RESUME MATERI PRAKTIKUM PBO
MODUL 1-4

Nama : Juan Verrel Tanuwijaya
NIM : 121140072

Kelas : RB

I. Modul 1

1. Bahasa Pemograman Python :

Python adalah sebuah bahasa pemrograman tingkat tinggi yang diciptakan oleh Guido van Rossum pada akhir tahun 1980-an. Python memiliki sintaks yang mudah dibaca dan dipahami, sehingga cocok untuk pemula dalam dunia pemrograman. Bahasa Python mendukung paradigma pemrograman seperti object-oriented, functional, dan structured, dan memiliki banyak modul/library yang memudahkan dalam pengembangan aplikasi. Python juga dapat digunakan untuk berbagai jenis pengembangan aplikasi, seperti desktop, mobile, web, otomatisasi, hacking, IoT, dan robotika. Keunggulan bahasa Python adalah mudah untuk dipelajari dan dipahami, memiliki sintaks yang bersih dan rapi, serta mampu mengatasi banyak masalah kompleks dengan mudah.

2. Sintaks Dasar :

- Statement :
  Dalam Python, sebuah perintah atau instruksi yang dapat dieksekusi disebut statement. Secara default, sebuah statement dianggap berakhir ketika ada baris baru (newline) di dalam kode. Namun, Python memungkinkan pembuatan statement yang terdiri dari beberapa baris dengan menggunakan tanda backslash ().
- Python tidak menggunakan kurung kurawal sebagai penanda blok kode yang dikelompokkan, melainkan menggunakan spasi atau tab (4 spasi). Untuk kode yang berada di dalam blok yang sama, harus memiliki jumlah spasi yang sama pada awal barisnya.
- Variabel adalah tempat penyimpanan yang digunakan untuk menampung data atau nilai tertentu. Saat mendeklarasikan sebuah variabel dalam pemrograman, penting untuk mengetahui tipe data yang berkaitan dengan variabel yang akan dideklarasikan.
- Python memiliki sejumlah operator, yaitu :
  -Operator Aritmatika
  -Operator Perbandingan
  -Operator Penugasan
  -Operator Logika
  -Operator Bitwise
  -Operator Identitas
  -Operator Keanggotaan
- Ada 4 tipe data bentukan dalam python, yaitu :
  -List
  -Tuple
  -Set
  -Dictionary
- Percabangan dan perulangan adalah konsep dasar dalam pemrograman, termasuk dalam bahasa pemrograman Python. Percabangan atau "if statement" adalah salah satu bentuk struktur kontrol dalam Python, yang memungkinkan program untuk memilih tindakan yang berbeda berdasarkan kondisi tertentu. Perulangan atau "loop" adalah struktur kontrol lainnya dalam Python, yang memungkinkan program untuk mengeksekusi blok kode tertentu secara berulang. Dalam Python, terdapat dua jenis perulangan: "for loop" dan "while loop".
- Fungsi adalah sebuah blok kode yang dibuat untuk melakukan tugas tertentu yang spesifik, dan dapat dipanggil dari tempat lain dalam program. Fungsi biasanya menerima masukan atau input dalam bentuk argumen, melakukan operasi tertentu pada argumen tersebut, dan mengembalikan hasilnya sebagai keluaran atau output. Dalam Python, sebuah fungsi didefinisikan dengan kata kunci "def" diikuti oleh nama fungsi dan argumen yang diinginkan di dalam kurung.

3. Contoh Kode

```
#Inisiasi variabel coba sebagai batas percobaan login
coba=1
#Dibuat percabangan while dengan kondisi coba lebih kecil atau sama
dengan 3
while coba<=3:
    #Penginputan variabel user dan pw(password)
    user=input("Username anda : ")
    pw=input("Password anda : ")

    #Kondisi jika input user dan pw benar akan berhasil login
    if user=="informatika" and pw=="12345678":
        print("Berhasil login!")
        break

    #Jika input user atau password salah
    else:
        coba+=1
        #Jika jumlah percobaan sudah 3 kali gagal
        if coba==4:
            print("Akun anda diblokir!")
        #Jika jumlah percobaan belum 3 kali gagal
        else:
            print("Username atau password salah coba lagi")
```

II. Modul 2

1. Class

Dalam pemrograman berorientasi objek, "class" adalah blueprint atau rancangan untuk menciptakan objek. Objek dalam hal ini adalah entitas yang memiliki karakteristik atau atribut tertentu serta perilaku atau method tertentu. Di Python, class didefinisikan menggunakan kata kunci "class", diikuti dengan nama class.

