Getter.....	16
6. Decorator.....	16
Pertemuan 3	18
1. Abstraksi	18
2. Enkapsulasi	18
3. Objek di Python	19
Pertemuan 4	20
1. Pewarisan (Inheritance).....	20
2. Polymorphism	22
3. Overriding	22
4. Casting	23

Pertemuan 1

1. Sejarah Bahasa Pemrograman Python

Bahasa pemrograman Python dibuat oleh Guido Van Rossum pada tahun 1980 di Centrum Wiskunde & Informatica, Belanda. Python mendukung paradigma pemrograman seperti Object-Oriented, Functional, dan Structured. Bahasa Python populer karena sintaks yang tergolong mudah dan modul yang banyak.

Python sangat mementingkan *readability* pada kode, itulah sebabnya python tidak memakai kurung kurawal, sebagai gantinya menggunakan spasi.

2. Dasar Pemrograman Python

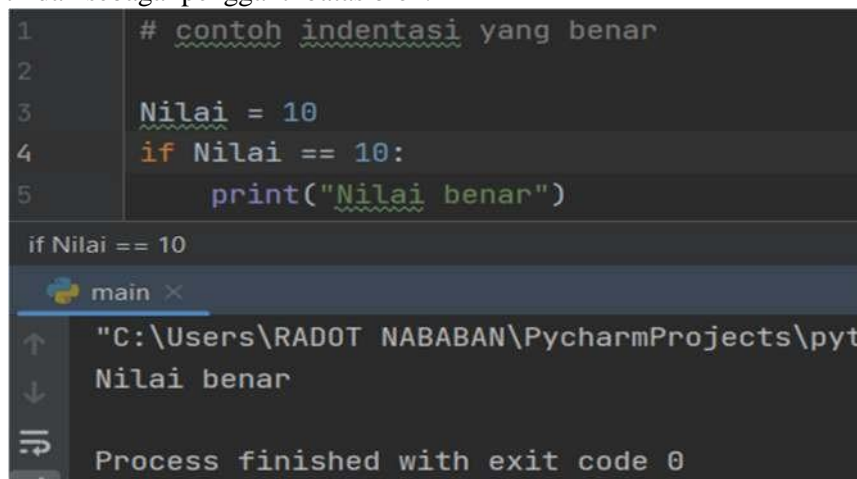
Python memiliki beberapa sintaks dasar, seperti :

1. Statement

Semua perintah yang mampu dieksekusi Python

2. 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.



```
1 # contoh indentasi yang benar
2
3 Nilai = 10
4 if Nilai == 10:
5     print("Nilai benar")
```

if Nilai == 10

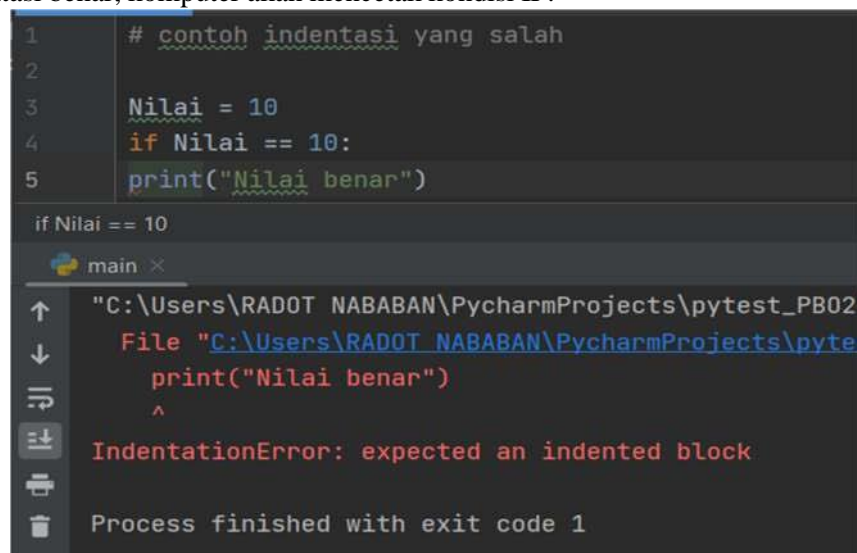
main x

"C:\Users\RADOT NABABAN\PycharmProjects\pyt

Nilai benar

Process finished with exit code 0

Jika indentasi benar, komputer akan mencetak kondisi IF.



```
1 # contoh indentasi yang salah
2
3 Nilai = 10
4 if Nilai == 10:
5     print("Nilai benar")
```

if Nilai == 10

main x

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02

File "C:\Users\RADOT NABABAN\PycharmProjects\pyte

print("Nilai benar")

^

IndentationError: expected an indented block

Process finished with exit code 1

Jika indentasi salah, compiler akan mengirimkan error **IndentationError**

3. 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*.

```
1 # membuat variabel dengan menuliskan tipe data
2 angka1 = int(10) # integer
3 string1 = str("10") # string
4 CekBenar1 = bool(True) # boolean
5 float1 = float(3.1) # float
6
7 # membuat variabel tanpa menuliskan tipe data
8 angka2 = 10 # integer
9 string2 = "10" # string
10 CekBenar2 = True # boolean
11 float2 = float(3.1) # float
12
13
```

4. Operator

Python memiliki beberapa operator yaitu : .

