RESUME PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN 1 – 4

Oleh :

Muhammad Shahih Indra Sakti - 121140184

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2023

Daftar Isi

Pertemuan 1

1. Pengenalan Bahasa Pemrograman Python
   Python adalah bahasa pemrograman tingkat tinggi yang sering digunakan untuk pengembangan web, ilmu data, kecerdasan buatan, dan aplikasi desktop. Python dikembangkan pada awal 1990-an oleh Guido van Rossum dan memiliki sintaks yang mudah dibaca dan dipahami.

   Bahasa pemrograman python didukung oleh library(modul) yang berlimpah dan Python bisa dipakai untuk pemrograman desktop maupun mobile, CLI, GUI, web, otomatisasi, hacking, IoT,robotika, dan lain sebagainya.

2. Dasar Pemrograman Python
   Bahasa pemrograman pyhon memiliki beberapa sintaks :

   1. Statement
      Statement adalah Semua perintah yang bisa dieksekusi Python.

   2. Baris dan Indentasi
      Baris pada Python menunjujk pada setiap instruksi atau pernyataan kode yang ditulis dalam satu baris. Setiap baris biasanya berisi satu pernyataan kode dan diakhiri dengan karakter baris baru. Identasi pada Python menunjuk pada spasi atau tabulasi yang digunakan untuk menunjukkan blok kode.
      Contoh :

```
def hello():
    print("Hello, World!")
```

   3. Variabel dan Tipe Data Primitif
      Variabel pada Python adalah lokasi memori yang digunakan untuk menyimpan nilai. Setiap variabel memiliki nama dan tipe data tertentu, dan dapat diisi dengan nilai yang sesuai dengan tipe datanya. Tipe data primitif pada Python adalah tipe data bawaan yang tersedia di dalam bahasa Python.
      Contoh :

```
bil1=true #boolean
var1=10 #integer
des=6.54 #float
hur= "contoh kata" #string
```

4. Operator

   Operator pada python antara lain :

   a. Operator Aritmatika

      Operator aritmatika merupakan operator yang digunakan untuk melakukan operasi matematika, seperti penjumlahan, pengurangan, perkalian, pembagian.

      Contoh :

```
bil1=int(input("bil pertama: "))
bil2=int(input("bil kedua: "))
hasil=bil1+bil2
```

   b. Operator Perbandingan

      Operator perbandingan merupakan operator yang digunakan untuk membandingkan 2 buah nilai. Hasil perbandingannya adalah True atau False tergantung kondisi.

      Contoh :

```
a=9
b=4
print(a>b) # true karena kondisi benar
print(a<b) # false karena kondisi salah
```

   c. Operator Penugasan

      Operator penugasan merupakan operator yang digunakan untuk memberi nilai ke variabel.

      Contoh :

```
x = 10
x += 5 # setara dengan x = x + 5
print(x) # output: 15
```

```
x = 10
x -= 3 # setara dengan x = x - 3
print(x) # output: 7
```

d. Operator Logika

Operator logika merupakan operator yang digunakan untuk melakukan operasi logika.

Contoh :

```
x = 5
print(x > 3 and x < 10) # output: True

y = 15
print(y > 20 and y < 10) # output: False
```

e. Operator Bitwise

Operator bitwise merupakan operator yang melakukan operasi bit terhadap operand.

Contoh :

```
a = 10   # 1010 dalam biner
b = 6    # 0110 dalam biner

c = a & b  # 0010 dalam biner
print(c)   # Output: 2
```

f. Operator Identitas

Operator identitas merupakan operator yang memeriksa apakah dua buah nilai atau variabel berada pada lokasi memori yang sama.

Contoh :

```
a = [1, 2, 3]
b = a
c = [1, 2, 3]

print(a is b)   # Output: True
print(a is c)   # Output: False
```

g. Operator Keanggotaan

Operator keanggotaan merupakan operator yang digunakan untuk memeriksa apakah suatu nilai atau variabel merupakan anggota atau ditemukan di dalam suatu data.