- Atribut/Property
  Dalam pemrograman berorientasi objek, "atribut" adalah variabel yang ada di dalam sebuah class yang digunakan untuk menyimpan nilai atau informasi yang berhubungan dengan objek yang dibuat dari class tersebut. Di Python, atribut class dapat didefinisikan di dalam method "init" atau method lain di dalam class, dan diakses menggunakan operator titik (.) pada objek yang telah dibuat dari class tersebut.
- Method
  Dalam Python, method adalah fungsi yang didefinisikan di dalam sebuah class dan beroperasi pada objek dari class tersebut. Method didefinisikan untuk melakukan operasi pada objek, mengubah nilai atribut dari objek, atau mengembalikan nilai dari objek. Method dapat diakses melalui objek yang dibuat dari class tersebut. Saat sebuah method dipanggil pada objek, maka objek akan menjadi argumen pertama yang secara otomatis dikirim ke dalam method tersebut, yang biasanya dinamakan

dengan kata "self". Dalam method, self digunakan untuk mengakses atribut dari objek itu sendiri.

2. Objek

Objek adalah suatu instance atau perwujudan dari class. Dalam Python, class merupakan blueprint atau cetak biru untuk membuat objek dengan karakteristik atau perilaku tertentu. Setiap objek memiliki atribut dan method yang ditentukan oleh class-nya. Atribut adalah data yang disimpan dalam objek, sedangkan method adalah fungsi yang terkait dengan objek dan digunakan untuk mengoperasikan objek tersebut. Dalam Python, objek dibuat dengan menggunakan constructor yaitu method yang bernama __init__() dan akan dijalankan secara otomatis ketika objek dibuat. Constructor digunakan untuk menginisialisasi nilai atribut pada objek yang dibuat.

3. Magic Method

Magic method atau dikenal juga dengan nama special method adalah method yang dimulai dan diakhiri dengan dua garis bawah atau double underscore (__). Magic method digunakan untuk mengimplementasikan fitur-fitur khusus dalam class Python, seperti operator overloading, konversi tipe data, dan akses ke atribut.

4. Konstruktor

Konstruktor adalah sebuah method khusus dalam Python yang secara otomatis dipanggil pada saat sebuah objek dibuat dari sebuah kelas. Konstruktor dalam Python biasanya digunakan untuk menginisialisasi (menetapkan nilai awal) atribut-atribut dari objek yang akan dibuat.

5. Destruktor

Destruktor dalam Python adalah sebuah method khusus yang dipanggil secara otomatis ketika sebuah objek dari suatu kelas dihapus atau di-"destroyed" dari memori. Destruktor umumnya digunakan untuk membersihkan sumber daya yang dipesan oleh objek ketika objek tersebut tidak diperlukan lagi, seperti menutup koneksi database atau melepaskan alokasi memori yang digunakan oleh objek.

6. Setter dan Getter

Setter dan getter adalah metode atau fungsi yang digunakan untuk mengatur atau mendapatkan nilai dari atribut dalam sebuah kelas. Setter merupakan metode yang digunakan untuk mengatur atau mengubah nilai dari suatu atribut dalam kelas. Setter biasanya digunakan untuk melakukan validasi atau pengolahan data sebelum nilai atribut diubah. Getter merupakan metode yang digunakan untuk mendapatkan nilai dari suatu atribut dalam kelas. Getter biasanya digunakan untuk mengakses nilai atribut yang di-private atau di-protected pada kelas

7. Decorator

Decorator dalam Python adalah fungsi yang digunakan untuk mengubah fungsi lain dengan menambahkan fungsionalitas tambahan ke dalamnya, tetapi tanpa mengubah struktur aslinya. Decorator dapat diterapkan pada fungsi, metode, ataupun class.