- a. 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.

```
1 # contoh pemakaian operator aritmatika
2 a = 0
3 b = 7
4 print("Pertandingan Man United vs Liverpool berakhir dengan skor", a + b, "-", a)
5
```

Run: main x

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\main.py"

Pertandingan Man United vs Liverpool berakhir dengan skor 7 - 0

Process finished with exit code 0

- b. 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)

```
1 # contoh pemakaian operator perbandingan
2 a = 10
3 b = 7
4 print(a > b) # mengeluarkan true karena kondisi benar
5 print(a < b) # mengeluarkan false karena kondisi salah
6
```

Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe"

True

False

- c. 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).

```
1 # contoh pemakaian operator penugasan
2 a = 10
3 b = 7
4 # operator (=)
5 c = a+b
6 print("Hasil dari c di operator = adalah : ")
7 print(c)
8 # operator (+=)
9 c += b
10 print("Hasil dari c di operator += adalah : ")
11 print(c)
12 # operator (*=)
13 c -= a
14 print("Hasil dari c di operator -= adalah : ")
15 print(c)
```

Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Sc
Hasil dari c di operator = adalah :
17
Hasil dari c di operator += adalah :
24
Hasil dari c di operator -= adalah :
14

Process finished with exit code 0

- d. 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).

```
1 #contoh pemakaian operator logika
2 a = 10
3 b = 19
4
5 operator1 = a > 15 and a < b
6 operator2 = a > 15 or a < b
7
8 print("Kondisi operator 1 adalah ", operator1)
9 print("Kondisi operator 2 adalah ", operator2)
10
```

Run: main ×

```
"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\ve
Kondisi operator 1 adalah False
Kondisi operator 2 adalah True

Process finished with exit code 0
```

- e. 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)

```
main.py × lat3pbopy.py × PYTugas4PrakPBO.py × PYTuga
1 #contoh pemakaian operator identitas
2 a = "Barcelona FC"
3 b = "PSMS Medan"
4 c = "Barcelona FC"
5
6 print(a is c)
7 print(a is not b)
8
9
10
```

Run: main ×

```
"C:\Users\RADOT NABABAN\PycharmProjects\pytest.
True
True

Process finished with exit code 0
```

- f. 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).

```
1  #contoh pemakaian operator keanggotaan
2  ini_List = [1,3,5,6,7,8,9,10]
3  print(1 in ini_List)
4  print(99 not in ini_List)
5  print("----")
6  #contoh lain (string)
7  ini_string = "Leo Messi is GOAT"
8  print("GOAT" in ini_string)
9
10
```

Run: main x

"C:\Users\RADOT NABABAN\PycharmProjects\py
True
True

True

5. Tipe Data Bentukan

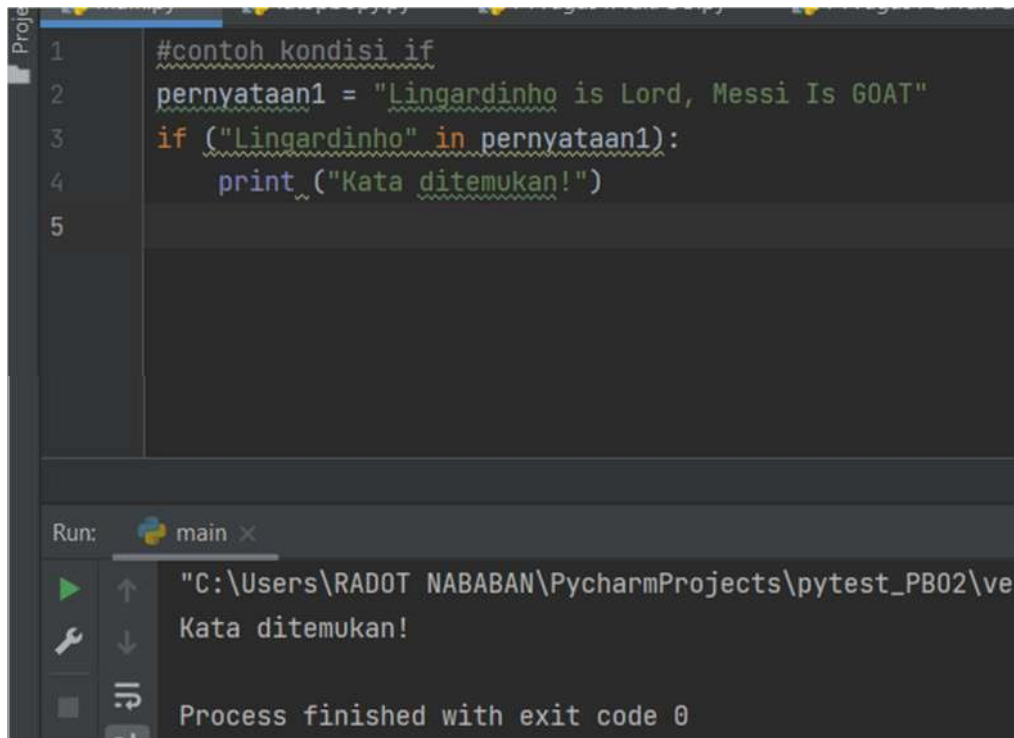
Ada 4 Tipe data bentukan yaitu, **List** (kumpulan data yang dapat diubah dan memungkinkan ada anggota yang sama), **Tuple** (kumpulan data yang tidak dapat diubah dan memungkinkan ada anggota yang sama), **Set** (kumpulan data yang tidak terindeks dan tidak ada anggota yang sama), dan **Dictionary**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

```
1  #contoh penulisan tipe data bentukan
2  ini_List = [1,3,5,6,7,8,9,10]
3  ini_tuple = (1,3,4,5,6,19,6,1)
4  ini_dictionary = {"Messi": "GOAT", "Lingardinho": "LORD"}
5  ini_set = {"aku", "kamu", "dia"}
6
```

