Nama : Fathurrahman Al Hafid

NIM : 12114088

Tugas : Praktikum PBO (RB)

1. Abstraksi

   Abstraksi Adalah kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

   Contoh Implementasi konsep Abtraksi: Mobil adalah kumpulan sistem pengapian, sistem kemudi, sistem pengereman. Orang tidak perlu berpikir bahwa mobil adalah kumpulan dari puluhan atau ribuan bagian-bagian kecil (mesin, kabel, baut,roda, rem, dsb). Orang hanya perlu berpikir bahwa mobil adalah sebuah objek yang telah memiliki prilaku spesifik, yang dapat digunakan sebagaialat transportasi, sehingga dia/mereka tinggal menggunakannya atau mengendarainya tanpa harus mengetahui kerumitan proses yang terdapat didalam mobil tersebut. Ini artinya, si pembuat mobil telah menyembunyikan semua kerumitan-kerumitan proses yang terdapat didalam mobil dan pengguna tidak perlu mengetahui bagaimana sistem mesin , transmisi, danrem berkerja. Konsep seperti inilah yang dinamakan dengan abstraksi.

Contoh Codingan:

```
class Mahasiswa:  # Public     "Mahasiswa": Unknown word.
    global_variable_jumlah = 0     "jumlah": Unknown word.

    def __init__(self, nama, semester):
        self.nama = nama
        self.semester = semester
        Mahasiswa.global_variable_jumlah = Mahasiswa.global_variable_jumlah + 1

    def printjumlah(self):     "printjumlah": Unknown word.
        print("total mahasiswa adalah ",     "mahasiswa": Unknown word.
              Mahasiswa.global_variable_jumlah, " Orang")     "Mahasiswa": Unknown

    def printprofil(self):     "printprofil": Unknown word.
        print("Nama : ", self.nama)
        print("Semester: ", self.semester)


Mhs1 = Mahasiswa("Indah", 8)     "Mahasiswa": Unknown word.

Mhs1.printprofil()     "printprofil": Unknown word.
Mhs1.printjumlah()     "printjumlah": Unknown word.
```

2. Enkapsulasi

Enkapsulasi adalah cara menyembunyikan detail tentang cara kerja kelas sehingga bagian lain dari program Anda tidak perlu mengetahuinya. Enkapsulasi adalah cara untuk melindungi properti dan metode kelas agar tidak diakses oleh sumber luar. Ini dapat membantu memastikan bahwa hanya informasi tertentu tentang kelas yang dapat diakses, yang membuatnya lebih mudah untuk dipahami dan digunakan.

Enkapsulasi memungkinkan kita untuk mengontrol properti dan metode mana yang dapat diakses oleh program lain. Ini mencegah kesalahan saat seseorang mencoba mengubah properti atau metode yang tidak dimaksudkan untuk diubah oleh program lain.

- Enkapsulasi Objek: Public, Protected dan Private

Untuk membatasi hak akses kepada property dan method di dalam sebuah class, Objek Oriented Programming menyediakan 3 kata kunci, yakni Public, Protected dan Private. Kata kunci ini diletakkan sebelum nama property atau sebelum nama method.

a. Pengertian Hak Akses: Public

Ketika sebuah property atau method dinyatakan sebagai public, maka seluruh kode program di luar class bisa mengaksesnya, termasuk class turunan.

Contoh codingan:

```
class Mobil:
    def __init__(self, merk, tahun):
        self.merk = merk  # atribut publik
        self.tahun = tahun  # atribut private

mobil1 = Mobil("Toyota", 2010)
print(mobil1.merk)  # Output: Toyota
print(mobil1.tahun)
```

b. Pengertian Hak Akses: Protected

Jika sebuah property atau method dinyatakan sebagai protected, berarti property atau method tersebut tidak bisa diakses dari luar class, namun bisa diakses oleh class itu sendiri atau turunan class tersebut. Apabila kita mencoba mengakses protected property atau protected method dari luar class, akan menghasilkan error.

