TUGAS 5

Nama : David Gunawan
NIM : 121140062
Kelas : PBO RB

Pengenalan Bahasa Pemrograman Python

Python adalah bahasa pemrograman tingkat tinggi yang sangat populer karena sintaks yang mudah dipahami, mudah dibaca dan ditulis, serta fleksibilitas yang tinggi. Bahasa pemrograman ini didesain pada tahun 1991 oleh Guido van Rossum dan mendapat nama dari acara televisi Monty Python.

Python bersifat interpreted, artinya kode Python tidak dijalankan langsung oleh komputer, melainkan melalui program interpreter yang menerjemahkan kode Python menjadi bahasa mesin. Hal ini membuat pengembangan program Python menjadi lebih cepat dan mudah karena tidak memerlukan proses kompilasi seperti pada bahasa pemrograman lain seperti C dan C++.

Sintaks dasar Python

- Statement

  Statement pada Python adalah sebuah perintah yang dieksekusi oleh interpreter Python untuk melakukan tindakan tertentu. akhir dari sebuah statement adalah baris baru (newline) tapi dimungkinkan membuat statement yang terdiri dari beberapa baris menggunakan backslash (\).

- Baris dan Indentasi

  Baris pada Python adalah satu baris teks atau perintah yang dieksekusi oleh interpreter Python. Setiap baris pada Python biasanya diakhiri dengan tanda enter (baris baru).

  Indentasi pada Python adalah penggunaan spasi atau tab untuk menandakan blok kode yang terpisah dari blok kode yang lain. Dalam Python, indentasi digunakan untuk menandakan blok kode pada perulangan, fungsi, percabangan, dan sebagainya. Jumlah spasi atau tab yang digunakan harus konsisten di setiap blok kode yang sama, misalnya satu blok kode perulangan harus memiliki indentasi yang sama di setiap iterasi.

Variabel dan Tipe data Primitif

Variabel pada Python adalah sebuah nama yang digunakan untuk menyimpan nilai. Variabel pada Python tidak perlu dideklarasikan tipe datanya secara eksplisit, karena tipe data dari sebuah variabel akan ditentukan secara otomatis oleh interpreter Python berdasarkan nilai yang diberikan pada variabel tersebut.

Tipe data primitif pada Python adalah tipe data yang tidak dapat dipecahkan lagi menjadi tipe data yang lebih sederhana. Tipe data primitif pada Python meliputi:

- Integer (int): tipe data untuk bilangan bulat, contohnya: 1, 2, -3, 0, dsb.
- Float: tipe data untuk bilangan desimal, contohnya: 3.14, -0.5, 1.0, dsb.
- Boolean (bool): tipe data untuk nilai kebenaran, yaitu True atau False.
- String (str): tipe data untuk teks atau karakter, contohnya: "Halo", "123", "Python", dsb.

Operator

1. Operator Aritmatika

digunakan untuk melakukan operasi matematika, seperti penjumlahan, pengurangan, perkalian, pembagian, dan sebagainya. Beberapa contoh operator aritmatika pada Python antara lain:

- + (penjumlahan)
- - (pengurangan)
- * (perkalian)
- / (pembagian)
- % (modulus atau sisa bagi)
- ** (pangkat)
- // (pembagian bulat)

2. Operator perbandingan

digunakan untuk membandingkan dua nilai atau variabel dan menghasilkan nilai boolean True atau False. Beberapa contoh operator pembanding pada Python antara lain:

- == (sama dengan)
- != (tidak sama dengan)
- > (lebih besar dari)
- < (kurang dari)
- >= (lebih besar dari atau sama dengan)
- <= (kurang dari atau sama dengan)

3. Operator penugasan

digunakan untuk mengisi nilai pada sebuah variabel. Beberapa contoh operator assignment pada Python antara lain:

- = (mengisi nilai)
- += (menambahkan nilai ke variabel dan mengisinya kembali)
- -= (mengurangi nilai dari variabel dan mengisinya kembali)
- *= (mengalikan nilai ke variabel dan mengisinya kembali)
- /= (membagi nilai dari variabel dan mengisinya kembali)

4. Operator Logika

digunakan untuk mengkombinasikan dua nilai atau variabel boolean dan menghasilkan nilai boolean True atau False. Beberapa contoh operator logika pada Python antara lain:

- ‘and’ (kondisi terpenuhi jika kedua nilai boolean adalah True)
- ‘or’ (kondisi terpenuhi jika salah satu nilai boolean adalah True)
- ‘not’ (membalik nilai boolean dari True menjadi False atau sebaliknya)

5. Operator Bitwise

digunakan untuk melakukan operasi bit pada bilangan bulat. Beberapa contoh operator bitwise pada Python antara lain:

- & (bitwise AND)
- | (bitwise OR)
- ^ (bitwise XOR)
- ~ (bitwise NOT)
- << (bitwise shift left)
- >> (bitwise shift right)

6. Operator Identitas

digunakan untuk membandingkan identitas objek pada Python. Beberapa contoh operator identitas pada Python antara lain:

- is (True jika dua objek memiliki identitas yang sama)
- is not (True jika dua objek tidak memiliki identitas yang sama)

7. Operator keanggotaan

digunakan untuk memeriksa apakah sebuah nilai atau variabel terdapat pada sebuah sequence atau container pada Python. Beberapa contoh operator keanggotaan pada Python antara lain:

- in (True jika nilai terdapat dalam sequence atau container)
- not in (True jika nilai tidak terdapat dalam sequence atau container)

Tipe Data Bentukan

Tipe data bentukan pada Python adalah tipe data yang terdiri dari beberapa nilai atau elemen yang disimpan sebagai satu kesatuan yang sama. Berikut adalah beberapa tipe data bentukan pada Python:

1. List

   List adalah tipe data bentukan yang terdiri dari beberapa nilai atau elemen yang ditempatkan dalam tanda kurung siku [ ] dan dipisahkan oleh koma. Setiap elemen pada list dapat memiliki tipe data yang berbeda-beda. List bersifat mutable, artinya elemen pada list dapat diubah.

2. Tuple

   Tuple adalah tipe data bentukan yang terdiri dari beberapa nilai atau elemen yang ditempatkan dalam tanda kurung ( ) dan dipisahkan oleh koma. Setiap elemen pada tuple dapat memiliki tipe data yang berbeda-beda. Tuple bersifat immutable, artinya elemen pada tuple tidak dapat diubah setelah didefinisikan.

3. Set

   Set adalah tipe data bentukan yang terdiri dari beberapa nilai atau elemen yang ditempatkan dalam tanda kurung kurawal { } dan dipisahkan oleh koma. Set tidak memperdulikan urutan elemen dan tidak memperbolehkan adanya elemen yang duplikat. Set bersifat mutable, artinya elemen pada set dapat diubah.

4. Dictionary

   Dictionary adalah tipe data bentukan yang terdiri dari beberapa pasangan key-value yang ditempatkan dalam tanda kurung kurawal { } dan dipisahkan oleh koma. Setiap key harus unik dan harus diikuti oleh tanda titik dua (:) dan diikuti dengan value. Value pada dictionary dapat memiliki tipe data yang berbeda-beda. Dictionary bersifat mutable, artinya pasangan key-value pada dictionary dapat diubah.

