

LAPORAN PRAKTIKUM PBO

Resume Materi Minggu 1 – 4



Disusun oleh:

Nama : Riksan Cahyowadi

NIM : 121140106

Kelas : PBO (RB)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN

2023

Pertemuan 1

A. Variabel dan tipe data

Di dalam bahasa pemrograman Python, variabel digunakan untuk menyimpan nilai yang akan digunakan dalam program. Python adalah bahasa pemrograman yang memiliki sifat dinamis, sehingga kita tidak perlu menentukan tipe data variabel saat kita membuat variabel. Ada 4 tipe data primitive yang ada di python yaitu:

- int (integer), tipe data ini digunakan untuk menyimpan bilangan bulat
- str (string), tipe data yang digunakan untuk menyimpan teks
- float, tipe data yang digunakan untuk menyimpan bilangan pecahan
- bool (Boolean), tipe data yang digunakan untuk menyimpan nilai True/False (kebenaran)

B. Operator

Operator pada pemrograman Python adalah simbol atau karakter khusus yang digunakan untuk melakukan operasi pada nilai atau variabel dalam program Python. Dalam pemrograman, operator digunakan untuk melakukan berbagai tugas, seperti melakukan perhitungan matematika, membandingkan nilai, menggabungkan string, dan sebagainya.

Ada 7 jenis operator pada python yaitu :

1. Operator aritmatika, operator yang digunakan untuk melakukan operasi matematika, seperti penjumlahan, pengurangan, perkalian, pembagian, dan sebagainya.
 - Operator penjumlahan (+)
 - Operator pengurangan (-)
 - Operator perkalian (*)
 - Operator pembagian (/)
 - Operator modulus (%)
 - Operator pangkat (**)
 - Operator pembagian bilangan bulat (//)
2. Operator perbandingan, operator yang digunakan untuk membandingkan 2 buah nilai.
 - Operator sama dengan (==)
 - Operator tidak sama dengan (!=)
 - Operator lebih besar dari (>)
 - Operator lebih kecil dari (<)
 - Operator lebih besar dari sama dengan (>=)
 - Operator lebih kecil dari sama dengan (<=)
3. Operator logika, operator yang digunakan untuk melakukan operasi logika.
 - Operator logika AND (and)
 - Operator logika OR (or)

- Operator logika NOT (not)
4. Operator Assignment, operator yang digunakan untuk memberi nilai ke variabel.
 - Operator penugasan (=)
 - Operator penugasan tambah (+=)
 - Operator penugasan kurang (-=)
 - Operator penugasan kali (*=)
 - Operator penugasan bagi (/=)
 - Operasi penugasan modulus (%=)
 - Operator penugasan pangkat (**=)
 - Operator penugasan bagi bulat (//=)
 5. Operator Identitas, operator yang memeriksa apakah dua buah nilai (atau variabel) berada pada lokasi memori yang sama.
 - Operator identitas sama (is)
 - Operator identitas tidak sama (is not)
 6. Operator Keanggotaan, operator yang digunakan untuk memeriksa apakah suatu nilai atau variabel merupakan anggota atau ditemukan di dalam suatu data (string, list, tuple, set, dan dictionary).
 - Operator keanggotaan (in)
 - Operator bukan keanggotaan (not in)
 7. Operator bitwise, operator yang melakukan operasi bit terhadap operand. Operator ini beroperasi bit per bit sesuai dengan namanya.
 - Operator bitwise AND (&)
 - Operator bitwise OR (|)
 - Operator bitwise NOT (~)
 - Operator bitwise XOR (^)
 - Operator bitwise right shift (>>)
 - Operator bitwise left shift (<<)

C. Percabangan

Percabangan adalah salah satu konsep dasar dalam pemrograman yang memungkinkan program untuk membuat keputusan berdasarkan kondisi tertentu. Dalam Python, percabangan dapat diimplementasikan menggunakan pernyataan if, elif, dan else.

