RESUME
PEMROGRAMAN BERORIENTASI OBJEK

Oleh :

Vebie Yoseva Theresia Pasaribu **121140016**

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2023

1. PENGENALAN DAN DASAR PEMROGRAMAN PYTHON

1.1 Pengenalan Pemrograman Python

Python merupakan salah satu bahasa pemrograman yang diciptakan oleh Guido van Rossum pada tahun 1991 yang dimana python ini merupakan bahasa pemrograman yang mudah untuk dipelajari karena struktur dari sintaksnya mudah dipahami dan rapi. Bahasa pemrograman python mendukung banyak paradigma pemrograman seperti functional, object-orientedm dan structured. Python dapat digunakan dibanyak platform dan bahasa yang cukup popular yang meliputi *Data Science, Machine Learning, Internet of Things,* dan lainnya. Python juga merupakan bahasa pemrograman yang memiliki *library* yang luas yang dimana hal ini dapat memungkinkan programmer untu mengembangkan hingga kebidang lainnya.

1.2 Dasar Pemrograman Python

1.2.1 Variabel dan Tipe Data

Dalam pemrograman python, programmer dapat membangun suatu variable tanpa mendefinisikan tipe data dari variable tersebut. Tipe data merupakan penentu atau jenis dari sebuah data. Tipe data dapat berupa angka/numerik (integer, float), teks(string), karakter(char), Boolean(bool), dan lainnya.

Contoh :

```
nama = "Vebie Yoseva" #string
NIM = 121140016 #integer
BB = 45.3 #float
mahasisawa = true  #boolean
```

1.2.2 Operator

Dalam bahasa pemrograman python terdapat beberapa operator seperti :

- **Operator aritmatika,** memuat penjumlahan, pengurangan, perkalian, pembagian, dan sebagainya.
- **Operator perbandingan**, akan membandingkan 2 nilai dimana hasil perbandingannya adalah '*True*' atau '*False*' tergantung pada kondisi yang ada.
- **Operator penugasan,** berfungsi untuk memberikan nilai ke variabel.

- **Operator logika**, yang dapat digunakan untuk melakukan operasi logika.
- **Operator bitwise**, melakukan operasi bit terhadap operand.
- **Operator identitas**, akan memeriksa apakah nilai atau variabel berada pada posisi atau tempat yang sama.
- **Operator keanggotaan**, akan memeriksa kosndisi nilai atau variabel yang ada merupakan anggota atau terdapat didalam suatu data.

1.2.3 Tipe Data Bentukan

Dalam bahasa pemrograman python ada 4 tipe data bentukan, yaitu :

- List :
  - Tipe data ini mempunyai indeks dimuali dari 0
  - Datanya berurut dan tidak berubah
  - Dapat dilakukan perubahan
  - Memungkinkan adanya anggota yang sama
  - Ditandai dengan tanda kurung siku []
- Tuple :
  - Mempunyai indeks dimulai dai 0
  - Datanya berurut dan tidak berubah
  - Tidak dapat dilakukan perubahan
  - Memungkinkan adanya anggota yang sama
  - Ditandai dengan tanda kurung bulat ()
- Dictionary :
  - Disajikan berpasangan dengan kunci-nilai dan dapat dirujuk dengan nama kunci
  - Datanya berurutan dan tidak berubah
  - Dapat dilakukan perubahan
  - Tidak memungkinkan adanya anggota yang sama
  - Ditandai dengan tanda kurung kurawal {}

- Set :
  - Tidak mempunyai indeks
  - Datanya tidak berurutan
  - Tidak dapat dilakukan perubahan
  - Tidak memungkinkan adanya anggota yang sama
  - Ditandai dengan tanda kurung kurawal {}

1.2.4 Percabangan

Dalam percabangan akan dilakukan pengecekan terhadap kondisi yang ada. Jika kondisinya bernilai '*True*' maka akan menjalankan sintaks yang ada pada kolom aksi dan jika bernilai '*False*' tidak akan menjalankan blok aksi.