6. Kondisi Percabangan

Python memiliki 4 percabangan yaitu :

- a. Percabangan IF. Ini akan terpenuhi jika kondisi pertama terpenuhi, jika tidak maka tidak akan mengeksekusi apapun



The screenshot shows the PyCharm IDE with a Python file open. The code defines a string `pernyataan1` and checks if `"Lingardinho"` is in it. The output window shows the string was found.

```
1 #contoh kondisi if
2 pernyataan1 = "Lingardinho is Lord, Messi Is GOAT"
3 if ("Lingardinho" in pernyataan1):
4     print("Kata ditemukan!")
5
```

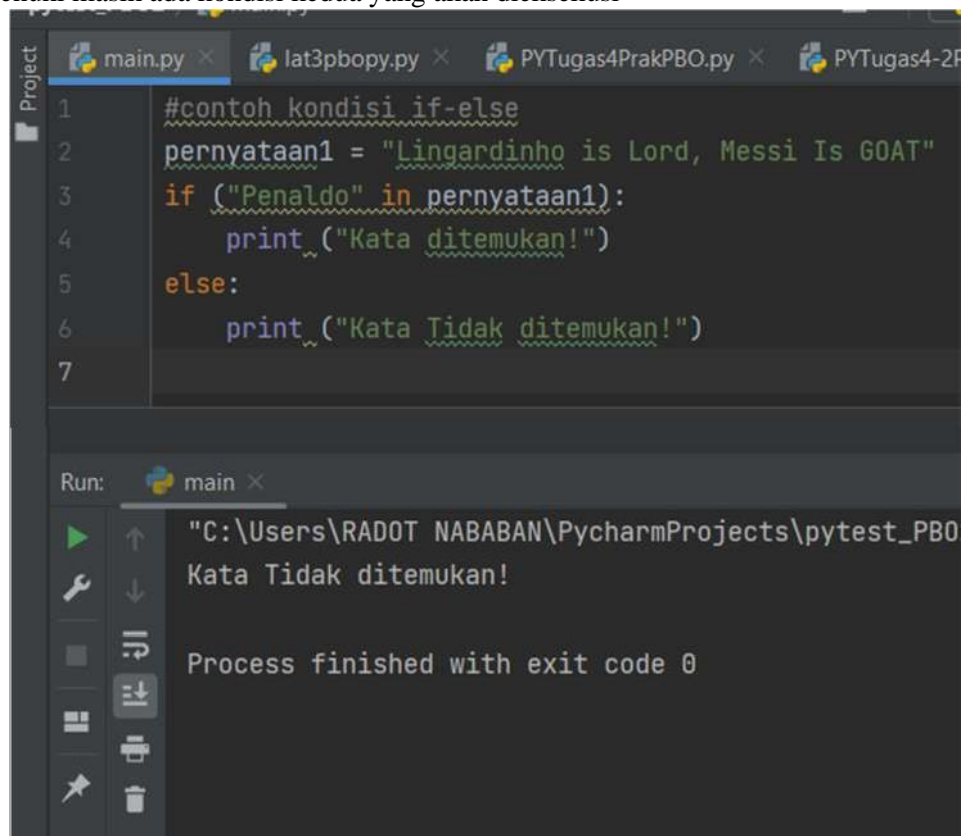
Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\main.py"

Kata ditemukan!

Process finished with exit code 0

- b. Percabangan IF-ELSE. Ini akan terpenuhi jika kondisi pertama terpenuhi, namun jika tidak terpenuhi masih ada kondisi kedua yang akan dieksekusi



The screenshot shows the PyCharm IDE with a Python file open. The code defines a string `pernyataan1` and checks if `"Penaldo"` is in it. Since it's not, the else block executes, printing "Kata Tidak ditemukan!".

```
1 #contoh kondisi if-else
2 pernyataan1 = "Lingardinho is Lord, Messi Is GOAT"
3 if ("Penaldo" in pernyataan1):
4     print("Kata ditemukan!")
5 else:
6     print("Kata Tidak ditemukan!")
7
```