8. Contoh Kode

```
#Pembuatan class Mahasiswa
class Mahasiswa:
    #Konstruktor dengan atribut dari class Mahasiswa
    def __init__(self, nama, nim, kelas_pbo_siakad,jumlah_sks):
        self.nama=nama
        self.nim=nim
```

```
15.         self.kelas=kelas_pbo_siakad
16.         self.sks=jumlah_sks
17.
18.     #Method untuk menampilkan info atribut
19.     def infodiri(self):
20.         print(f"Nama = {self.nama}\nNim = {self.nim}\nKelas PBO Siakad
    = {self.kelas}\nJumlah SKS = {self.sks}")
21.
22.     #Method untuk mengubah atribut kelas
23.     def ubahkelas(self,kelasbaru):
24.         self.kelas=kelasbaru
25.
26.#Inisiasi Objek Manusia dengan kelas Mahasiswa
27.manusia=Mahasiswa("Juan", "121140072", "RB", "4 SKS")
28.#Pemanggilan Method
29.manusia.infodiri()
30.manusia.ubahkelas("RA")
31.manusia.infodiri()
```

III. Modul 3

1. Abstraksi

Abstraksi dalam class adalah sebuah konsep pemrograman berorientasi objek yang memungkinkan pengguna untuk hanya fokus pada informasi penting atau relevan dari suatu objek atau sistem, dan mengabaikan detail implementasi yang lebih kompleks. Dalam class Python, abstraksi dapat dicapai melalui penggunaan metode abstrak dan kelas abstrak.

2. Enkapsulasi

Enkapsulasi adalah konsep dalam pemrograman berorientasi objek yang memungkinkan pembatasan akses langsung ke beberapa komponen objek, dan hanya mengizinkan akses melalui metode yang telah ditentukan. Dalam Python, enkapsulasi dapat dicapai dengan membuat atribut dan method dengan modifier akses tertentu.

Pada Python, terdapat tiga jenis modifier akses, yaitu:

- Public: Atribut atau method yang dapat diakses dari mana saja, baik dari dalam maupun luar class. Didefinisikan tanpa menggunakan underscore pada awal nama atribut atau method.
- Protected: Atribut atau method yang hanya dapat diakses dari dalam class atau subclassnya. Didefinisikan dengan menggunakan satu underscore pada awal nama atribut atau method.
- Private: Atribut atau method yang hanya dapat diakses dari dalam class tempat mereka didefinisikan. Didefinisikan dengan menggunakan dua underscore pada awal nama atribut atau method.

3. Contoh Kode

```
#Inisiasi Class dan Atribut
class Mobil:
    #Inisiasi atribut kelas yang merupakan variabel konstan yang dimiliki oleh semua objek yang dibuat.
    tipekendaraan="berat"
    def __init__(self,nama,pabrik,tahun):
        #Inisiasi atribut Public, atribut publik dapat diakses didalam maupun diluar kelas itu sendiri, cara mendaklarasikannya adalah dengan menulis nama variabel biasa secara default.
        self.nama = nama
        #Inisiasi atribut Private, atribut Private hanya dapat diakses didalam kelas itu sendiri sehingga tidak dapat diakses diluar kelas tersebut, cara mendeklarasikannya adalah dengan menambahkan "__" didepan variabel atribut.
        self.__pabrik = pabrik
        #Inisiasi atribut Protected, atribut Protected hanya dapat diakses oleh kelas turunannya sendiri, cara mendeklarasikannya adalah dengan menambahkan "_" didepan nama variabel atribut, akan tetapi dalam python atribut protected tetap dapat diakses diluar kelasnya sendiri seperti atribut public.
        self._tahun = tahun

    def printprivate(self):
        print(self.__pabrik)

mobil1=Mobil("Zenix", "Toyota", 2014)
mobil2=Mobil("Camaro", "Chevrolet", 2017)

#Dikarenakan atribut nama merupakan atribut publik, maka dapat diakses tanpa menggunakan fungsi dari dalam kelas tersebut dan langsung dapat di print dari luar kelas tersebut.
print(mobil1.nama)
#Sedangkan untuk atribut pabrik hanya dapat diakses menggunakan fungsi dari dalam kelas tersebut dikarenakan tidak dapat diakses dari luar kelasnya sendiri.
mobil1.printprivate()
#Dan atribut Protected tetap dapat diakses diluar kelasnya sendiri pada Python
print(mobil1._tahun)

#Print atribut kelas yang memiliki nilai konstan oleh semua objek kelas tersebut
print("Mobil 1 :", mobil1.tipekendaraan)
print("Mobil 2 :", mobil2.tipekendaraan)
```