1. Percabangan IF

Kode blok akan dieksekusi jika kondisi bernilai true. Contoh :

```
main.py
1 harga = 10000
2 uangSaya = 5000
3
4 if harga > uangSaya:
5     print("Maaf, uang anda kurang")
6 |
```

2. Percabangan IF-ELSE

Pernyataan if-else digunakan untuk mengeksekusi blok kode tertentu jika kondisi yang diberikan benar (True), dan blok kode lain jika kondisi salah (False). Contoh:

```
main.py
1 harga = 5000
2 uangSaya = 10000
3
4 if harga > uangSaya:
5     print("Maaf, uang anda kurang")
6 else:
7     print("Uang anda cukup untuk membeli barang ini")
```

3. Percabangan IF-ELIF-ELSE

Pernyataan if-elif-else digunakan untuk mengeksekusi blok kode tertentu jika beberapa kondisi yang berbeda diberikan benar (True). Jika tidak ada satu pun kondisi yang benar, maka blok kode else akan dieksekusi. Contoh:

```
main.py
1 harga = 5000
2 uangSaya = 10000
3
4 if harga > uangSaya:
5     print("Maaf, uang anda kurang")
6
7 elif uangSaya == harga:
8     print("Uang anda Pas untuk membeli barang ini")
9 else:
10    print("Uang kembalian anda sebesar " + str(uangSaya - harga))
```

D. Perulangan

Pada Python, terdapat dua jenis perulangan yaitu perulangan for dan perulangan while.

1. Perulangan for

Perulangan for digunakan untuk melakukan iterasi pada setiap elemen pada suatu objek seperti list, tuple, atau string. Contoh perulangan for adalah sebagai berikut:

```
main.py
1 games = ["Ender Lilies", "Hollow Knight", "Elden Ring"]
2
3 for ListGame in games:
4     print(ListGame)
```

2. Perulangan While

Perulangan while digunakan untuk mengeksekusi blok kode selama kondisi yang diberikan benar (True). Contoh perulangan while dan outpunya adalah sebagai berikut:

```
main.py
1 # Program untuk menghitung jumlah bilangan dari 1 sampai n
2 angka = int(input("Beri inputan nilai n: "))
3
4 i = 1
5 total = 0
6
7 while i <= angka:
8     total += i
9     i += 1
10
11 print("Jumlah bilangan dari 1 sampai", angka, "adalah", total)
12
```

Outputnya :

```
✓ ↗ 📄  
Beri inputan nilai n: 7  
Jumlah bilangan dari 1 sampai 7 adalah 28  
  
...Program finished with exit code 0  
Press ENTER to exit console.█
```

E. Fungsi

Fungsi pada Python adalah sebuah blok kode yang dapat digunakan untuk melakukan tugas tertentu atau menghasilkan nilai tertentu. Fungsi pada Python dapat menerima input (dalam bentuk parameter) dan mengembalikan output (dalam bentuk nilai kembalian).

Fungsi pada Python ditentukan dengan menggunakan kata kunci "def" diikuti dengan nama fungsi dan parameter (jika ada), dan diakhiri dengan tanda titik dua. Blok kode yang terdapat di dalam fungsi diindentasi ke kanan. Berikut adalah contoh fungsi:

```
main.py  
1 def luasPersegi(panjang, lebar):  
2     luas = panjang * lebar  
3     return luas
```

Pertemuan 2

A. Class (Kelas)

Class merupakan blueprint untuk membuat objek. Class menyediakan struktur dasar untuk objek dan mendefinisikan variabel dan metode yang dimiliki oleh objek tersebut. Dengan menggunakan class, kita dapat membuat banyak objek dengan struktur dan fitur yang sama.