Contoh :

```
print("Masukkan nilai x: ")
x = int(input())

if  x > 0 :
    print ("bilangan positif")

elif x < 0 :
    print ("bilangan negatif")

else :
    print ("bilangan nol")
```

1.2.5 Perulangan

Dalam bahasa pemrograman python terdapat 2 jenis perulangan, yaitu perulangan for dan while. Sama seperti bahasa pemrograman lainnya, perulangan for mempunya perbedaan dengan perulangan while dimana perulangan for akan melakukan pengulangan sebanyak syarat yang sudah diketahui banyak perulangannya dan while melakukan perulangan dengan syarat yang diberikan dan tidak tentu akan terjadi berapa banyak perulangannya.

1.2.6 Fungsi

Dalam python ada 2 jenis fungsi, yaitu fungsi buatan dan bawaan python.

Fungsi ini berguna ketika program akan menjalankan operasi tertentu secara berkali- kali berdasarkan permintaan.

Contoh :

```
def greetings ():
    print ("hello")
```

2. OBJEK DAN KELAS DALAM PYTHON

2.1 Kelas (*Class*)

Kelas atau *Class* merupakan suatu prototipe yang bentuk pengguna untuk mendefenisikan suatu objek beserta atribut- atribut yang menjadi ciri dari objek pada kelas tersebut.

```
class Produk():
    jumlah_produk = 0

    def __init__(self, nama, harga):
        self.nama = nama
        self.harga = harga
        Produk.jumlah_produk += 1

    def berapa_jumlah(self):
        print('Total Produk Kita: ', Produk.jumlah_produk)

    def detail_produk(self):
        print("Nama : ", self.nama)
        print("Harga: ", self.harga)
        print()
```

2.1.1 Atribut/ Property

Dalam sebuah kelas dapat didirikan beberapa atribut. Ada 2 jenis atribut, yaitu atribut kelas dan atribut objek. Atribut kelas berbeda dengan atribut objek dimana atribut kelas dimiliki oleh sebuah kelas dan juga dimiliki oleh setiap objek dan sedangkan atribut objek merupakan atribut yang dimiliki oleh masing- masing objek.

- Atribut kelas

```
class Produk():
    jumlah_produk = 0
```

- Atribut objek

```
def __init__(self, nama, harga):
    self.nama = nama
    self.harga = harga
    Produk.jumlah_produk += 1
```

2.1.2 Method

Method merupakan jenis fungsi khusus yang didefinisikan dalam kelas. Semua atribut dan objek yang berada dalam kelas yang sama akan memiliki method yang sama.

```
def berapa_jumlah(self):
    print('Total Produk Kita: ', Produk.jumlah_produk)

def detail_produk(self):
    print("Nama : ", self.nama)
    print("Harga: ", self.harga)
    print()
```

2.2 Objek

Objek merupakan sesuatu atau wujud yang didefinisikan dalam suatu kelas. Objek dapat digunakan untuk mewakili sebuah kelas yang dimana akan mempermudah ketika kelas dipanggil.

```
produk1 = Produk('kuaci', 500)

produk2 = Produk('kacang', 300)
```

2.3 Magic method

Magic method merupakan salah satu method yang digunakan untk mengubah sifat dari suatu objek. Megic method tidak dipanggil secara langsung, tetapi dipanggil sistem secara internal.