Percabangan

Percabangan memungkinkan program untuk mengambil keputusan berdasarkan kondisi tertentu dan melakukan tindakan yang berbeda tergantung pada apakah kondisi tersebut benar atau salah. Contoh beberapa percabangan pada python yaitu :

1. Percabangan if

   Contoh :

```
angka = 10
if angka<10:
    print("<10")
```

2. Percabangan if-else
   Contoh :

```
angka = 10
if angka<10:
    print("<10")
else:
    print(">10")
```

3. Percabangan if-elif-else

```
angka = 10
if angka<0:
    print("<0")
elif angka>0:
    print(">0")
elif angka==0:
    print("0")
else:
    print("Bukan angka")
```

Perulangan

Perulangan atau loop adalah salah satu fitur penting dalam pemrograman, termasuk dalam bahasa pemrograman Python. Perulangan memungkinkan program untuk mengeksekusi suatu blok kode secara berulang-ulang selama kondisi tertentu terpenuhi. Ada dua jenis perulangan di Python, yaitu perulangan for dan perulangan while.

1. Perulangan for
   Perulangan for pada Python digunakan untuk mengulangi suatu blok kode sebanyak jumlah elemen dalam sebuah iterable (seperti list, tuple, set, dan dictionary). Contoh :

```
for i in range (10):
    print (i)
```

2. Perulangan while
   Perulangan while pada Python digunakan untuk mengulangi suatu blok kode selama kondisi tertentu terpenuhi. Contoh :

```
i = 0
while i<10:
    print(i)
    i+=1
```

Fungsi

Fungsi adalah blok kode yang dapat dipanggil atau dijalankan kembali berulang kali dalam sebuah program. Fungsi pada Python dapat membantu untuk memecah program menjadi bagian-bagian yang lebih kecil dan terorganisir sehingga lebih mudah untuk dibaca, dimengerti, dan dikelola. Contoh :

```
def tambah(a,b):
    return a+b
```

Kelas

Kelas atau Class pada Python adalah struktur dasar untuk membuat objek. Class dapat dianggap sebagai blueprint atau cetak biru untuk membuat objek dengan karakteristik tertentu. Dalam sebuah class, terdapat atribut dan method yang menentukan perilaku dari objek yang dibuat dari class tersebut. Class terdiri atas atribut dan method.

1. Atribut

   Atribut adalah variabel yang didefinisikan dalam sebuah class dan digunakan untuk menyimpan data atau informasi yang berhubungan dengan objek yang dibuat dari class tersebut. Atribut dapat diakses menggunakan objek yang dibuat dari class tersebut.

   Contoh :

```
class angka():
    def __init__(self,angka):
        self.nomor = angka
```

   nomor merupakan sebuah atribut pada class tersebut

2. Method

   Method adalah fungsi yang didefinisikan dalam sebuah class dan digunakan untuk memanipulasi atau memproses data yang berhubungan dengan objek yang dibuat dari class tersebut. Method dapat diakses menggunakan objek yang dibuat dari class tersebut.

   Contoh :

```
class angka():
    def __init__(self,angka):
        self.nomor = angka

    def tambahangka(self):
        self.nomor+=1
```

   tambahangka() merupakan sebuah method pada class tersebut

Objek

Objek pada Python adalah sebuah instansi dari sebuah class. Objek terdiri dari data (atribut) dan kode (method) yang beroperasi pada data tersebut. Dalam hal ini, objek adalah representasi dari sebuah entity yang memiliki sifat-sifat dan perilaku tertentu.

Dalam Python, objek dapat dibuat dengan cara menginstansiasi sebuah class menggunakan sintaksis nama_objek = nama_class() yang akan membuat sebuah objek dari class tersebut. Setiap objek yang dihasilkan dari sebuah class memiliki karakteristik yang sama, namun nilai-nilai dari atribut dapat berbeda antar objek, tergantung pada inisialisasi saat objek dibuat.

Contoh :

```
mahasiswa = orang()
```

Magic Method

Magic method pada Python adalah method-method khusus yang dimulai dan diakhiri dengan double underscore (__). Magic method dapat digunakan untuk melakukan aksi tertentu pada objek seperti operasi aritmatika, pemanggilan objek, dan manipulasi objek. Magic method juga dikenal sebagai dunder methods (double underscore methods).