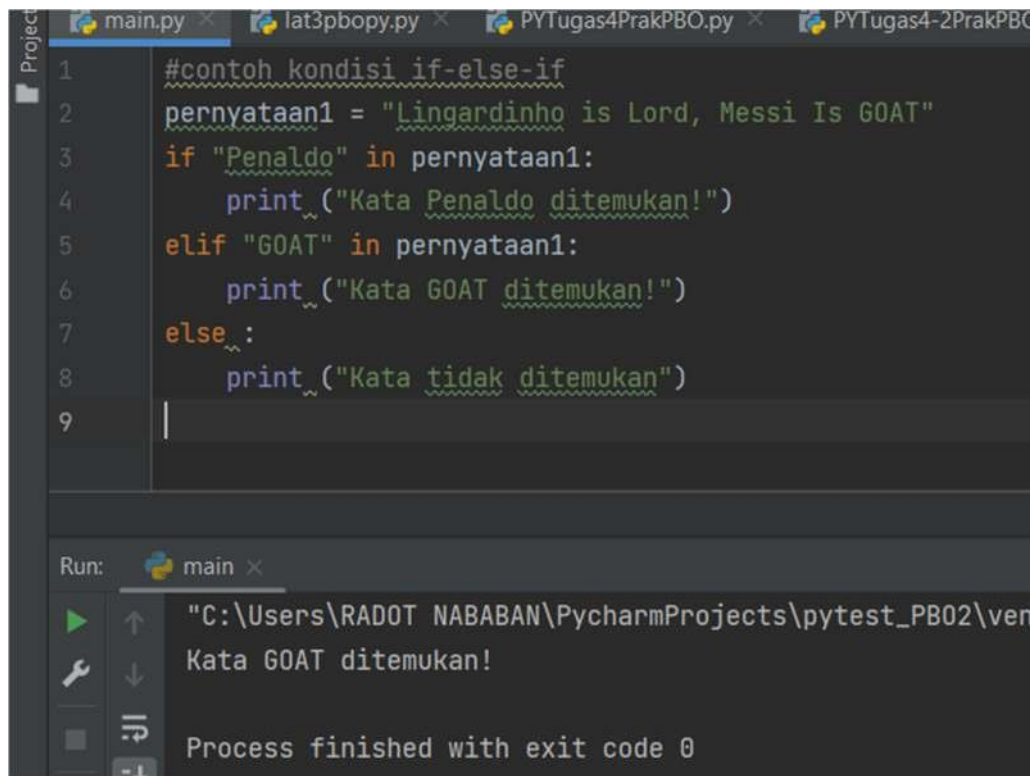
Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\main.py"

Kata Tidak ditemukan!

Process finished with exit code 0

- c. Percabangan IF-ELIF. Ini akan terpenuhi jika kondisi pertama terpenuhi, namun jika tidak terpenuhi masih ada kondisi yang lain bersintaks **elif** yang akan dicek kondisinya, jika belum memenuhi kondisi juga maka kondisi terakhir(else) yang akan dieksekusi.



```
1 #contoh kondisi if-else-if
2 pernyataan1 = "Lingardinho is Lord, Messi Is GOAT"
3 if "Penaldo" in pernyataan1:
4     print("Kata Penaldo ditemukan!")
5 elif "GOAT" in pernyataan1:
6     print("Kata GOAT ditemukan!")
7 else:
8     print("Kata tidak ditemukan")
9
```

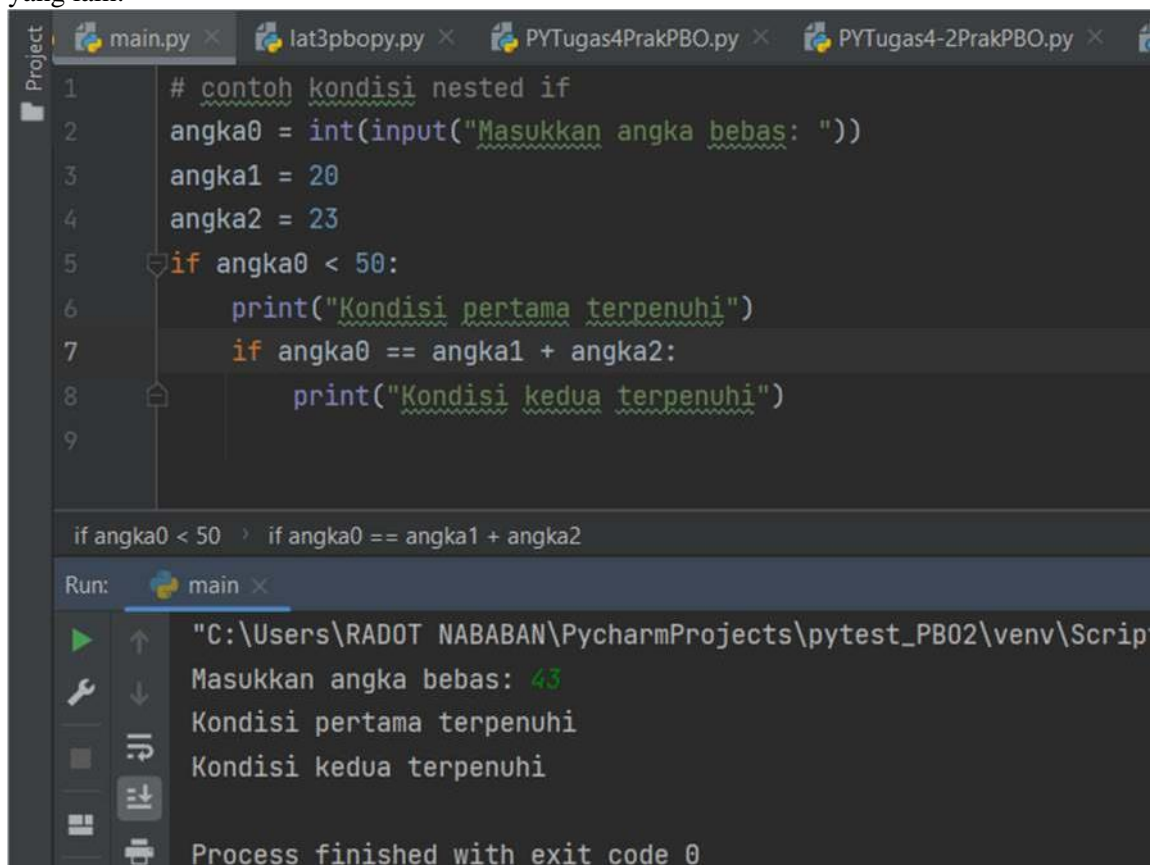
Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\main.py"

Kata GOAT ditemukan!