2.4 Setter and Getter

Setter merupakan metode yang berfungsi utnuk mengubah atribut yang dimiliki sebuah objek dan Getter merupakan metode yang berfungsi untuk mengakses atribut atau metode dari sebuah objek.

```
class Student:
    _school_name = 'Institut Teknologi Sumatera'

    def __init__(self, full_name, current_age):
        self.__name = full_name
    self.__age = current_age

    def getName(self): # method getter untuk nama
        return self.__name

    def getAge(self): # method getter untuk usia
        return self.__age

    def setName(self, new_name):
        self.__name = ner_name

Student1 = Student("Eko", 30)
print(Studen1.getName())
print(Studen1.getAge())
```

2.5 Decorator

Dalam python, decorator dapat diterapkan diatas fungsi untuk memperluas fungsionalitas dari fungsi yang mendasarinya dan mendukung fleksibilitas dalam memanipulasi fungsi yang ada. Decorator dapat digunakan untuk memodifikasi suatu fungsi menggunakan fungsi lain. Terdapat 3 dekorator bawaan, yaitu property, staticmethod, classmethod.

3. ABSTRAKSI DAN ENKAPSULASI

3.1 Abstraksi

Abstraksi dalam python dapat diterapkan melalui kelas yang dibuat. Abstraksi berguna untuk mengurangi kompleksitas dengan hanya memperlihatkan atribut dan menyembunyikan detail yang kurang penting dari pengguna. Dengan itu, pengguna hanya mengetahui apa yang dikerjakan oleh objek, tetapi tidak mengetahui seperti apa mekanisme yang terjadi dibelakangnya.

3.2 Enkapsulasi

Enkapsulasi merupakan sebuah metode atau teknik yang dapat berguna untuk menyembunyikan detail dari sebuah atribut dalam sebuat kelas. Hal tersebut berguna untuk mengurangi kemungkinan untuk diakses melalui luar kelas sehingga elemen penting dapat lebih terjaga.

Untuk membatasi hal tersebut, ada 3 jenis *access modifier* dalam bahasa pemrograman python, yaitu :

- *Public Access Modifier :*
  - Tanpa tanda atau tanpa gatis bawah
  - Atribut atau *method* dapat diakses dari luar
- *Protected Access Modifier :*
  - Ditandai dengan garis bawah tunggal (_) dibagian depan atribut atau *method*
  - Atribut dan *method* hanya dapat diakses dari dalam kelas dan turunan kelasnya
- *Private Access Modifier :*
  - Ditandai dengan garis bawah ganda(__) dibagian depan atribut atau *method*
  - Atribut dan *method* hanya dapat diakses dari dalam kelasnya saja.

```
class Sepeda:
    def __init__(self):
        self.nama = "Monarch" #atribut public
        self._tipe = "Mountain Bike" #atribut protected
        self.__kecepatan = 0 #atribut privat

    def __atur_ulang_kecepatan(self): #privat
        self.__kecepatan = 0
```

4. PEWARISAN DAN POLIMORFISME

4.1 *Inheritance* (Pewarisan)

Dalam python terdapat konsep pewarisan, dimana objek dapat dibuat sebagai induk dari objek- objek lainnya dan setiap objek turunan akan memiliki sifat dan perilaku yang sama dengan objek induk. Pewarisan juga dapat diterapkan pada kelas, suatu kelas dapat menjadi kelas induk dari kelas yang diturunkan.

Pewarisan juga dapat menambahkan konstruktor pada kelas turunan yang nantinya kelas turunan akan mempunya konstruktor sendiri tanpa menghilangkan kosntruktor pada kelas induknya.

```
class Pelajar :
    def __init__(self, nama, kelas):
        self.nama = nama
        self.kelas = kelas

    def biodata(self):
        print()
        print()
        print ("FORM:")
        print()
        print (f"Nama: {self.nama}\nKelas: {self.kelas}")

class Kursus(Pelajar):
    def __init__(self,nama, kelas, mapel, ruangan):
        super().__init__(nama, kelas)
        self.mata_pelajaran = mapel
        self.ruangan = ruangan

    def mendaftar(self):
        print("---melakukan pendaftaran---")
        print(f"Mata Pelajaran: {self.mata_pelajaran}\nRuangan: {self.ruangan}")

    def keluar(self):
        print("---telah menyelesaikan kursus---")
        print(f"Mata Pelajaran: {self.mata_pelajaran}")

daftar = Kursus ("vebie", "6", "Matematika", "RA01")
daftar2 = Kursus("yoseva", "5", "Fisika", "RB03")
daftar.biodata()
daftar.mendaftar()

daftar2.biodata()
daftar2.keluar()
```