Pendefinisian class dimulai dengan kata kunci "class" diikuti dengan nama class, dan diakhiri dengan tanda titik dua. Setiap class biasanya terdapat variabel (atribut/properti) dan fungsi (method). Berikut contoh class:

```
main.py
1 class penduduk:
2     def __init__(self, nama, pekerjaan, domisili):
3         self.nama = nama
4         self.pekerjaan = pekerjaan
5         self.domisili = domisili
6
7     def cetakInfo(self):
8         print(self.nama, "memiliki pekerjaan sebagai ", self.pekerjaan,
9               " dan bertempat tinggal di ", self.domisili)
10
11 Warga1 = penduduk("Riksan", "Web developer", "Merbau Mataram")
12 Warga2 = penduduk("\nCahyowadi", "Wiraswasta", "Tanjung Bintang")
13
14 Warga1.cetakInfo()
15 Warga2.cetakInfo()
```

Pada contoh class tersebut terdapat method dan attrribut. Penjelasan Method dan Atribut adalah sebagai berikut

1. Method

Method pada Python adalah sebuah fungsi yang terkait dengan sebuah objek dari suatu class. Method biasanya digunakan untuk mengakses atau memanipulasi data pada sebuah objek. Method pada Python didefinisikan dalam class dan dipanggil pada objek dari class tersebut. Pada program diatas yang merupakan method adalah def cetakInfo()

2. Atribut

Atribut pada Python adalah sebuah variabel yang didefinisikan dalam suatu class dan berfungsi sebagai data yang terkait dengan objek dari class tersebut. Setiap objek dari suatu class memiliki atribut yang unik, meskipun mungkin memiliki nilai yang sama dengan objek lain dari class yang sama.

Atribut dibagi menjadi 2 yaitu atribut class dan atribut objek. Atribut class adalah atribut yang terkait dengan class itu sendiri dan dapat diakses melalui class tersebut. Sedangkan atribut objek adalah atribut yang terkait dengan objek dari suatu class dan

dapat diakses melalui objek tersebut. Pada program diatas yang merupakan atribut adalah nama, pekerjaan, dan domisili.