Beberapa contoh dari magic method pada Python:

- __init__() : magic method yang dipanggil saat objek dibuat dan digunakan untuk inisialisasi atribut objek.
- __str__() : magic method yang mengembalikan string representasi dari sebuah objek ketika objek tersebut diubah menjadi string.
- __len__() : magic method yang mengembalikan panjang dari sebuah objek seperti list, tuple, atau string.
- __add__() : magic method yang digunakan untuk melakukan operasi penjumlahan objek.
- __eq__() : magic method yang digunakan untuk membandingkan kesamaan dua objek.

Konstruktor

Konstruktor adalah method khusus yang dipanggil secara otomatis ketika sebuah objek dibuat. Nama konstruktor pada Python selalu sama, yaitu __init__(). Konstruktor biasanya digunakan untuk melakukan inisialisasi atribut atau variabel pada objek yang baru dibuat. Konstruktor dapat menerima parameter yang digunakan untuk menginisialisasi atribut objek.

Destruktor
Destruktor adalah method khusus lainnya pada kelas dalam bahasa pemrograman Python. Nama destruktor pada Python selalu sama, yaitu __del__(). Destruktor dipanggil secara otomatis ketika sebuah objek dihapus dari memori atau saat program berakhir. Destruktor biasanya digunakan untuk membersihkan sumber daya atau melakukan tindakan terakhir pada objek sebelum dihapus dari memori.

Setter dan Getter
Getter adalah method yang digunakan untuk mengambil nilai atau membaca nilai dari suatu variabel atau atribut dalam objek. Getter umumnya memiliki nama yang sama dengan variabel atau atribut yang akan dibaca nilainya, namun diawali dengan prefix 'get_.'
Sedangkan Setter adalah method yang digunakan untuk mengubah nilai dari suatu variabel atau atribut dalam objek. Setter umumnya memiliki nama yang sama dengan variabel atau atribut yang akan diubah nilainya, namun diawali dengan prefix 'set_.'

Decorator
Decorator pada Python adalah sebuah fitur yang memungkinkan kita untuk mengubah fungsi atau method pada saat runtime. Decorator dapat digunakan untuk menambahkan fungsionalitas atau perilaku tambahan pada sebuah fungsi atau method tanpa perlu mengubah kode asli dari fungsi atau method tersebut.

Decorator pada Python didefinisikan dengan menggunakan simbol @ diikuti dengan nama fungsi decorator yang akan digunakan. Fungsi decorator adalah fungsi yang menerima sebuah fungsi atau method sebagai argumen, kemudian mengembalikan sebuah fungsi baru dengan perilaku yang diubah.

Sifat sifat pada PBO

Terdapat 4 sifat dasar pada pemrograman berorientasi objek yaitu

1. Abstraksi

Abstraksi dalam pemrograman berorientasi objek (PBO) adalah proses menyembunyikan detail implementasi objek dan hanya mengekspos fungsionalitas penting yang relevan untuk pengguna. Hal ini dilakukan untuk mempermudah penggunaan objek dan meningkatkan keamanan dan fleksibilitas program.

Dalam PBO, abstraksi dapat dilakukan dengan membuat class abstract atau interface yang hanya mendefinisikan method tanpa memberikan implementasi pada method tersebut. Objek dari class abstract atau interface ini tidak dapat dibuat, namun dapat digunakan oleh class lain yang mewarisi atau mengimplementasikan method yang didefinisikan.

Contoh :

```
from abc import ABC,
abstractmethod
class hewan(ABC):
    @abstractmethod
    def bersuara():
        pass

class anjing(hewan):
    def bersuara(self):
        print("GUK GUK")
```

```
class kucing(hewan):
    pass

anjing = anjing()
kucing = kucing()

anjing.bersuara()
kucing.bersuara()
```

Pada Contoh diatas, hewan adalah class abstract yang mendefinisikan method bersuara() tanpa memberikan implementasi. Class kucing dan anjing mewarisi class hewan dan dapat mengimplementasikan method bersuara() sesuai dengan fungsi classnya. Pada class kucing, dapat dilihat bahwa fungsi bersuara tidak didefinisikan ulang. Hal tersebut menyebabkan error karna method abstract harus didefinisikan ulang pada class turunannya.