4.2 Polymorphism

Pada python, *Polymorphism* menggunakan konsep *duck typing* yaitu mengidentifikasi objek dari perilakunya bukan dari tipe datanya. Selama function yang berjalan bisa memanggil *method* pada objek yang ingin dipanggil, ini dapat mengasumsikan bahwa objek tersebut sudah benar tanpa mengecek tipe datanya.

4.3 Override/Overriding

Overriding pada pewarisan kondisi dimana kelas turunan mempunyai *method* yang sama kelas induk baik nama, parameter, maupun *return type*. Akan tetapi, fungsi yang dimiliki kedua kelas tersebut adalah berbeda. Dengan *overriding*, *method* yang ada pada kelas turunan dapat dimodifikasi atau diganti total jika *method* tersebut tidak sesuai dengan kelas turunan.

4.4 Overloading

Overloading pada perwarisan ini merupakan proses dimana *Polymorphism* dapat membuat beberapa *method* atau fungsi dengan nama yang sama, tetapi parameter yang dimiliki berbeda.

4.5 Multiple Inheritance

Dalam pyhton dapat membuat pewarisan banyak kelas. Sebuah kelas turunan dapat memiliki lebih dari satu kelas induk. Pewarisan ini dapat dilakukan hingga beberpa turunan deperti kelas turunan dari kelas dasar dan turunannya dapat diwarisi ke dalam kelas turunan yang baru dibuat.

4.6 Method Resolution Order

Ini merupakan metode untuk melakukan pencarian *method* dalam kelas hirarki dimana *method* akan dicari pertama kali di kelas objek dan jika tidak ditemukan akan dilanjutkan pencarian ke kelas lainnya (*multiple inheritance*). *Method Resolution Order* memiliki urutan python yaitu bawah-atas dan kiri- kanan.

4.7 Dynamic Cast

Dynamic Cast merupkan proses yang berguna untuk mengubahkan nilai dari suatu tipe data ke tipe data lainnya. Ada 2 tipe perngubahan, yaitu implisit dan eksplisit. Implisit dimana konversi tipe data ke tipe data lainnya akan dilakukan secara otomatis dengan python tanpa campur tangan pengguna dan ekplisit, dimana pengguna yang akan melakukan konversi tipe data objek ke tipe data lainnya dengan fungsi bawaan python dan ini dapat berisiko kehilangan data.

4.8 Casting

Ada beberapa *casting* dalam pyhton, seperti :

- *Downcasting,* dimana atribut dari kelas turunan diakses oleh kelas induk.
- *Upcasting,* dimana atribut yang dimiliki kelas induk diakses oleh kelas turunan.
- *Type Casting,* melakukan konversi tipe kelas agar memiliki sifat/perilaku tertentu yang secara default tidak dimiliki oleh kelas tersebut.

REFERENSI

Modul Praktikum Pemrograman Berorientasi Objek Minggu 1

Modul Praktikum Pemrograman Berorientasi Objek Minggu 2

Modul Praktikum Pemrograman Berorientasi Objek Minggu 3

Modul Praktikum Pemrograman Berorientasi Objek Minggu 4

https://jagongoding.com/python/dasar/overview/

https://www.ekrut.com/media/7-dasar-bahasa-pemrograman-python

https://belajarpython.com/tutorial/object-class-python/

https://ediweb.dev/python/memahami-class-pada-python

https://minarsih.com/artikel/belajar-python-lanjutan-29-class-abstraction

https://jagongoding.com/python/menengah/oop/pewarisan/