RESUME MATERI
PRAKTIKUM PBO RB
TUGAS MINGGU 5

Oleh:

Attar Akram Abdillah (121140013)

Program Studi Teknik Informatika

Institut Teknologi Sumatera

2023

Daftar Isi

1. Pengenalan Bahasa Python

Python adalah bahasa pemrograman yang dibuat oleh Guido Van Rossum pada tahun 1980-an. Python menjadi bahasa yang populer karena sintaksnya yang mudah, library yang melimpah, dan pengaplikasiannya yang luas. Python merupakan bahasa yang mementingkan keterbacaan dari suatu kode sehingga untuk mewujudkan itu Python tidak menggunakan kurung kurawal ataupun keyword dalam penulisan kodenya dan menggantikannya dengan spasi untuk memisahkan blok kode.

Sintaks yang dimiliki oleh bahasa pemrograman Python kurang lebih sama seperti dengan bahasa lainnya seperti bahasa C ataupun Java, hanya saja dituliskan dalam ekspresi yang berbeda. Contohnya untuk melakukan perbandingan, pengguna cukup menuliskan ‘elif’ dibandingkan ‘else if’ pada C++ untuk menambahkan kondisi pada perbandingan.

2. Objek dan Kelas Dalam Python

Pada Python dikenal konsep objek dan kelas. Konsep dari kelas dan objek ini akan digunakan dalam melakukan penerapan Object Oriented Programming (OOP) pada Python. Selain itu dikenal juga konsep-konsep seperti magic methods, konstruktor, destruktor, setter dan getter, dan decorator.

2.1. Kelas

Kelas atau class merupakan sebuah cetak biru dari suatu objek yang akan dibuat. Kelas akan memuat atribut, variabel, dan juga metode yang akan dimiliki oleh sebuah objek.

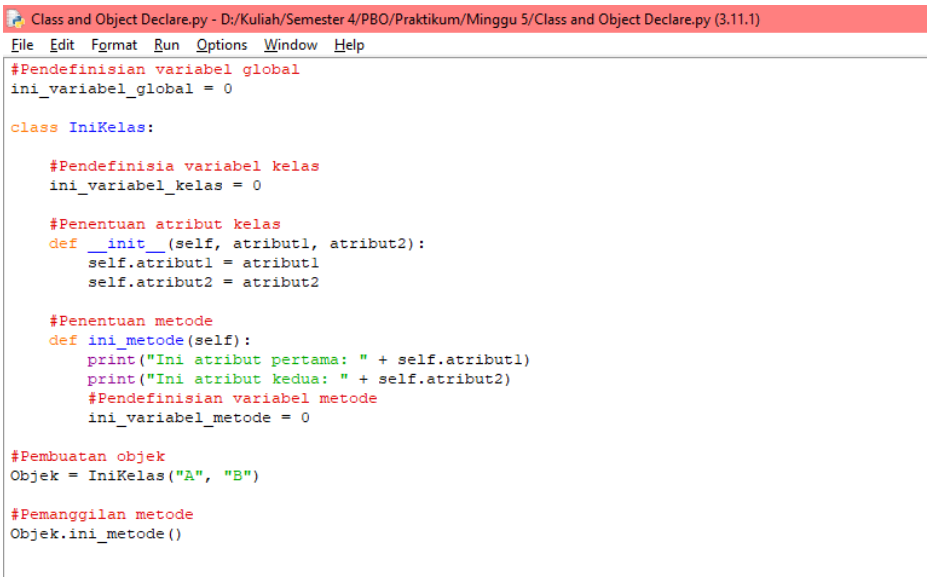
- Variabel

  Variabel adalah sesuatu yang mengandung nilai. Variabel ini dapat berupa variabel global, variabel kelas, ataupun variabel metode. Perberdaan antara ketiga jenis variabel tersebut adalah luas cakupan variabel tersebut berlaku.
- Atribut

  Atribut adalah suatu variabel yang akam ditentukan nilainya setiap pendeklarasian suatu objek. Nilai dari variabel atribut ini akan berbeda antara satu objek dengan objek lainnya.
- Metode

  Metode adalah suatu fungsi yang dimiliki dari suatu objek. Metode ini akan lebih dikenal sebagai fungsi pada bahasa pemrograman C.

Berikut contoh dari penerapan dari konsep kelas pada suatu program:



2.2. Objek

Objek pada Python berfungsi sebagai sesuatu yang akan mewakilkan sebuah kelas. Suatu objek akan memiliki atribut, metode, serta variabel dari sebuah kelas yang dimana ia termasuk ke dalamnya. Objek dideklarasikan dengan memanggil nama kelas dari objek dan menambahkan tanda kurung dibelakangnya, tanda kurung tersebut biasanya akan diisi dengan atribut-atribut dari objek tersebut.