IV. Modul 4

1. Inheritance

Inheritance dalam class adalah salah satu konsep OOP (Object Oriented Programming) yang memungkinkan sebuah class untuk mewarisi atribut dan metode dari class lain yang disebut superclass atau parent class. Dalam inheritance, sebuah class baru (yang disebut subclass atau child class) dibuat berdasarkan pada class yang sudah ada (superclass atau parent class). Subclass dapat mengakses atribut dan metode yang diwarisi dari superclass, dan juga dapat menambahkan atribut dan metode baru atau mengubah atribut dan metode yang sudah ada. Dalam Python, inheritance didefinisikan dengan menempatkan nama superclass di dalam tanda kurung setelah nama subclass.

2. Polymorphism

Polymorphism dalam class adalah konsep OOP (Object Oriented Programming) di mana sebuah objek dapat memiliki banyak bentuk atau perilaku yang berbeda-beda, tergantung pada konteks atau lingkungan di mana objek tersebut digunakan. Dalam Python, polymorphism sering terjadi ketika sebuah class memiliki metode dengan nama yang sama dengan metode di class lain. Metode-metode tersebut dapat diimplementasikan dengan cara yang berbeda-beda di setiap class, dan dapat dipanggil dengan cara yang sama tanpa perlu mengetahui implementasi masing-masing class.

3. Override / Overriding

Override/Overriding dalam class adalah konsep di mana sebuah subclass (child class) dapat mengubah atau menimpa implementasi metode yang sudah ada di superclass (parent class). Dalam Python, override dilakukan dengan mendefinisikan ulang metode superclass di dalam subclass dengan nama yang sama. Ketika objek dari subclass dipanggil untuk memanggil metode tersebut, metode yang terdapat di subclass akan dijalankan dan mengabaikan metode yang ada di superclass.

4. Overloading

Overloading dalam class adalah konsep di mana sebuah class memiliki beberapa metode dengan nama yang sama tetapi berbeda dalam jumlah atau jenis parameter yang diterima. Dalam Python, overloading tidak didukung secara langsung, karena Python mendukung penggunaan parameter yang bersifat opsional dan fleksibel. Namun, kita dapat menggunakan beberapa trik untuk membuat metode dengan perilaku yang terlihat seperti overloading. Salah satu cara untuk melakukan overloading di Python adalah dengan menggunakan parameter *args atau **kwargs. Parameter *args digunakan untuk menerima argumen berupa tuple yang dapat berisi banyak nilai, sedangkan **kwargs digunakan untuk menerima argumen berupa dictionary.

5. Multiple Inheritance

Multiple inheritance dalam class adalah konsep di mana sebuah subclass (child class) dapat mewarisi sifat dari dua atau lebih superclass (parent class) secara bersamaan. Hal ini memungkinkan subclass untuk memiliki sifat yang kompleks dan dapat mengkombinasikan fitur-fitur yang ada di banyak superclass. Dalam Python, multiple inheritance dapat diimplementasikan dengan menentukan dua atau lebih superclass yang ingin diwarisi oleh subclass dalam deklarasi class.

6. Method Resolution Order (MRO)

Method Resolution Order (MRO) di Python adalah aturan yang digunakan untuk menentukan urutan pengecekan metode (method lookup order) dalam multiple inheritance. Dalam Python, setiap class memiliki daftar superclass yang mungkin berisi beberapa superclass. MRO menentukan urutan dalam pencarian metode yang dilakukan oleh interpreter Python ketika subclass mengakses atribut atau metode dari superclass. Pencarian dimulai dari kelas saat ini dan dilanjutkan ke atas

hingga kelas teratas. MRO memastikan bahwa setiap metode hanya dipanggil sekali dalam urutan warisan.