Contoh :

```
a = [1, 2, 3]

print(2 in a)   # Output: True
print(4 in a)   # Output: False
```

5. Tipe Data Bentukan

   Ada 4 tipe data bentukan yaitu list, tiple, set dan dictionary.

   a. List merupakan tipe data struktur yang digunakan untuk menyimpan sekumpulan nilai dalam satu variabel. List didefinisikan dengan kurung siku dan nilai-nilainya dipisahkan dengan koma.

      Contoh:

```
my_list = [1, 2, 3, "Hello", "World"]
```

   b. Tuple merupakan tipe data struktur yang mirip dengan list, namun nilai-nilainya bersifat tidak dapat diubah setelah didefinisikan. Tuple didefinisikan dengan kurung biasa dan nilai-nilainya dipisahkan dengan koma.

      Contoh:

```
my_tuple = (1, 2, 3, "Hello", "World")
```

   c. Set merupakan tipe data struktur yang digunakan untuk menyimpan kumpulan nilai yang tidak memiliki urutan tertentu dan tidak diizinkan duplikat. Set didefinisikan dengan kurung kurawal dan nilai-nilainya dipisahkan dengan koma.

      Contoh:

```
my_set = {1, 2, 3, "Hello", "World"}
```

   d. Dictionary merupakan tipe data struktur yang digunakan untuk menyimpan pasangan key-value dalam satu variabel. Key pada dictionary bersifat unik dan tidak diizinkan duplikat. Dictionary didefinisikan dengan kurung kurawal dan pasangan key-value dipisahkan dengan tanda titik dua.

      Contoh:

```
my_dict = {"name": "John", "age": 30, "gender": "male"}
```

6. Percabangan

   Ada 4 jenis percabangan yaitu Percabangan IF, Percabangan IF-ELSE, Percabangan IF-ELSE-IF dan Nested IF.

   a. Percabangan IF merupakan percabangan yang apabila kondisi pertama terpenuhi maka akan selesai.

      Contoh :

```
x = 10

if x > 5:
    print("x lebih besar dari 5")
```

b. Percabangan IF-ELSE merupakan percabangan yang digunakan untuk mengeksekusi suatu blok kode jika kondisi yang diberikan bernilai benar, dan mengeksekusi blok kode lain jika kondisi tersebut bernilai salah.
Contoh :

```
nilai = 65

if nilai >= 70:
    print("Selamat, kamu lulus!")
else:
    print("Maaf, kamu tidak lulus")
```

c. Percabangan IF-ELSE-IF merupakan percabangan yang digunakan untuk mengeksekusi beberapa blok kode jika kondisi yang diberikan memiliki beberapa kemungkinan nilai.
Contoh :

```
nilai = 65

if nilai >= 80:
    print("Nilai kamu A")
elif nilai >= 70:
    print("Nilai kamu B")
elif nilai >= 60:
    print("Nilai kamu C")
else:
    print("Maaf, kamu tidak lulus")
```

d. Nested IF merupakan percabangan yang digunakan ketika kita ingin mengeksekusi percabangan IF di dalam blok kode IF lainnya.
Contoh :

```
nilai = 75
usia = 20

if nilai >= 70:
    if usia >= 18:
        print("Selamat, kamu lulus dan sudah cukup umur")
    else:
        print("Selamat, kamu lulus, tapi kamu belum cukup umur")
else:
    print("Maaf, kamu tidak lulus")
```

7. Perulangan

   Dalam python terdapat dua perulangan yaitu Perulangan For dan Perulangan While.

   a. Perulangan for merupakan perulanagn yang digunakan untuk melakukan perulangan sejumlah kali berdasarkan jumlah elemen yang ada pada sebuah objek.

      Contoh :

```
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

   b. Perulangan while merupakan perulanagn yang digunakan untuk melakukan perulangan selama kondisi yang diberikan bernilai benar.

      Contoh :

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

8. Fungsi

   Fungsi pada Python adalah blok kode terorganisir yang dapat digunakan untuk melakukan tugas tertentu. Fungsi pada Python didefinisikan dengan menggunakan kata kunci "def", diikuti dengan nama fungsi, dan diakhiri dengan tanda kurung dan titik dua.

   Contoh :

```python
# contoh fungsi sederhana
def greet():
    print("Hello, welcome!")