Berikut contoh pembuatan dari sebuah objek:

```
#Pembuatan objek
Objek = IniKelas("A", "B")
```

3. Abstraksi

Abstraksi adalah konsep yang digunakan dalam pemrograman berbasis objek untuk memperlihatkan atribut yang penting dan menyembunyikan detail yang tidak penting bagi user. Ini dilakukan untuk mengurangi kompleksitas dari kode bagi user.

Berikut contoh penerapan dari konsep abstraksi:

Abstract.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 5/Abstract.py (3.11.1)

File Edit Format Run Options Window Help

```
#Import modul abstraksi
from abc import ABC
from abc import abstractmethod

#Pendefinisian kelas abstraksi
class Abstrak(ABC):
    def __init__(self, atribut):
        self.atribut = atribut

    #Pembuatan metode abstrak
    @abstracmethod
    def metode_abstrak(self):
        pass

#Pembuatan kelas turunan dari kelas abstrak
class Turunan(Abstrak):
    def __init__(self, atribut):
        super().__init__(atribut)

    #Pembuatan ulang metode abstrak supaya kode berjalan
    def metode_abstrak(self):
        print("Ini adalah turunan dari metode abstrak")
        print("Atribut dari objek ini adalah " + self.atribut)
```

4. Enkapsulasi

Enkapsulasi adalah suatu metode yang digunakan untuk menyembunyikan informasi tertentu dan melindungi informasi tersebut. Dalam konsep enkapsulasi terdapat tiga jenis hak akses, yaitu public, protected, dan private.

4.1. Public

Public access modifier merupakan bentuk dasar/umum dari sebuah variabel atau metode, hal tersebut dikarenakan ketika sebuah variabel atau metode dideklarasikan maka secara default akan menjadi public. Dengan public access modifier variabel atau metode dapat diakses dimana saja.

4.2. Protected

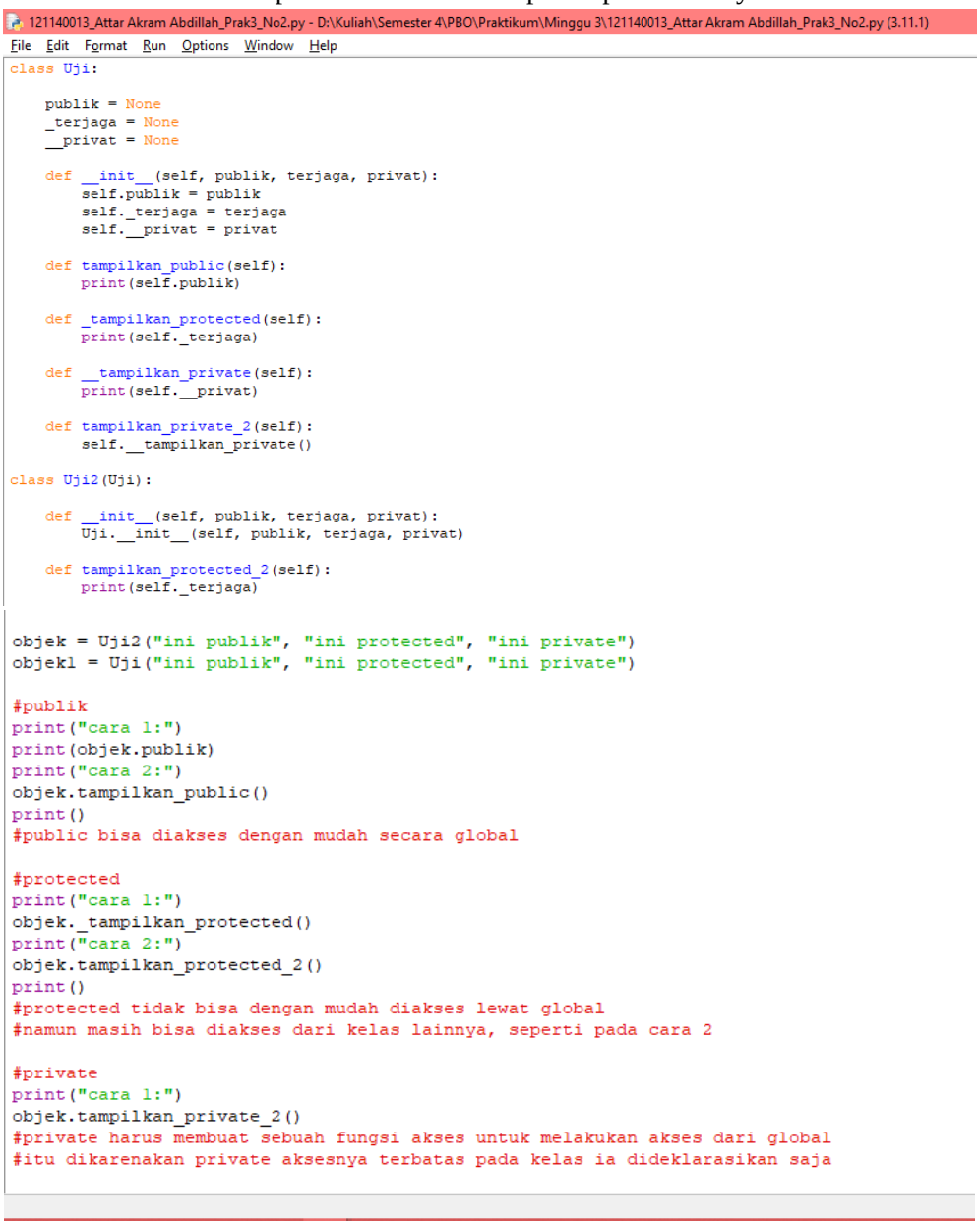
Protected access modifier berbeda dengan public access modifier karena pada hak akses ini sebuah variabel atau metode hanya bisa diakses dari kelas itu sendiri atau kelas turunannya. Untuk mendeklarasikan suatu variabel atau metode protected kita cukup menambahkan satu garis bawah (_) sebelum nama dari variabel tersebut.