7. Dynamic Cast

Operasi dynamic cast digunakan untuk mengubah tipe data dari suatu objek atau pointer ke tipe data yang lain secara dinamis, tergantung pada tipe data yang dimiliki oleh objek atau pointer pada saat runtime.

8. Casting

Casting dalam bahasa Python merujuk pada proses mengubah tipe data suatu objek menjadi tipe data yang berbeda. Python merupakan bahasa pemrograman yang dinamis, artinya tipe data dari suatu variabel atau objek ditentukan pada saat runtime, sehingga casting dalam Python tidak perlu dilakukan secara eksplisit seperti di beberapa bahasa pemrograman lainnya. Dalam Python, tipe data dari suatu objek dapat diubah secara implisit melalui operasi yang dilakukan pada objek tersebut.

9. Contoh Kode

```
class Komputer:
    def __init__(self,nama,jenis,harga,merk):
        self.nama=nama
        self.jenis=jenis
        self.harga=harga
        self.merk=merk

class Processor(Komputer):
    def __init__(self,nama,jenis,harga,merk,jumlah,kecepatan):
        super().__init__(nama,jenis,harga,merk)
        self.jumlah_core=jumlah
        self.kecepatan_core=kecepatan

    def tampil(self):
        print(f"Processor {self.nama} produksi {self.merk}")

class RAM(Komputer):
    def __init__(self,nama,jenis,harga,merk,kapasitas):
        super().__init__(nama,jenis,harga,merk)
        self.kapasitas=kapasitas

    def tampil(self):
        print(f"RAM {self.nama} produksi {self.merk}")

class HDD(Komputer):
    def __init__(self,nama,jenis,harga,merk,kapasitas,rpm):
        super().__init__(nama,jenis,harga,merk)
        self.kapasitas=kapasitas
        self.rpm=rpm

    def tampil(self):
        print(f"SATA {self.nama} produksi {self.merk}")
```

```
42.
43.class VGA(Komputer):
44.    def __init__(self,nama,jenis,harga,merk,kapasitas):
45.        super().__init__(nama,jenis,harga,merk)
46.        self.kapasitas=kapasitas
47.
48.    def tampil(self):
49.        print(f"VGA {self.nama} produksi {self.merk}")
50.
51.class PSU(Komputer):
52.    def __init__(self,nama,jenis,harga,merk,daya):
53.        super().__init__(nama,jenis,harga,merk)
54.        self.daya=daya
55.
56.    def tampil(self):
57.        print(f"PSU {self.nama} produksi {self.merk}")
58.
59.#main
60.rakit=[]
61.
62.p1 = Processor('Core i7 7740X',"Baru",4350000,'Intel',4,'4.3GHz')
63.p2 = Processor('Ryzen 5 3600',"Second",250000,'AMD', 4,'4.3GHz')
64.ram1 = RAM('DDR4 SODimm PC19200/2400MHz',"Baru",328000,'V-Gen','4GB')
65.ram2 = RAM('DDR4 2400MHz',"Second ",328000,'G.SKILL', '4GB')
66.hdd1 = HDD('HDD 2.5 inch',"Baru", 295000,'Seagate','500GB',7200)
67.hdd2 = HDD('HDD 2.5 inch',"Second", 295000,'Seagate', '1000GB',7200)
68.vga1 = VGA('VGA GTX 1050',"Baru",'Asus',250000,'2GB')
69.vga2 = VGA('1060Ti', "Second", 250000,'Asus','8GB')
70.psu1 = PSU('Corsair V550', "Baru", 250000,'Corsair','500W')
71.psu2 = PSU('Corsair V550', "Second", 250000,'Corsair','500W')
72.
73.rakit.extend([[p1,ram1,hdd1,vga1,psu1],[p2,ram2,hdd2,vga2,psu2]])
74.
75.for x in range(len(rakit)):
76.    print("Komputer", x+1, ":")
77.    for y in range(len(rakit[x])):
78.        rakit[x][y].tampil()
79.    print()
```