# memanggil fungsi greet
greet()

# contoh fungsi dengan parameter
def multiply(x, y):
    return x * y

# memanggil fungsi multiply dan menyimpan hasilnya ke dalam variabel result
result = multiply(3, 4)
print(result)
```

Pertemuan 2

1. Kelas

   Kelas atau class pada python bisa kita katakan sebagai sebuah blueprint (cetakan) dari objek (atau instance) yang ingin kita buat. Dengan menggunakan kelas kita dapat mendesain objek secara bebas. Kelas berisi dan mendefinisikan atribut/properti dan metode untuk objeknya nanti.
   Contoh :

```python
# membuat kelas Person
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print("Hello, my name is " + self.name + " and I am " + str(self.age) + " years old.")

# membuat objek dari kelas Person
person1 = Person("John", 30)

# memanggil metode greet pada objek person1
person1.greet()
```

2. Objek
   Objek merupakan sesuatu yang “mewakili” kelas. Objek berfungsi sebagai pengganti pemanggilan sebuah kelas, maka sebuah objek hanya dapat mewakili sebuah kelas saja.
   Contoh :

```python
# membuat kelas Person
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print("Hello, my name is " + self.name + " and I am " + str(self.age) + " years old.")

# membuat objek dari kelas Person
person1 = Person("John", 30)

# memanggil metode greet pada objek person1
person1.greet()
```

3. Magic Method
   Magic merupakan adalah metode yang diawali dan diakhiri dengan double underscore (dunder). Method ini tidak dipanggil secara langsung, tapi dipanggil sistem secara internal ketika melakukan sesuatu seperti menggunakan operator tambah (__add__), membuat objek (_init_), dan lain-lain.
   Contoh :

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return "Person: " + self.name

    def __eq__(self, other):
        return self.name == other.name and self.age == other.age
```

4. Konstruktor
   Konstruktor merupakan method yang “pasti” dijalankan secara otomatis pada saat sebuah objek dibuat untuk mewakili kelas tersebut. Seperti dengan method lain, konstruktor dapat melakukan operasi seperti melakukan print.
   Contoh :

```python
# membuat kelas Person
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

# membuat objek dari kelas Person
person1 = Person("John", 30)
person2 = Person("Jane", 25)

# mencetak nilai atribut name dan age dari objek person1 dan person2
print(person1.name)
print(person1.age)
print(person2.name)
print(person2.age)
```

5. Destruktor

   Destruktor merupakan fungsi yang dipanggil ketika user menghapus objek. Fungsi ini bekerja secara otomatis, jadi tidak perlu dilakukan pemanggilan.

   Contoh :

```python
# membuat kelas Person
class Person:
    def __init__(self, name):
        self.name = name

    def __del__(self):
        print("Objek telah dihapus")

# membuat objek dari kelas Person
person1 = Person("John")

# menghapus objek person1
del person1
```

6. Setter dan Getter

Setter dan getter digunakan untuk melakukan enkapsulasi agar tidak terjadi perubahan data secara tidak sengaja. Setter merupakan method yang digunakan untuk menetapkan nilai suatu atribut khususnya atribut private dan protected (akan diajarkan lebih jelas pada materi minggu selanjutnya), sedangkan getter digunakan untuk mengambil nilai.

Contoh :

```
# membuat kelas Person
class Person:
    def __init__(self, name):
        self._name = name

    # getter untuk atribut name
    @property
    def name(self):
        return self._name

    # setter untuk atribut name
    @name.setter
    def name(self, name):
        self._name = name

# membuat objek dari kelas Person
person1 = Person("John")

# mengambil nilai atribut name
print(person1.name)

# mengubah nilai atribut name
person1.name = "Jane"

# mencetak nilai atribut name yang telah diubah
print(person1.name)
```

7. Decorator

   Decorator pada Python adalah sebuah fungsi yang digunakan untuk memodifikasi atau menambahkan fungsionalitas pada sebuah fungsi atau class yang sudah ada tanpa mengubah kode asli dari fungsi atau class tersebut. Decorator dapat didefinisikan dengan menggunakan simbol "@" diikuti dengan nama fungsi decorator.