Process finished with exit code 0

- d. Nested IF(Percabangan bersarang). Bisa membuat sebuah kondisi if didalam kondisi if yang lain.



```
1 # contoh kondisi nested if
2 angka0 = int(input("Masukkan angka bebas: "))
3 angka1 = 20
4 angka2 = 23
5 if angka0 < 50:
6     print("Kondisi pertama terpenuhi")
7     if angka0 == angka1 + angka2:
8         print("Kondisi kedua terpenuhi")
9
```

if angka0 < 50 > if angka0 == angka1 + angka2

Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\main.py"

Masukkan angka bebas: 43

Kondisi pertama terpenuhi

Kondisi kedua terpenuhi

Process finished with exit code 0

7. Kondisi Perulangan.

Python memiliki 2 perulangan yaitu :

- a. Perulangan For. Perulangan for digunakan untuk iterasi yang sudah diketahui batas perulangannya, sintaks umum dari perulangan for contohnya adalah **for i in range (x)** yang berarti pengulangan akan dilakukan dari 0 sampai x, **for i in range (x,y)** yang berarti pengulangan akan dilakukan dari x hingga y, dan **for i in range (x,y,z)** yang berarti pengulangan akan dilakukan dari x hingga y, dengan increment/decrement sejumlah z. Perulangan for juga bisa ditentukan batasnya secara otomatis sebanyak jumlah data yang ada di list, contoh sintaksnya yaitu **for i in (nama_list)**.

```
1 # contoh kondisi perulangan for
2 for i in range(1, 10, 2):
3     print(i, end=" ")
4 print()
5
6 ini_list = ["PSMS", "PSM", "PERSIJA", "PERSIB"]
7 for i in ini_list:
8     print(i, end=" ")
9
```

Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\main.py"

1 3 5 7 9
PSMS PSM PERSIJA PERSIB
Process finished with exit code 0

- b. Perulangan While. Perulangan while digunakan untuk iterasi yang memiliki batas perulangan adalah **jika** suatu kondisi terpenuhi. Contoh sintaksnya yaitu **while(kondisi):**.

```
1 # contoh kondisi perulangan while
2 listKota = ['Melbourne', 'Milan', 'San Diego', 'Doha', 'Kuala Lumpur',
3             'Medan', 'Canberra', 'Mumbai']
4
5 i = 0
6 while i < len(listKota):
7     print(listKota[i])
8     i += 1
9
```

while i < len(listKota)

Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\main.py"

Melbourne
Milan
San Diego
Doha
Kuala Lumpur
Medan
Canberra
Mumbai
Process finished with exit code 0

8. Fungsi

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

```
1  # contoh penggunaan fungsi
2  def pengali(angka):
3      hasil = angka * 3
4      print("Hasil pengalian angka bebas", angka, "dengan angka 3 adalah",
5  def pengalidua(i, j):
6      hasil = i * j
7      print("Hasil Pengalian angka ", i, "dan", j, "adalah ", hasil)
8  i = int(input("Masukkan angka bebas (1) : ")) # memasukkan angka
9  j = int(input("Masukkan angka bebas (2) : ")) # memasukkan angka
10 pengali(i) # memasukkan angka ke dalam fungsi pengali
11 pengali(j) # memasukkan angka ke dalam fungsi pengali
12 pengalidua(i, j)
13
```