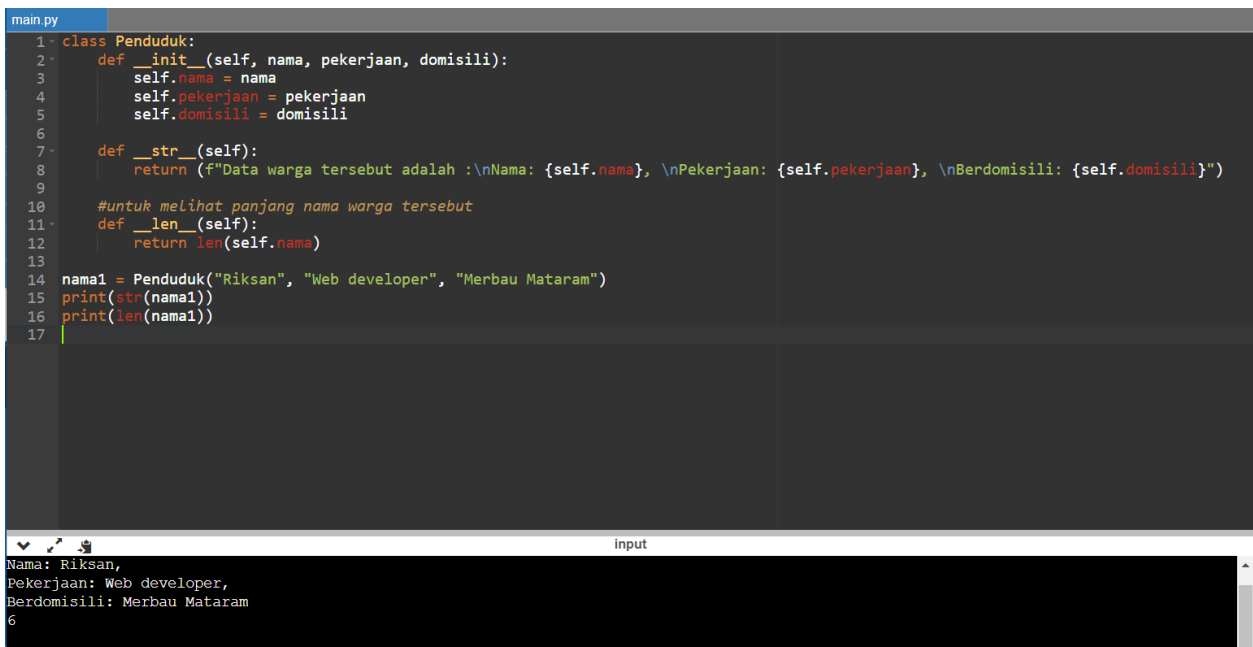
B. Objek

Objek merupakan suatu entitas yang terbentuk dan “mewakiliki” dari sebuah class atau tipe data tertentu. Objek Dapat dibuat dengan cara membuat instance(instance object) dari suatu class. Instance object adalah objek yang dibuat dari suatu class dengan menggunakan konstruktor class, yaitu method "init". Setelah objek terbentuk, kita dapat mengakses atribut dan metode objek tersebut menggunakan notasi titik (dot notation). Pada program diatas yang merupakan objek adalah Warga1 dan Warga2.

C. Magic Method

Magic method adalah method khusus yang dimiliki oleh sebuah objek yang didefinisikan dalam sebuah class dengan menggunakan double underscore atau dikenal juga dengan dunder (double underscore) method.

Magic method digunakan untuk melakukan tindakan tertentu secara otomatis pada sebuah objek ketika sebuah operasi atau tindakan tertentu dilakukan pada objek tersebut. Sebagai contoh, saat menginstansiasi sebuah objek, Python secara otomatis memanggil method khusus "init" yang didefinisikan di dalam class. Contoh program dengan Magic Method:



```
main.py
1 class Penduduk:
2     def __init__(self, nama, pekerjaan, domisili):
3         self.nama = nama
4         self.pekerjaan = pekerjaan
5         self.domisili = domisili
6
7     def __str__(self):
8         return (f'Data warga tersebut adalah :\nNama: {self.nama}, \nPekerjaan: {self.pekerjaan}, \nBerdomisili: {self.domisili}')
9
10    #untuk melihat panjang nama warga tersebut
11    def __len__(self):
12        return len(self.nama)
13
14    nama1 = Penduduk("Riksan", "Web developer", "Merbau Mataram")
15    print(str(nama1))
16    print(len(nama1))
17
```

input

Nama: Riksan,
Pekerjaan: Web developer,
Berdomisili: Merbau Mataram
6

D. Constructor

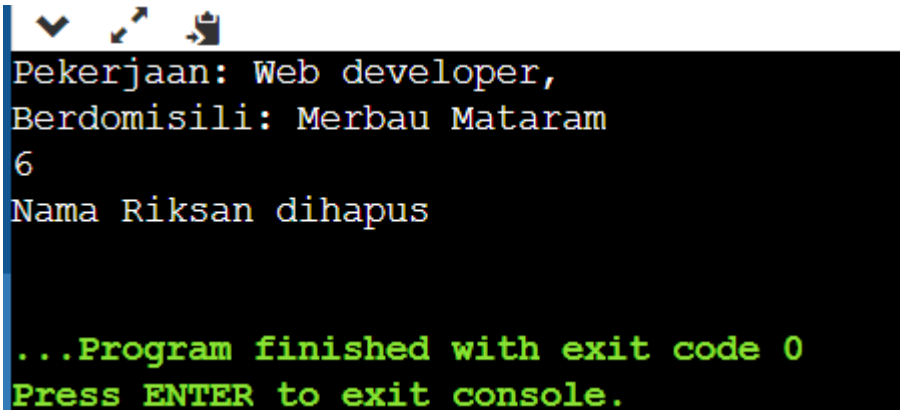
Constructor adalah method khusus dalam sebuah class di Python yang secara otomatis dipanggil saat objek dibuat. Constructor biasanya digunakan untuk menginisialisasi variabel instance dalam sebuah class. Nama constructor di Python selalu sama dengan nama class dan diawali dengan __init__(self).