4.3. Private

Pada private access modifier suatu variabel atau metode hanya dapat diakses di dalam kelas dimana variabel atau metode itu dideklarasikan. Untuk mendeklarasikan suatu variabel atau metode private kita cukup menambahkan dua garis bawah (__) sebelum nama dari variabel tersebut.

4.4. Implementasi Enkapsulasi

Berikut contoh implementasi dari konsep enkapsulasi di Python:



121140013_Attar Akram Abdillah_Prak3_No2.py - D:\Kuliah\Semester 4\PBO\Praktikum\Minggu 3\121140013_Attar Akram Abdillah_Prak3_No2.py (3.11.1)

File Edit Format Run Options Window Help

```
class Uji:

    publik = None
    _terjaga = None
    __privat = None

    def __init__(self, publik, terjaga, privat):
        self.publik = publik
        self._terjaga = terjaga
        self.__privat = privat

    def tampilkan_public(self):
        print(self.publik)

    def _tampilkan_protected(self):
        print(self._terjaga)

    def __tampilkan_private(self):
        print(self.__privat)

    def tampilkan_private_2(self):
        self.__tampilkan_private()

class Uji2(Uji):

    def __init__(self, publik, terjaga, privat):
        Uji.__init__(self, publik, terjaga, privat)

    def tampilkan_protected_2(self):
        print(self._terjaga)

objek = Uji2("ini publik", "ini protected", "ini private")
objek1 = Uji("ini publik", "ini protected", "ini private")

#publik
print("cara 1:")
print(objek.publik)
print("cara 2:")
objek.tampilkan_public()
print()
#public bisa diakses dengan mudah secara global

#protected
print("cara 1:")
objek._tampilkan_protected()
print("cara 2:")
objek.tampilkan_protected_2()
print()
#protected tidak bisa dengan mudah diakses lewat global
#namun masih bisa diakses dari kelas lainnya, seperti pada cara 2

#private
print("cara 1:")
objek.tampilkan_private_2()
#private harus membuat sebuah fungsi akses untuk melakukan akses dari global
#itu dikarenakan private aksesnya terbatas pada kelas ia dideklarasikan saja
```

4.5. Setter dan Getter

Dengan adanya pembatasan hak akses, tidak memungkinkan untuk kita melakukan akses ke suatu variabel atau metode dari suatu objek dengan mudah. Oleh karena itu, kita dapat mengimplementasikan konsep setter dan getter pada kode yang kita buat untuk melakukan akses terhadap variabel atau metode tersebut.

Berikut contoh implementasi dari setter dan getter pada kode di Python:

Setter Getter.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 5/Setter Getter.py (3.11.1)

File Edit Format Run Options Window Help

```
class Uji:
    def __init__(self, private):
        self.__private = private

    #Deklarasi metode getter
    def get_private(self):
        print(self.__private)

    #Deklarasi metode setter
    def set_private(self, update):
        self.__private = update

Objek = Uji("Ini Privat")

#Pemanggilan fungsi setter dan getter
Objek.get_private()
Objek.set_private("Dapat diakses")
Objek.get_private()
```

5. Inheritance

Inheritance adalah suatu konsep yang ada pada pemrograman berbasis objek yang dimana sebuah kelas dapat menurunkan atribut serta metodenya ke kelas lainnya. Kelas yang menjadi penyumbang atribut dan metode akan disebut sebagai parent class (kelas orang tua) dan kelas yang menerima sumbangan atribut dan metode akan disebut dengan child class (kelas anak).