Run: main ×

```
"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\env\Scripts\python.
Masukkan angka bebas (1) : 3
Masukkan angka bebas (2) : 4
Hasil pengalian angka bebas 3 dengan angka 3 adalah 9
Hasil pengalian angka bebas 4 dengan angka 3 adalah 12
Hasil Pengalian angka 3 dan 4 adalah 12

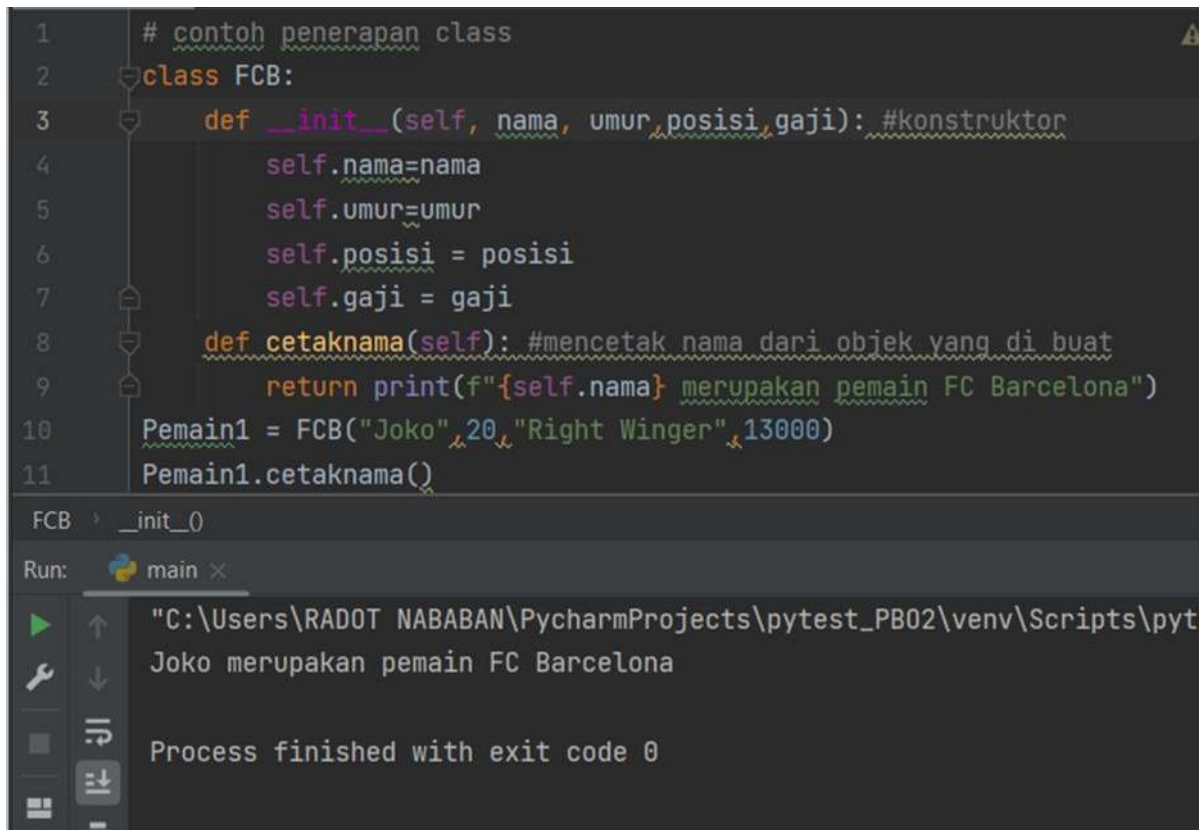
Process finished with exit code 0
```

Pertemuan 2

1. Kelas

Kelas di Python didefinisikan sebagai blueprint dari objek yang akan dibuat. Satu buah kelas dapat menampung banyak objek.

Membuat class dapat dilakukan dengan memakai kata kunci **class** diikuti dengan nama kelas. Contohnya **class Bebas**:. Penerapan secara langsung adalah sebagai berikut.



```
1  # contoh penerapan class
2  class FCB:
3      def __init__(self, nama, umur, posisi, gaji): #konstruktor
4          self.nama=nama
5          self.umur=umur
6          self.posisi = posisi
7          self.gaji = gaji
8      def cetaknama(self): #mencetak nama dari objek yang di buat
9          return print(f"{self.nama} merupakan pemain FC Barcelona")
10  Pemain1 = FCB("Joko", 20, "Right Winger", 13000)
11  Pemain1.cetaknama()
```

FCB > __init__()

Run: main ×

"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\pyt
Joko merupakan pemain FC Barcelona

Process finished with exit code 0

Pada gambar diatas dibuat suatu class **FCB** dengan konstruktor **init**, dan fungsi **cetaknama**. Didefinisikan sebuah variabel **Pemain1** dengan memanggil class **FCB**. Kemudian mencetak nama dengan fungsi **cetaknama**.

Didalam sebuah kelas terdapat variabel(atribut) dan method(fungsi). Atribut merupakan hal yang di deklarasikan di sebuah class. Atribut mempunyai 2 jenis yaitu Atribut Objek dan Atribut Kelas. Perbedaan dari kedua atribut tersebut adalah penempatannya, dimana atribut objek ditaruh didalam objek, sedangkan atribut kelas diletakkan di sebuah kelas(atribut public).Method merupakan fungsi yang terdapat di suatu kelas. Contoh penerapan dari kedua hal tersebut adalah sebagai berikut :


```

class FCB:
    jumlah_pemain = 0 # merupakan atribut kelas

    def __init__(self, nama):
        self.nama = nama # merupakan atribut objek

    def cetaknama(self): # merupakan sebuah method
        return print(f"{self.nama} merupakan pemain FC Barcelona")

```

Pengguna dapat memanggil atribut dan memanggil method. Cara memanggil method dan atribut adalah sebagai berikut:

```

Pemain1.cetaknama() # memanggil method
print(Pemain1.nama) # memanggil atribut

```

```

Marselino merupakan pemain FC Barcelona
Marselino

Process finished with exit code 0

```

Cara memanggil atribut dan objek beserta hasilnya terdapat di gambar diatas.

2. Objek

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

```

Pemain1 = FCB("Joko", 20, "Right Winger", 13000)

```

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 :

```
1 # contoh penerapan magic method
2 class Ruangan:
3     def __init__(self):
4         self.list_siswa=["Fajar", "Anto", "Riyandi", "Muklis"] # menulis list siswa
5     def hapusSiswa(self):
6         self.list_siswa.pop() # menghapus siswa dari urutan paling akhir
7     def len(self):
8         return len(self.list_siswa) # memanggil banyaknya siswa
9
10 admin1 = Ruangan()
11 print(admin1.list_siswa)
12 admin1.hapusSiswa()
13 print(len(admin1))
14 print(admin1.list_siswa)
15
```

Run: lat3pbopy x

```
"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\User
['Fajar', 'Anto', 'Riyandi', 'Muklis']
3
['Fajar', 'Anto', 'Riyandi']
```