E. Destruktor

Destructor adalah sebuah method khusus dalam sebuah class di Python yang dipanggil saat objek dihapus dari memori. Destructor digunakan untuk membersihkan sumber daya yang digunakan oleh objek sebelum objek tersebut dihapus dari memori. Pada python sendiri, destructor dipanggil menggunakan “__del__”. Contoh program destructor:

```
main.py
1 class Penduduk:
2     def __init__(self, nama, pekerjaan, domisili):
3         self.nama = nama
4         self.pekerjaan = pekerjaan
5         self.domisili = domisili
6
7     def __str__(self):
8         return (f'Data warga tersebut adalah :\nNama: {self.nama}, \nPekerjaan: {self.pekerjaan}, \nBerdomisili: {self.domisili}')
9
10    #untuk melihat panjang nama warga tersebut
11    def __len__(self):
12        return len(self.nama)
13
14    def __del__(self):
15        print(f'Nama {self.nama} dihapus')
16
17    nama1 = Penduduk("Riksan", "Web developer", "Merbau Mataram")
18    print(str(nama1))
19    print(len(nama1))
20
```

Outputnya :



```
Pekerjaan: Web developer,
Berdomisili: Merbau Mataram
6
Nama Riksan dihapus

...Program finished with exit code 0
Press ENTER to exit console.
```

Objek “self.nama” dihapus oleh __del__() dan mencetak perintahnya.

F. Setter dan Getter

Setter dan getter adalah dua metode yang digunakan dalam pemrograman berorientasi objek untuk mengakses dan mengubah nilai variabel private dalam sebuah kelas. Setter digunakan untuk mengatur atau mengubah nilai variabel private, sedangkan getter digunakan untuk mengambil atau mengakses nilai variabel private. Contoh program setter dan getter:


```
main.py
1 class Penduduk:
2     def __init__(self, gaji = 8000000):
3         self._gaji = gaji
4
5     #Getter
6     def getGaji(self):
7         return self._gaji
8
9     #Setter
10    def setGaji(self, value):
11        self._gaji = value
12
13    nama1 = Penduduk()
14    nama1.setGaji = 8888888
15    print(str(nama1.setGaji))
16
```

Outputnya :

```
88888888

...Program finished with exit code 0
Press ENTER to exit console.
```

Nilai gaji berubah menggunakan setter dan getter, yang semula gaji 8000000 menjadi 8888888.

Pertemuan 3

A. Abstraksi

Abstraksi salah satu konsep penting dalam penerapan Object Oriented Programming(OOP). Abstraksi memungkinkan programmer untuk menyembunyikan detail implementasi dari sebuah kelas dan hanya menampilkan fitur-fitur yang relevan bagi pengguna kelas tersebut, sedangkan enkapsulasi memungkinkan programmer untuk menyembunyikan data dan metode yang tidak perlu diketahui oleh pengguna kelas tersebut. Contoh program dengan abstraksi:

```
main.py
1 from abc import ABC, abstractmethod
2
3 class Manusia(ABC):
4     @abstractmethod
5     def berjalan(self):
6         pass
7
8 class Hewan(ABC):
9     def berjalan(self):
10         print("Hewan tersebut sedang berjalan")
11
12 Kucing = Hewan()
13 print(Kucing.berjalan()) #output : Hewan tersebut sedang berjalan
```

B. Enkapsulasi

Enkapsulasi memungkinkan programmer untuk menyembunyikan data dan metode yang tidak perlu diketahui oleh pengguna kelas. Dengan menyembunyikan akses dari luar kelas tersebut, elemen penting yang terdapat dalam kelas dapat lebih terjaga, dan menghindari kesalahan jika elemen tersebut diubah secara tidak sengaja.