Contoh codingan:

```
class Kendaraan:
    def __init__(self, merk, tahun):
        self._merk = merk  # atribut protected
        self.__tahun = tahun  # atribut private

class Mobil(Kendaraan):
    def __init__(self, merk, tahun, warna):
        super().__init__(merk, tahun)
        self.warna = warna  # atribut publik

    def get_merk(self):
        return self._merk

mobil1 = Mobil("Toyota", 2010, "biru")
print(mobil1.get_merk())  # Output: Toyota
```

c. Pengertian Hak Akses: Private

Hak akses terakhir dalam konsep enkapsulasi adalah private. Jika sebuah property atau method di-set sebagai private, maka satu-satunya yang bisa mengakses adalah class itu sendiri. Class lain tidak bisa mengaksesnya, termasuk class turunan.

Contoh codingan:

```
class Mobil:  # Private
    __merk = None
    __warna = None

    def __init__(self, merk, warna):
        self.__merk = merk
        self.__warna = warna

    def _tampilMobil(self):    "tampil": Unknown word.
        print("Merk mobil: ", self.__merk)    "Merk": Unknown word.
        print("Warna Mobil: ", self.__warna)    "Warna": Unknown word.


car1 = Mobil("Toyota", "Merah")    "Merah": Unknown word.
car1._tampilMobil()    "tampil": Unknown word.
```

3. Inheritance (Pewarisan)

   Dalam pewarisan, satu kelas dapat mewarisi beberapa karakteristik dan metode dari kelas lain. Ini terkadang disebut superclass atau kelas induk. Subclass, atau kelas anak, dibuat dengan mewarisi karakteristik dan metode tersebut dari superclass. Dalam pewarisan, subclass dapat mengakses dan menggunakan atribut dan metode superclass. Ini dapat mempermudah pembuatan subkelas tertentu atau untuk menyertakan fitur tambahan daripada superkelas.

   Di Python, Anda bisa menggunakan warisan untuk membuat tipe objek baru yang didasarkan pada tipe objek yang sudah ada.

   Contoh codingan:

```
class Rumah:
    def __init__(self, biaya_hidup, jumlah_penghuni):
        self.biaya_hidup = biaya_hidup
        self.jumlah_penghuni = jumlah_penghuni

    def pengeluaran(self):
        return self.biaya_hidup * self.jumlah_penghuni


class Toko:
    def __init__(self, pendapatan, jumlah_hari):
        self.pendapatan = pendapatan
        self.jumlah_hari = jumlah_hari

    def hitung_pendapatan(self):
        return self.pendapatan * self.jumlah_hari


class Ruko(Rumah, Toko):
    def __init__(self, biaya_hidup, jumlah_penghuni, pendapatan, jumlah_hari):
        Rumah.__init__(self, biaya_hidup, jumlah_penghuni)
        Toko.__init__(self, pendapatan, jumlah_hari)

    def hitung_laba_bersih(self):
        print("Toko memiliki laba bersih",
              self.hitung_pendapatan() - self.pengeluaran())


Herman = Ruko(2000000, 3, 3000000, 22)
print(Herman.pengeluaran())
print(Herman.hitung_pendapatan())
Herman.hitung_laba_bersih()
```

4. Poliomorphism

Polimorfisme adalah fitur OOP yang memungkinkan objek berperilaku berbeda tergantung pada konteks penggunaannya. Misalnya, Anda dapat memanggil objek "anjing" satu kali, dan "kucing" di lain waktu, dan menerima hasil yang berbeda karena objek tersebut dapat dilihat sebagai anjing atau kucing dalam situasi yang berbeda.

Polimorfisme dapat dilakukan dengan dua cara - dengan membebani (menambahkan metode dengan nama yang sama, tetapi dengan parameter atau operasi yang berbeda) atau mengganti (mengganti atau mengganti metode di kelas induk dengan kelas anak).

Contoh codingan:

```
class Hewan:    "Hewan": Unknown word.
    def __init__(self, nama):
        self.nama = nama

    def suara(self):    "suara": Unknown word.
        pass

class Anjing(Hewan):    "Anjing": Unknown word.
    def suara(self):    "suara": Unknown word.
        return "Guk guk!"

class Kucing(Hewan):    "Kucing": Unknown word.
    def suara(self):    "suara": Unknown word.
        return "Meow!"

hewan1 = Anjing("Spike")    "hewan": Unknown word.
hewan2 = Kucing("Garfield")    "hewan": Unknown word.

print(hewan1.nama + " bersuara " + hewan1.suara())  # Output: Spike bersuara Guk guk!
print(hewan2.nama + " bersuara " + hewan2.suara())  # Output: Garfield bersuara Meow!
```

5. Python

Python adalah bahasa pemrograman yang diciptakan pada tahun 1980-an oleh Guido Van Rossum. Ini memiliki berbagai paradigma pemrograman yang berbeda, dan menjadi semakin populer karena sintaksnya yang mudah dan banyak pustaka dan modul yang tersedia.