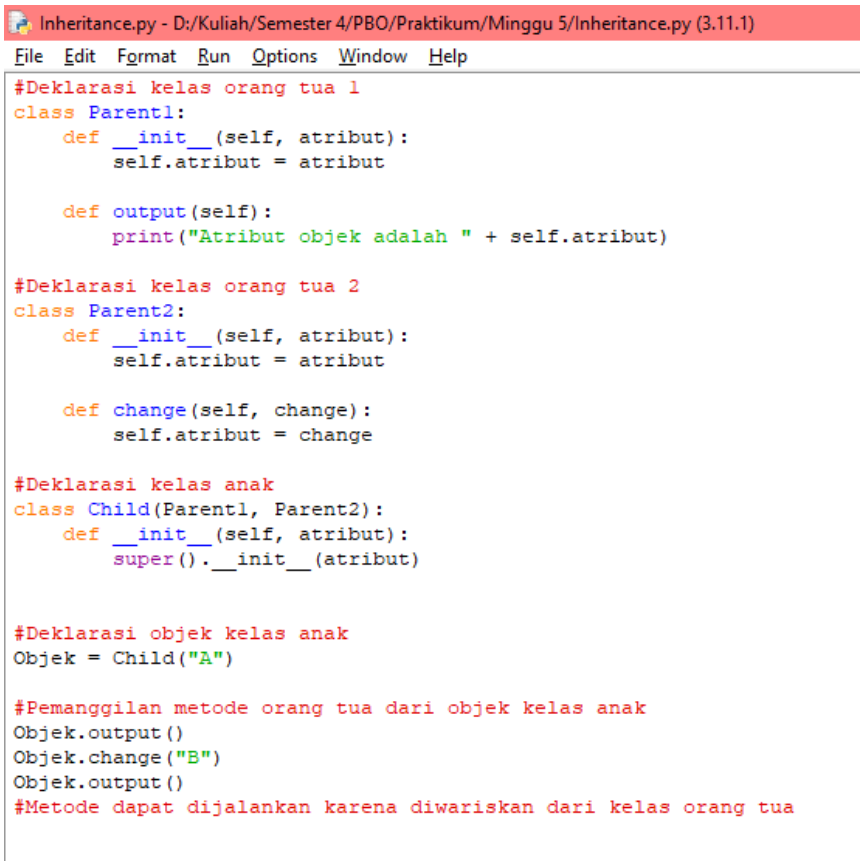
Berikut contoh penerapan dari konsep inherintance di Python:

Inheritance.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 5/Inheritance.py (3.11.1)

File Edit Format Run Options Window Help

```
#Deklarasi kelas orang tua
class Parent:
    def __init__(self, atribut):
        self.atribut = atribut

    def output(self):
        print("Atribut objek adalah " + self.atribut)


#Deklarasi kelas anak
class Child(Parent):
    def __init__(self, atribut):
        super().__init__(atribut)


#Deklarasi objek kelas anak
Objek = Child("A")

#Pemanggilan metode orang tua dari objek kelas anak
Objek.output()
#Metode dapat dijalankan karena diwariskan dari kelas orang tua
```

Kelas anak juga bisa mewarisi atribut serta metode dari beberapa orang tua. Kondisi ini disebut dengan multiple inheritance. Berikut contoh penerapan dari multiple inheritance:



6. Polymorphism

Polymorphism memungkinkan suatu interface yang sama untuk memerintah objek untuk melakukan aksi yang mungkin secara prinsip sama namun secara proses berbeda. Ada 4 cara dalam melakukan implementasi polymorphism ini yaitu dengan operator, fungsi, kelas, dan inheritance.

Berikut contoh penerapan dari konsep polymorphism di Python:

Polymorphism.py - D:/Kuliah/Semester 4/PBO/Praktikum/Minggu 5/Polymorphism.py (3.11.1)

File Edit Format Run Options Window Help

```
#Deklarasi kelas 1
class Mobil:
    def __init__(self, atribut):
        self.__atribut = atribut

    def berjalan(self):
        print("Vroooom....")

#Deklarasi kelas 2
class Motor:
    def __init__(self, atribut):
        self.__atribut = atribut

    def berjalan(self):
        print("Ngeeeeng....")

#Deklarasi objek tiap kelas
mobil = Mobil("RX7")
motor = Motor("R1")

#Pemanggilan fungsi yang sama pada tiap kelas
for kendaraan in (mobil, motor):
    kendaraan.berjalan()
```

Daftar Pustaka

Modul Praktikum PBO - 1
Modul Praktikum PBO - 2
Modul Praktikum PBO - 3
Modul Praktikum PBO - 4