2. Enkapsulasi

Enkapsulasi merupakan sebuah konsep yang bertujuan untuk menyembunyikan detail implementasi objek dan hanya mengekspos antarmuka publik yang dapat diakses oleh pengguna. Hal ini dilakukan untuk membatasi akses langsung ke data dan method dari objek dan mengontrol bagaimana data dan method dapat diakses dan dimodifikasi.

enkapsulasi dapat dicapai dengan menggunakan access modifier seperti public, private, dan protected pada data dan method dalam sebuah class. Access modifier ini menentukan apakah suatu data atau method dapat diakses dan dimodifikasi oleh pengguna atau hanya dapat diakses dan dimodifikasi oleh class itu sendiri.

- Public: Data atau method yang ditandai dengan access modifier public dapat diakses dan dimodifikasi dari luar class.
- Private: Data atau method yang ditandai dengan access modifier private hanya dapat diakses dan dimodifikasi dari dalam class tersebut.
- Protected: Data atau method yang ditandai dengan access modifier protected hanya dapat diakses dan dimodifikasi oleh class tersebut dan class yang mewarisi class tersebut.

Contoh :

```
class anjing():
    def __init__(self, umur):
        self.__hewan = type(self).__name__
        self.umur = umur

    def __gantiumur(self,umurbaru):
        self.umur = umurbaru

    def bungkus(self,umurbaru):
        self.__gantiumur(umurbaru)

    def getinfo(self):
        print(f"Jenis : {self.__hewan}, umur : {self.umur}")

anjing = anjing(15)

anjing.getinfo()
#Terjadi error jika method ataupun atribut private diakses langsung
# anjing.__hewan = "Burung"
# anjing.__gantiumur(20)
anjing.bungkus(20)
anjing.getinfo()
```

Pada contoh diatas, class anjing mempunyai 2 atribut yaitu __hewan yang merupakan atribut private dan umur yang merupakan atribut publik. Didalam class anjing terdapat 3 fungsi yaitu __gantiumur() yang merupakan method private, bungkus() yang merupakan method publik, dan getinfo() yang merupakan method publik. Fungsi getinfo() merupakan contoh enkapsulasi karna mengakses atribut private melalui method. Apabila kita mencoba mengakses langsung atribut ataupun method private tersebut diluar class, maka akan terjadi error. Fungsi

bungkus juga merupakan contoh enkapsulasi karna membungkus private method didalam public method agar bisa diakses diluar classnya. Kita hanya perlu memanggil fungsi bungkus(), dan fungsi bungkus() didalam class tersebut akan memanggil fungsi __gantiumur() yang merupakan private method.

Dengan menggunakan enkapsulasi, kita dapat membatasi akses dan modifikasi langsung ke data dan method dalam sebuah class, sehingga mengurangi kemungkinan terjadinya kesalahan dalam penggunaan objek. Hal ini juga dapat meningkatkan keamanan dan fleksibilitas program karena kita dapat mengontrol bagaimana objek dapat diakses dan dimodifikasi.

3. Inheritance

Inheritance atau pewarisan adalah salah satu konsep yang memungkinkan untuk membuat sebuah class baru yang mewarisi sifat atau perilaku dari class yang sudah ada (parent class atau superclass). Class baru yang dihasilkan disebut dengan child class atau subclass.

Dalam inheritance, subclass dapat mengakses semua data dan method dari superclass, termasuk constructor dan destructor, sehingga subclass dapat menambahkan atau memodifikasi perilaku dari superclass tanpa harus menuliskan ulang seluruh kode dari superclass. Dalam hal ini, subclass dapat menambahkan atribut atau method baru, dan memodifikasi atau menimpa method yang sudah ada pada superclass.