- Tipe data
  a. Primitif

  Di dalam bahasa python terdapat beberapa tipe data primitif yaitu:

Tipe Data	Jenis	Nilai
bool	Boolean	True atau false
int	Bilangan bulat	Seluruh bilangan bulat
float	Bilangan real	Seluruh bilangan real
string	Teks	Kumpulan karakter

Contoh pendeklarasian variabel dalam Python :

  b. Bentukan

Ada 4, dengan perbedaan penggunaan:

- List

  Sebuah kumpulan data yang terurut, dapat diubah, dan memungkinkan ada anggota yang sama
- Tuple

  Sebuah kumpulan data yang terurut, tidak dapat diubah, dan memungkinkan ada anggota yang sama
- Set

  Sebuah kumpulan data yang tidak berurutan, tidak terindeks, dan tidak memungkinkan ada anggota yang sama
- Dictionary

  Sebuah kumpulan data yang tidak berurutan, dapat diubah, tidak memungkinkan ada anggota yang sama

- Operator

  Python memiliki sejumlah operator, yaitu:

  a. Operator aritmatika

Operator	Nama dan Fungsi	Contoh
+	Penjumlahan, menjumlahkan 2 buah operand	x + y
–	Pengurangan, mengurangkan 2 buah operand	x – y
*	Perkalian, mengalikan 2 buah operand	x * y
/	Pembagian, membagi 2 buah operand	x / y

**	Pemangkatan, memangkatkan bilangan	x **y
//	Pembagian bulat, menghasilkan hasil bagi tanpa koma	x // y
%	Modulus, menghasilkan sisa pembagian 2 bilangan	x % y

b. Operator perbandingan

>	Lebih besar dari – Hasilnya True jika nilai sebelah kiri lebih besar dari nilai sebelah kanan	x > y
<	Lebih kecil dari – Hasilnya True jika nilai sebelah kiri lebih kecil dari nilai sebelah kanan	x < y
==	Sama dengan – Hasilnya True jika nilai sebelah kiri sama dengan nilai sebelah kanan	x == y
!=	Tidak sama dengan – Hasilnya True jika nilai sebelah kiri tidak sama dengan nilai sebelah kanan	x != y
>=	Lebih besar atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih besar atau sama dengan nilai sebelah kanan	x >= y
<=	Lebih kecil atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih kecil atau sama dengan nilai sebelah kanan	x <= y

c. Operator penugasan

=	Menugaskan nilai yang ada di kanan ke operand di sebelah kiri	**c = a + b** menugaskan a + b ke c
+=	Menambahkan operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	**c += a** sama dengan **c = c + a**
-=	Mengurangi operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	**c -= a** sama dengan **c = c - a**
*=	Mengalikan operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	**c *= a** sama dengan **c = c * a**
/=	Membagi operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	**c /= a** sama dengan **c = c * a**
**=	Memangkatkan operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	**c **= a** sama dengan **c = c ** a**
//=	Melakukan pembagian bulat operand di kanan terhadap operand di kiri dan hasilnya disimpan di operand yang di kiri	**c //= a** sama dengan **c = c // a**
%=	Melakukan operasi sisa bagi operand di	**c %= a** sama dengan

d. Operator logika

Operator	Penjelasan	Contoh
and	Hasilnya adalah True jika kedua operandnya bernilai benar	x and y
or	Hasilnya adalah True jika salah satu atau kedua operandnya bernilai benar	x or y
not	Hasilnya adalah True jika operandnya bernilai salah (kebalikan nilai)	not x