Ada 3 jenis Access Modifier pada enkapsulasi, yaitu:

- Public Access Modifier
Modifier ini dapat diakses dari dalam class ataupun dari luar class. Biasanya secara default, semua object akan berjenis Public.
- Protected Access Modifier
Modifier ini hanya dapat diakses dari dalam kelas dan turunan kelasnya saja. Disimbolkan dengan 1 underscore(_)
- Private Access Modifier
Modifier ini hanya dapat diakses dari dalam kelasnya saja. Disimbolkan dengan 2 underscore (__). Apabila kita mencoba mengakses dari luar kelas maka program akan menghasilkan AttributeError.

Contoh program dengan Enkapsulasi:

main.py

```
1 class Biodata:
2     def __init__(self, nama, umur):
3         self.public_nama = nama
4         self._protected_umur = umur
5
6     def public_method(self):
7         print("Ini adalah public method.")
8
9     def _protected_method(self):
10        print("Ini adalah protected method.")
11
12    def __private_method(self):
13        print("Ini adalah private method.")
14
15 biodata = Biodata("Riksan", 20)
16 print(biodata.public_nama) #Output: Riksan
17 print(biodata._protected_umur) #Output: 20
18 print(biodata.__private_gender) #Output: object has no attribute
19 biodata.public_method()
20 biodata._protected_method()
21 biodata._Biodata__private_method()
```

Pertemuan 4

A. Inheritance

Inheritance adalah konsep yang memungkinkan suatu class untuk mewarisi sifat-sifat atau metode dari class lain yang lebih umum atau induk. Class yang mewarisi sifat-sifat atau metode dari class induk disebut subclass atau child class, sedangkan class yang memberikan sifat-sifat atau metode kepada subclass disebut superclass atau parent class. Dalam Python, untuk membuat subclass, kita dapat mendefinisikan class baru dan menyebutkan superclass sebagai parameter dalam definisi class. Contoh program Inheritance:

```
main.py
1 class Manusia:
2     def __init__(self, nama, umur):
3         self.nama = nama
4         self.umur = umur
5
6 class Mahasiswa(Manusia):
7     def info(self):
8         print(f>Nama mahasiswa: {self.nama}')
9         print(f'Umur: {self.umur}')
```

10
11 nama1 = Mahasiswa("Riksan", 20)
12 nama1.info() #Output: Nama mahasiswa: Riksan
13 # Umur: 20

B. Polymorphism

Polymorphism adalah konsep dalam pemrograman objek yang memungkinkan objek untuk memiliki banyak bentuk atau perilaku. Dalam pemrograman Python, polymorphism dapat dicapai dengan 2 cara:

- Method overriding
- Method overloading.

Method overriding adalah ketika subclass memiliki metode dengan nama yang sama dengan metode di superclass, sehingga subclass dapat mengganti perilaku metode tersebut dengan perilaku yang berbeda.

Sedangkan method overloading adalah ketika sebuah class memiliki beberapa metode dengan nama yang sama tetapi dengan jumlah dan tipe parameter yang berbeda. Dalam Python, method overloading tidak didukung secara langsung, namun dapat dicapai dengan menggunakan argumen opsional atau argumen default.

Contoh program polymorphism:

main.py

```
1 class Luas:
2     def hitung_luas(self):
3         pass
4
5 class Lingkaran(Luas):
6     def __init__(self, r):
7         self.jari_jari = r
8
9     def hitung_luas(self):
10        return 3.14 * self.jari_jari ** 2
11
12 class Persegi(Luas):
13     def __init__(self, s):
14         self.sisi = s
15
16     def hitung_luas(self):
17        return self.sisi ** 2
18
19 bentuk1 = Lingkaran(11)
20 bentuk2 = Persegi(15)
21
22 print("Luas lingkaran: ", bentuk1.hitung_luas()) #Output: Luas lingkaran = 379.94
23 print("Luas persegi: ", bentuk2.hitung_luas()) #Output: Luas persegi = 225
24
```