Penggunaan magic method yang digunakan adalah magic method `len` yang berguna untuk menentukan panjang dari sebuah list. Pemanggilan sebuah magic method `len` terdapat di kotak merah di gambar diatas. Jika tidak memakai magic method `len`, maka akan keluar error sebagai berikut

```
Traceback (most recent call last):
  File "C:\Users\RADOT NABABAN\PycharmProjects\pytest
    print(len(admin1))
TypeError: object of type 'Ruangan' has no len()
```

4. Konstruktor dan Desktruktor

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 gambar dibawah

```
def __init__(self, nama, umur, posisi, gaji): #merupakan konstruktor
    self.nama = nama #atribut dalam konstruktor
    self.umur = umur
    self.posisi = posisi
    self.gaji = gaji
    self.jumlah_pemain += 1
def __del__(self): #merupakan pemanggil otomatis destruktur
    print("Objek Dihapus!")
```

```
18 Pemain1 = FCB("Marselino", 20, "Right Winger", 13000)
19 print(Pemain1.nama) # memanggil atribut
```

Run: main ×

```
"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\ve
Marselino
Objek Dihapus!
```

Destruktor akan otomatis terpanggil tanpa dipanggil oleh pengguna sehingga mengeluarkan kalimat “Objek Dihapus”.

5. Setter and 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.

```
def get_gaji(self): #getter
    return self.__gaji
def set_gaji(self, gaji_baru): #setter
    self.__gaji = gaji_baru
```

6. Decorator








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

```
@property
def gaji(self): #getter
    print("Fungsi getter dipanggil")
    return self.__gaji
@gaji.setter
def gaji_(self, gaji_baru): #setter
    print("Fungsi setter dipanggil")
    self.__gaji = gaji_baru
```

Hasilnya adalah sebagai berikut :


```
26 print(Pemain1.nama) # memanggil atribut
27 print(Pemain1.gaji)
28 Pemain1.gaji += 5000
29 print(Pemain1.gaji)
```

Run:  main ×

  "C:\Users\RADOT NABABAN\PycharmProjects\
Marselino
 Fungsi getter dipanggil
13000
 Fungsi getter dipanggil
 Fungsi setter dipanggil
 Fungsi getter dipanggil
18000


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. Enkapsulasi

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. Perbedaan 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 :

```
File "C:\Users\Kuboy\AppData\Local\Temp\161663\pytest_1602\test_1602.py", line 16, in <module>
    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

```
Nasabah1= Nasabah("Tok Dalang", 13, 12890003, 100000)
```

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

```
#akses atribut  
print(Nasabah1.nama) #akan mengeluarkan nama dari nasabah 1
```

Hasilnya adalah sebagai berikut

Tok Dalang

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:

```
2 class FCB:
3     def __init__(self, nama, umur): # merupakan konstruktor
4         self.nama = nama # atribut dalam konstruktor
5         self.umur = umur
6
7     class Pelatih(FCB): #kelas turunan dari FCB
8         def __init__(self, nama, umur, tahun_kepelatihan):
9             super().__init__(nama, umur) #fungsi memakai atribut di FCB
10            self.tahun_kepelatihan = tahun_kepelatihan
11
12        def cetak(self):
13            print(f"{self.nama} merupakan pelatih FC Barcelona di tahun"
14                  f" {self.tahun_kepelatihan}")
15
16
17 Pelatih1 = Pelatih("Djajang Nurdjaman", 20, 2008)
18 Pelatih1.cetak()
19
```

Output yang dihasilkan adalah sebagai berikut

```
Djajang Nurdjaman merupakan pelatih FC Barcelona di tahun 2008

Process finished with exit code 0
```

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.

```

2 class FCB:
3     def __init__(self, nama, umur): # merupakan konstruktor
4         self.nama = nama # atribut dalam konstruktor
5         self.umur = umur
6
7     class Pelatih(FCB): #kelas turunan dari FCB
8         def __init__(self, nama, umur, tahun_kepelatihan):
9             super().__init__(nama, umur) #fungsi memakai atribut di FCB
10            self.tahun_kepelatihan = tahun_kepelatihan
11
12            def cetak(self):
13                print(f"{self.nama} merupakan pelatih FC Barcelona di tahun"
14                      f" {self.tahun_kepelatihan}")
15
16
17 Pelatih1 = Pelatih("Djajang Nurdjaman", 20, 2008)
18 Pelatih1.cetak()
19

```

Pembeda child class dari parent class tersebut adalah terletak di penambahan fungsi cetak, dan yang menandakan identik dan mengambil variabel dari class parent adalah terletak di kunci super().

b. Multiple Inheritance

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:

```

2 class FCB:
3     jumlah_staff = 0 # merupakan atribut kelas
4
5     def __init__(self, nama, umur, gaji): # merupakan konstruktor
6         self.nama = nama # atribut dalam konstruktor
7         self.umur = umur
8         self.gaji = gaji
9         self.jumlah_staff += 1
10
11     class Pemain(FCB):
12         def __init__(self, nama, umur, gaji, posisi, tahun_kontrak):
13             super().__init__(nama, umur, gaji)
14             self.posisi = posisi
15             self.tahun_kontrak = tahun_kontrak
16         def cetak(self):
17             print(f"{self.nama} merupakan pemain FC Barcelona dengan posisi {self.posisi}")
18
19     class Pelatih(FCB):
20         def __init__(self, nama, umur, gaji, formasi, asal_negara, tahun_kepelatihan):
21             super().__init__(nama, umur, gaji)
22             self.formasi = formasi
23             self.asal_negara = asal_negara
24             self.tahun_kepelatihan = tahun_kepelatihan
25         def cetak(self):
26             print(f"{self.nama} merupakan pelatih FC Barcelona di tahun {self.tahun_kepelatihan}")
27

```

Di gambar tersebut kelas FCB menurunkan kelas lain berupa kelas Pemain dan Kelas Pelatih. Kedua kelas turunan tersebut memakai atribut dari kelas FCB secara bersamaan.

Contoh dari penurunan bertingkat adalah sebagai berikut:

```
1 # contoh penerapan
2 class FCB:
3     def __init__(self, nama, umur, gaji): # merupakan konstruktor
4         self.nama = nama # atribut dalam konstruktor
5         self.umur = umur
6         self.gaji = gaji
7     # turunan dari kelas FCB
8     class Pemain(FCB):
9         def __init__(self, nama, umur, gaji, posisi):
10             super().__init__(nama, umur, gaji)
11             self.posisi = posisi
12         def cetak(self):
13             print(f"{self.nama} merupakan pemain FC Barcelona dengan posisi {self.posisi}")
14     # turunan dari kelas pemain
15     class CalonPemain(Pemain):
16         def __init__(self, nama, umur, gaji, tahun_kontrak, tim_sekarang):
17             super().__init__(nama, umur, gaji, tahun_kontrak)
18             self.tim_sekarang = tim_sekarang
19
20         def cetak(self):
21             print(f"{self.nama} merupakan calon pemain FC Barcelona yang sekarang berada di tim {self.tim_sekarang}")
22
23 Pemain1 = Pemain("Marselino", 20, 13000, "Right Winger")
24 Calon1 = CalonPemain("Leo Messi", 30, 15000, "Left Back", "Persib Bandung")
25 Calon1.cetak()
```

Gambar tersebut menurunkan **kelas Pemain** yang sudah diturunkan dari **kelas FCB** ke **kelas CalonPemain**. Penurunan di kelas CalonPemain ditambahkan kembali atribut tambahan berupa `tim_sekarang`. Jika di run, akan menghasilkan kalimat sebagai berikut

```
Run: PYTugas4PrakPBO x
"C:\Users\RADO\NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\Users\RADO\NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe"
Leo Messi merupakan calon pemain FC Barcelona yang sekarang berada di tim Persib Bandung
Process finished with exit code 0
```

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


```

1 class Kocheng():
2     def cetak(self):
3         print("Ini Kocheng ")
4 class KochengOren(Kocheng):
5     def cetak(self):
6         print("Ini Koceng Oren ygy")
7
8 Hewan = KochengOren()
9 Hewan.cetak()
10

```

Akan menghasilkan output sebagai berikut

```

Ini Koceng Oren ygy

Process finished with exit code 0

```

4. Casting

Casting memiliki beberapa jenis yaitu :

- a. 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

```

1 class Kocheng():
2     def tampil(self):
3         print(f"Ini Kocheng dengan pemilik bernama {self.pemilik} ")
4 class KochengOren(Kocheng):
5     def __init__(self, pemilik, umur):
6         super().__init__()
7         self.pemilik = pemilik
8         self.umur = umur
9     def cetak(self):
10        print("Ini Kocheng Oren ygy")
11
12 Hewan = KochengOren("Yanto", 13)
13 Hewan.tampil()
14

```

Jika di run, akan menampilkan hasil sebagai berikut :

```

Ini Kocheng dengan pemilik bernama Yanto

Process finished with exit code 0

```

- b. Upcasting, yaitu kelas child mengakses atribut yang ada di kelas parent, penerapannya yaitu sebagai berikut :

```
1 class Kocheng():
2     ras = "Eropa"
3     def __init__(self, ras):
4         self.ras = ras
5     class KochengOren(Kocheng):
6         def __init__(self, ras, pemilik, umur):
7             super().__init__(ras)
8             self.pemilik = pemilik
9             self.umur = umur
10        def cetak(self):
11            print(f"Ini Kocheng Oren dengan ras {super().ras}")
12
13
14 Hewan = KochengOren("Persia", "Yanto", 13)
15 Hewan.cetak()
16
```

Kode diatas akan menghasilkan output sebagai berikut :

```
Ini Kocheng Oren dengan ras Eropa
```

```
Process finished with exit code 0
```

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

```
1 # contoh penerapan magic method
2 class Ruangan:
3     def __init__(self):
4         self.list_siswa=["Fajar", "Anto", "Riyandi", "Muklis"] # menulis list siswa
5     def hapusSiswa(self):
6         self.list_siswa.pop() # menghapus siswa dari urutan paling akhir
7     def len(self):
8         return len(self.list_siswa) # memanggil banyaknya siswa
9
10 admin1 = Ruangan()
11 print(admin1.list_siswa)
12 admin1.hapusSiswa()
13 print(len(admin1))
14 print(admin1.list_siswa)
15
```

Run: lat3pbopy

```
"C:\Users\RADOT NABABAN\PycharmProjects\pytest_PB02\venv\Scripts\python.exe" "C:\User
['Fajar', 'Anto', 'Riyandi', 'Muklis']
3
['Fajar', 'Anto', 'Riyandi']
```