- Percabangan
    a. IF
    Percabangan IF hanya melihat dari satu kondisi saja.
    Contoh codingan:

```
angka = int (input("Masukkan angka: "))     "angk

if (angka > 0):     "angka": Unknown word.
    print("Bilangan tersebut adalah positif")
```

    b. IF-ELSE
    Percabangan IF-ELSE melihat dari dua kondisi
    Contoh codingan:

```
angka = int (input("Masukkan angka: "))     "ang

if (angka > 0):     "angka": Unknown word.
    print("Bilangan tersebut adalah positif")
elif (angka < 0):     "angka": Unknown word.
    print("Bilangan tersebut adalah negatif")
```

    c. IF-ELIF-ELSE
    Percabangan IF-ELIF-ELSE melihat dari 3 kondisi
    Contoh codingan:

```
angka = int (input("Masukkan angka: "))     "an

if (angka > 0):     "angka": Unknown word.
    print("Bilangan tersebut adalah positif")
elif (angka < 0):     "angka": Unknown word.
    print("Bilangan tersebut adalah negatif")
else:
    print("Bilangan tersebut adalah 0")     "Bi
```

- Perulangan
    a. For
    Pada perulangan for biasa digunakkan untuk iterasi pada urutan berupa
    list, tuple, atau string.
    Contoh codingan:

```
huruf = ["a", "b", "c", "d", "e"]

for i in huruf:     "huruf": Unknow
    print (i)
```

    b. While
    Dengan menggunakkan while maka dapat dilakukan perulangan selama kondisi tertentu terpenuhi.
    Contoh codingan:

```
i = 1
jumlah_genap = 0    "jumlah": Unknown word.

while i <= 10:
    if i % 2 == 0:
        jumlah_genap += 1    "jumlah": Unknown word.
    i += 1

print("Jumlah bilangan genap dari 1 sampai 10 adalah:", jumlah_genap)
```

6. Class dan Object
    - Class
      Kelas seperti cetak biru untuk membuat objek. Kelas berisi semua informasi yang diperlukan untuk membuat objek, termasuk atribut dan metode. Anda dapat membuat banyak objek dari satu kelas, dan kelas itu sendiri tidak dapat digunakan secara langsung - harus diimplementasikan ke dalam objek. Misalnya, kelas Mobil, Manusia, dan Kucing adalah contoh kelas.
      Contoh codingan:
    - Atribut
      Kelas adalah jenis variabel khusus yang mengacu pada sekumpulan objek terkait. Setiap objek yang dibuat dari sebuah kelas akan memiliki kumpulan atribut kelas yang sama, yang merupakan properti dari kelas itu sendiri. Ada juga atribut objek individu, yang berbeda untuk setiap objek di kelas.
    - Method
      Setiap kelas memiliki beberapa fitur dasar (seperti atribut) dan juga menyediakan fungsi khusus (seperti metode) yang dapat digunakan oleh objek yang dibuat dari kelas tersebut. Fungsi-fungsi ini seperti tindakan atau tugas yang dapat dilakukan oleh suatu objek.

      Contoh codingan:

```python
class Manusia:
    # atribut
    umur = 0
    jenis_kelamin = ""

    # method
    def berbicara(self):
        print("Halo, apa kabar?")

    def berjalan(self, jarak):
        print("Saya telah berjalan sejauh", jarak, "meter.")

# membuat objek manusia1
manusia1 = Manusia()

# mengakses atribut dan method objek manusia1
manusia1.umur = 25
manusia1.jenis_kelamin = "Laki-laki"
print("Umur manusia1:", manusia1.umur)
print("Jenis kelamin manusia1:", manusia1.jenis_kelamin)
manusia1.berbicara()
manusia1.berjalan(10)
```

- Objek
  Objek adalah representasi dari kelas. Untuk membuat objek yang mewakili kelas, Anda dapat menggunakan nama kelas yang diinginkan diikuti dengan tanda kurung.
- Magic Method
  Metode ajaib adalah metode yang dipanggil secara otomatis oleh sistem ketika melakukan hal-hal seperti menggunakan operator add (__add__), membuat objek (__init__), dan lain-lain.
- Konstruktor
  Konstruktor adalah metode khusus yang dipanggil secara otomatis saat Anda membuat objek dari kelas tertentu. Metode ini dapat melakukan hal-hal seperti mencetak informasi. Selain itu, Anda dapat meneruskan argumen ke konstruktor saat membuat objek.
  Contoh codingan:

```python
class Mahasiswa:
    def __init__(self, Nama, NIM, Kelas_PBO, Jumlah_SKS):
        self.nama = Nama
        self.NIM = NIM
        self.Kelas_PBO = Kelas_PBO
        self.Jumlah_SKS = Jumlah_SKS

    def cetakdata(self):
        print("Data mahasiswa: ")
        print()
        print("Nama:", self.nama)
        print()
        print("NIM:", self.NIM)
        print()
        print("Kelas PBO:", self.Kelas_PBO)
        print()
        print("Jumlah SKS yang diambil:", self.Jumlah_SKS)

Data_Mahasiswa = Mahasiswa("Fathurrahman Al Hafid", 121140088, "RB", 24)

Data_Mahasiswa.cetakdata()
```