   Contoh :

```python
# fungsi decorator untuk menghitung waktu eksekusi
def measure_time(func):
    def wrapper(*args, **kwargs):
        import time
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"Waktu eksekusi: {end_time - start_time} detik")
        return result
    return wrapper

# fungsi yang akan didekorasi
@measure_time
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

# menjalankan fungsi fibonacci dan menghitung waktu eksekusinya
result = fibonacci(30)
print(result)
```

Pertemuan 3

1. Abstraksi

   Abstraksi pada Python adalah sebuah konsep pemrograman yang digunakan untuk menyembunyikan detail implementasi dari pengguna. Abstraksi memungkinkan kita untuk memfokuskan perhatian pada fungsionalitas suatu program tanpa harus memperhatikan detail implementasi dari setiap komponen program.

   Pada Python, abstraksi dapat diimplementasikan dengan menggunakan class dan method. Class digunakan untuk mewakili sebuah konsep atau objek, sementara method digunakan untuk menerapkan fungsionalitas pada class tersebut.

2. Enkapsulasi

   Enkapsulasi pada Python adalah konsep pemrograman yang mengizinkan kita untuk menyembunyikan atau mengatur akses ke data atau method dalam sebuah class. Enkapsulasi memungkinkan kita untuk menjaga integritas data dan mencegah perubahan yang tidak diinginkan pada data atau method.

   Dalam Python, enkapsulasi dapat dicapai dengan menggunakan akses modifier, yaitu "public", "protected", dan "private". Dalam Python, akses modifier didefinisikan dengan menambahkan underscore pada awal nama variabel atau method.

   Contoh :

```python
class Car:
    def __init__(self, brand, model, color):
        self._brand = brand
        self._model = model
        self._color = color
        self._fuel = 0

    def fill_fuel(self, amount):
        self._fuel += amount

    def drive(self, distance):
        if self._fuel < distance / 10:
            print("Bensin tidak cukup. Silahkan isi bensin terlebih dahulu.")
        else:
            self._fuel -= distance / 10
            print(f"Mobil {self._brand} {self._model} berwarna {self._color} telah menempuh jarak {distance} km.")

    def get_brand(self):
        return self._brand

    def get_model(self):
        return self._model

    def get_color(self):
        return self._color

    def get_fuel(self):
        return self._fuel

car1 = Car("Toyota", "Avanza", "silver")
car1.fill_fuel(20)
car1.drive(100)
print(f"Sisa bensin: {car1.get_fuel()} liter.")
```

3. Objek di Python

   Untuk mengakses sebuah kelas di python dapat menggunakan objek.

   Contoh :

Pertemuan 4

1. Inheritance (Pewarisan)

Dalam bahasa pemrograman Python inheritance merupakan sebuah konsep dimana sebuah class dapat 'mewarisi' atribut dan method dari class yang lain. Dalam hal ini, class yang mewarisi atribut dan method disebut sebagai subclass, sedangkan class yang memberikan atribut dan method disebut sebagai superclass atau parent class.

Contoh :

```
class SuperClass:
    def __init__(self, x):
        self.x = x

    def display(self):
        print("Nilai x dari superclass:", self.x)

class SubClass(SuperClass):
    def __init__(self, x, y):
        super().__init__(x)
        self.y = y

    def display(self):
        super().display()
        print("Nilai y dari subclass:", self.y)

obj1 = SubClass(10, 20)
obj1.display()
```

a. Inheritance Identik
   Inheritance identik merupakan pewarisan yang menambahkan constructor pada class child sehingga class child memiliki constructornya sendiri tanpa menghilangkan constructor pada class parentnya. Inheritance ditandai dengan constructor menggunakan kata kunci super() pada class child.
   Contoh :

```python
class Person:
    def __init__(self, name):
        self.name = name

    def display(self):
        print("Nama:", self.name)

class Employee(Person):
    def __init__(self, name, id):
        Person.__init__(self, name)
        self.id = id

    def display(self):
        Person.display(self)
        print("ID:", self.id)

class Manager(Employee):
    def __init__(self, name, id, department):
        Employee.__init__(self, name, id)
        self.department = department

    def display(self):
        Employee.display(self)
        print("Departemen:", self.department)

obj1 = Manager("John Doe", "1234", "Marketing")
obj1.display()
```

2. Polymorphism
   Polymorphism merupakan sebuah konsep dalam pemrograman Python (dan juga dalam pemrograman berorientasi objek lainnya) dimana suatu objek dapat memiliki berbagai bentuk atau perilaku. Dalam konteks Python, Polymorphism biasanya berkaitan dengan kemampuan suatu objek untuk digunakan dalam berbagai situasi atau dalam konteks yang berbeda.