Contoh :

```
class orang():
    def __init__(self, nama, umur):
        self.nama = nama
        self.umur = umur

    def lihatnama(self):
        print(f"Nama : {self.nama}")

    def lihatinfo(self):
        print (f"Nama : {self.nama}, umur : {self.umur}")

class mahasiswa(orang):
    def __init__(self, nama, umur,nim):
        super().__init__(nama, umur)
        self.nim = nim

    def lihatinfo(self):
        print(f"Nama : {self.nama}, umur : {self.umur}, nim : {self.nim}")
```

```
mhs = mahasiswa("David",20,121140062)

mhs.lihatnama()
mhs.lihatinfo()
```

Pada kode diatas, terdapat dua class yaitu class orang yang merupakan superclass atau class parent dan class mahasiswa yang merupakan subclass atau class child. Semua atribut ataupun method yang ada pada class orang diturunkan semua ke class mahasiswa, sehingga class mahasiswa dapat mempunyai atribut dan method yang sama seperti class orang yang merupakan superclassnya. Method turunan pada child dapat menimpa method pada class parentnya dengan cara mendeklarasikan ulang method tersebut pada class turunannya, contohnya pada fungsi lihatinfo().

4. Polymorphism

Polymorphism (banyak bentuk) adalah konsep dalam pemrograman berorientasi objek yang memungkinkan untuk membuat banyak bentuk dari sebuah objek atau method yang sama, tetapi berperilaku berbeda tergantung pada konteksnya. Dengan kata lain, satu objek atau method dapat memiliki banyak bentuk yang berbeda dalam situasi yang berbeda.

Dalam pemrograman berorientasi objek, polymorphism dapat diimplementasikan dengan dua cara, yaitu overriding dan overloading.

- Overriding

  Overriding terjadi ketika sebuah kelas turunan (subclass) memiliki metode dengan nama yang sama seperti metode pada kelas induk (superclass). Saat metode tersebut dipanggil pada objek turunan, metode yang terdapat pada kelas turunan akan menimpa atau menggantikan metode pada kelas induk. Dengan kata lain, perilaku metode pada kelas turunan akan berbeda dengan perilaku metode pada kelas induk.

  Contoh :

```
class hewan():
    def suara():
        pass

class anjing(hewan):
    def suara(self):
        print("GUK GUK")

class kucing(hewan):
    def suara(self):
        print("MEOW MEOW")

anjing = anjing()
kucing = kucing()
```

```
anjing.suara()
kucing.suara()
```

Pada kode ditatas, hewan merupakan class induk sedangkan anjing dan kucing merupakan class turunan dari hewan. Class kucing dan hewan menimpa method suara() pada class induknya dan disesuaikan dengan kebutuhan dari classnya.

- Overloading

overloading terjadi ketika sebuah kelas memiliki dua atau lebih metode dengan nama yang sama tetapi dengan parameter yang berbeda. Saat metode tersebut dipanggil dengan parameter yang sesuai, metode yang sesuai akan dieksekusi

Contoh :

```
class hitung():
    def penjumlahan(self,x1,x2):
        return x1+x2

    def penjumlahan(self,x1,x2,x3):
        return x1+x2+x3

    def penjumlahan(self,x1,x2,x3,x4):
        return x1+x2+x3+x4
```

Pada contoh diatas, class hitung mempunyai 3 metode dengan nama yang sama yaitu penjumlahan(). Ketiganya mempunyai jumlah parameter yang berbeda. Jika kita memanggil metode penjumlahan dengan 2 parameter, maka yang terpanggil adalah metode yang mempunyai jumlah parameter 2. Overloading pada python tidak bekerja secara langsung dikarenakan python tidak memeriksa jumlah parameter atau tipe data parameter pada saat kompilasi atau runtime. Overloading bekerja secara baik pada bahasa pemrograman seperti Java atau C++.