- Destruktor

Destruktor adalah fungsi yang dipanggil ketika user menghapus objek. Fungsi ini
bekerja secara otomatis, jadi tidak perlu dilakukan pemanggilan. Tujuan adanya
destruktor adalah melakukan final cleaning up atau “bersih-bersih” terakhir sebelum sebuah objek benar-benar dihapus dari memori.

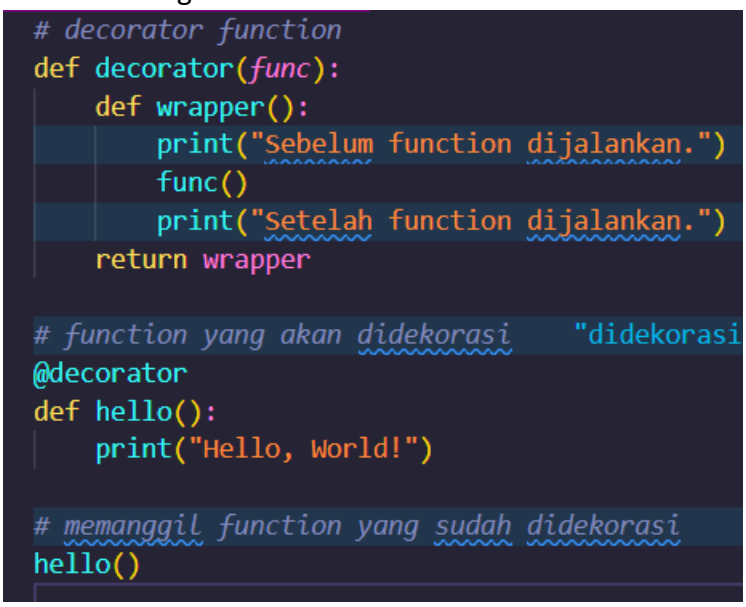
- Setter dan getter
  Setter dan getter digunakan untuk melakukan enkapsulasi agar tidak terjadi perubahan data secara tidak sengaja. Setter adalah method yang digunakan untuk
  menetapkan nilai suatu atribut khususnya atribut private dan protected. sedangkan getter digunakan untuk mengambil nilai.
  Contoh codingan:

```python
class PersegiPanjang:
    def __init__(self, panjang, lebar):
        self._panjang = panjang
        self._lebar = lebar

    # getter panjang
    @property
    def panjang(self):
        return self._panjang

    # setter panjang
    @panjang.setter
    def panjang(self, panjang):
        if panjang <= 0:
            print("Panjang harus lebih besar dari 0.")
        else:
            self._panjang = panjang

    # getter lebar
    @property
    def lebar(self):
        return self._lebar

    # setter lebar
    @lebar.setter
    def lebar(self, lebar):
        if lebar <= 0:
            print("Lebar harus lebih besar dari 0.")
        else:
            self._lebar = lebar

    def luas(self):
        return self._panjang * self._lebar
```

- Decorator

  Selain menggunakan fungsi setter dan getter tambahan seperti pada contoh sebelumnya, dalam Python kita juga dapat memanfaatkan property decorator untuk mendapatkan hasil serupa. Bedanya, melalui property decorator ini kita tidak perlu membuat fungsi lagi dengan nama yang berbeda-beda (cukup menggunakan 1 buah nama/variabel).

  Contoh codingan:



```
# decorator function
def decorator(func):
    def wrapper():
        print("Sebelum function dijalankan.")
        func()
        print("Setelah function dijalankan.")
    return wrapper

# function yang akan didekorasi    "didekorasi
@decorator
def hello():
    print("Hello, World!")

# memanggil function yang sudah didekorasi
hello()
```