3. Overriding
   Overriding merupakan sebuah proses mengganti atau menimpa definisi suatu method yang ada pada superclass oleh subclass. Dalam overriding, subclass dapat memberikan definisi ulang (yang berbeda) untuk method yang diwarisi dari superclass.
   Contoh :

```python
class Shape:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def area(self):
        pass

class Rectangle(Shape):
    def __init__(self, x, y, width, height):
        Shape.__init__(self, x, y)
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

class Circle(Shape):
    def __init__(self, x, y, radius):
        Shape.__init__(self, x, y)
        self.radius = radius

    def area(self):
        return 3.14 * (self.radius ** 2)

r = Rectangle(0, 0, 5, 10)
print("Luas Rectangle:", r.area())

c = Circle(0, 0, 5)
print("Luas Circle:", c.area())
```

4. Overloading
   Overloading merupakan sebuah proses memberikan nama yang sama pada beberapa method yang berbeda dalam sebuah class. Dalam overloading, method-method tersebut dapat memiliki jumlah dan jenis parameter yang berbeda, sehingga ketika dipanggil, Python dapat menentukan method mana yang harus digunakan berdasarkan parameter yang diberikan.
   Contoh :

```python
class Calculator:
    def add(self, x, y):
        return x + y

    def add(self, x, y, z):
        return x + y + z

    def add(self, x, y, z, w):
        return x + y + z + w
```

5. Multiple Inheritance

   Dengan Multiple Inheritance, kita dapat membuat class yang lebih kompleks dan fleksibel, tetapi kita perlu berhati-hati dalam merancang hierarki class agar tidak terjadi konflik nama atau kebingungan dalam penggunaan method dan atribut.

   Contoh :

```python
class A:
    def method_a(self):
        print("Ini adalah method dari kelas A")

class B:
    def method_b(self):
        print("Ini adalah method dari kelas B")

class C(A, B):
    def method_c(self):
        print("Ini adalah method dari kelas C")

objek_c = C()
objek_c.method_a()  # Output: Ini adalah method dari kelas A
objek_c.method_b()  # Output: Ini adalah method dari kelas B
objek_c.method_c()  # Output: Ini adalah method dari kelas C
```

6. Casting

   Casting mempunyai beberapa jenis yaitu :

   a. Downcasting

      Downcasting pada Python merupakan sebuah konsep dalam pemrograman berorientasi objek dimana kita mengubah suatu objek dari tipe yang lebih umum (superclass) menjadi tipe yang lebih khusus (subclass).

      Contoh :

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("Hewan ini sedang bersuara")

class Cat(Animal):
    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed

    def speak(self):
        print("Meow")

cat = Cat("Kitty", "Persia")

# Downcasting dari Cat menjadi Animal
animal = Animal("Animal")
animal = cat  # downcasting

# Memanggil method speak dari objek Animal yang sudah di-downcast
animal.speak()  # Output: Meow
```

   b. Upcasting

      Upcasting pada Python merupakan sebuah konsep dalam pemrograman berorientasi objek dimana kita mengubah suatu objek dari tipe yang lebih khusus (subclass) menjadi tipe yang lebih umum (superclass).

      Contoh :

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("Hewan ini sedang bersuara")

class Cat(Animal):
    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed
```

c. Type casting

Type casting pada Python merupakan proses mengubah tipe data dari suatu objek menjadi tipe data yang berbeda. Python mendukung beberapa jenis type casting, termasuk konversi antara tipe data numerik, konversi antara string dan tipe data numerik, dan konversi antara tipe data sequence seperti tuple dan list.