Nama : Andreas Pascalis Tristan

NIM : 121140017

Prodi : Teknik Informatika

RESUME PRAKTIKUM PYTHON

Python

- Syntax Dasar
  - Statement
    Statement adalah semua bentuk perintah yang dapat dieksekusi dalam program pyhton. Akhir statement ditandai dengan baris baru/ new line atau bisa menggunakan garis backlash(\).

```
variable = 'perintah'  # ini statement

ini_angka = 10 + \
    20 + \
        30
print(ini_angka)
```

- Baris dan Indentasi
  Dalam penulisan kodenya, python menggunakan indentasi atau tab dan spasi. Kode yang sama akan memiliki jumlah spasi yang sama pula, sehingga indentasi yang tidak sesusai akan menyebabkan error.

```
ini_angka = 10 + \
    20 + \
        30
    print(ini_angka)
```

```
\Tugas prak\codingan.py", line 6
    print(ini_angka)
IndentationError: unexpected indent
PS C:\>
```

  Biasanya pada kodingan akan ditandai dengan garis bawah merah dan akan muncul pesan seperti di atas Ketika error terjadi.

- Variable dan Tipe data
  Variabel merupakan tempat menyimpan data atau nilai. Untuk mendeklarasikan nilai tersebut diperlukan juga tipe data yang sesuai. Tipe data dasar yang ada adalah :

```
pagi = true #boolean
prima = 1, 3, 5 #int
phi = 3.14 #float
nama = "Andreas" #string / str
```

- Operator
  - Operator aritmatika : operator untuk menjalankan fungsi operasi matematik.

```
a = 1 + 2  #penjumlahan

b = 4 - 3  #pengurangan

c = 5 * 6  #perkalian

d = 8 / 4  #pembagian

e = 2 ** 3 #pemangkatan

f = 6 // 2 #pembagian bulat

g = 50 % 2 #modulus
```

- Operator perbandingan : operator untuk mebandingkan 2 buah bilangan dengan output True atau False.

```
a = 1 > 2  #operator lebih dari

b = 4 < 3  #operator kurang dari

c = 6 == 6  #operator sama dengan

d = 8 != 4  #operator tidak sama dengan

e = 2 >= 3 #operator lebih dari sama dengan

f = 6 <= 2 #operator kurang dari sama dengan
```

- Operator penugasan : operator untuk memberi sebuah nilai ke variable.

```
a = 5   #memberi nilai variabel a

b += 4  #b = b + 4

c -= 5  #c = c - 5

d /= 6  #d = d / 6

e *= 7  #e = e * 7

f **= 1 #f = f ** 1

g //=2  #g = g // 2

h %= 1  #h = h % 1
```

- Operator Bitwise : operator yang melakukan operasi bit terhadap operand.

```
a = 4 & 9   #bitwise and

b = 4 | 9   #bitwise or

a = 6       #bitwise not
b = ~a

a = 4^3     #bitwise xor

b = 4       #bitwise right shift
a = b >> 5

a = 1       #bitwise left shift
b = a << 2
```

- Operator identitas : operator untuk memeriksa apakah dua buah nilai variable berada pada lokasi yang sama.

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

a is b      #bernilai False karena 1 != 2
a is not c  #bernilai True karena 1 != 3
```

- Operator keanggotaan : operator ntuk memeriksa apakah nilai variable merupakan anggota suatu data.

```
nama = "Andreas Pascalis"
nama_akhir = "Pascalis"
nama_belakang = "Tristan"

nama_akhir in nama  #true karena "Pascalis" ada pada "Andreas Pascalis"
print(nama_akhir in nama)

nama_belakang not in nama #True karena "Tristan" tidak dalam "Andreas Pascalis"
print(nama_belakang not in nama)

nama_belakang in nama
print(nama_belakang in nama)  #False karena "Tristan" tidak dalam "Andreas Pascalis"
```

- Operator logika : operator untuk mengecek logika And, Or dan Not.

```
a = 1 and 12>9  #True bila kedua kondisi terpenuhi

b = 13<8 or 900 #True selama salah satu kondisi terpenuhi

c = 5   #True bila nilai dari operand salah
d = 5 != c
```

- Tipe data bentukan
  - List : sebuah kumpulan data terurut, dapat diubah dan nilai anggota boleh sama.

```
mie_goreng = ["bumbu", "kuah", "mie", "telur", "air"]  #contoh list
mie_goreng = [] #list kosong
```

  - Tuple : kumpulan berbagai tipe data yang isinya tidak dapat diubah.

```
manusia = ("tinggi", "Andi", 15) #contoh tuple
manusia =() #tuple kosong
```

  - Dictionary : kumpulan data yang tidak berurutan, dapat dibuah, dan anggotanya harus unik. Dan datanya memiliki penunjuk yang unik sebagai karakteristik.

```
kode = {'a':"Alice", 'b':"Barbeloth", 'm':"mixue"} #contoh dict
kode = {} #dictionary kosong
```

  - Set : kumpulan data yang tidak berurut maupun berindeks dan anggotanta harus unik.

```
artifact = {"flower", "plume", "sand", "goblet", "crown"} #contoh set
artifact = set() #set kosong
```

- Percabangan : ada IF, IF-ELSE, IF-ELSE-IF, dan nested IF.
  - Percabangan IF : percabangan dengan 1 kondisi.

```
a = int(input("masukkan angka : "))

if a <= 12:
    print(str(a) + " ya?") #akan dieksekusi bila a kurang dari sama dengan 12
```

  - Percabangan IF-ELSE : percabangan 2 kondisi dimana bila kondisi IF adalah False maka dijalankan kondisi ELSE/default.

```
a = int(input("masukkan angka : "))

if a <= 12:
    print(str(a) + " ya?") #akan dieksekusi bila a kurang dari sama dengan 12
else:
    print("banyak kali") #dieksekusi apabila kondisi if tidak terpenuhi
```

  - Percabangan IF-ELSE-IF : percabangan dengan lebih dari 2 kondisi kondisi else hanya akan dijalankan ketika semua opsi kondisi lain benar benar tidak terpenuhi.

```
a = int(input("masukkan angka : "))

if a <= 12:
    print(str(a) + " ya?") #akan dieksekusi bila a kurang dari sama dengan 12
elif a <= 40 and a>12:
    print(str(a)+ "???") #akan dieksekusi bila if tidak terpenuhi
else:
    print("banyak kali") #dieksekusi apabila kondisi if dan elif tidak terpenuhi
```

- Nested if : percabangan yang berada dalam percabangan.

```
a = int(input("masukkan angka : "))

if a <= 12: #kondisi awal
    if a == 0:  #akan dieksekusi bila kondisi awal terpenuhi
        print("ini nol.")
    else:   #dieksekusi bila nested if tidak terpenuhi
        print("angka kecil.")
else:   #kondisi akhir bila IF awal tidak terpenuhi
    print("angka besar.")
```

- Perulangan : ada 2 jenis perulangan dalam pyhton yaitu perulangan for dan while.
  - Perulangan for dengan 1 parameter : perulangan dilakukan dimulai dari indeks 0 sampai parameter tujuan.

```
for i in range (10): #akan print 0-9
    print(i)
```

  - Perulangan for dengan 2 parameter : perulangan dilakukan dari parameter pertama ke parameter kedua.

```
for j in range (2, 8): #akan print 2-7
    print(j)
```

  - Perulangan for dengan 3 parameter : perulangan dilakukan dari parameter 1 ke parameter 2 dengan parameter 3 berperan sebagai increment.

```
for k in range (0, 10, 2): #perulangan dari 0 sampai 10 dengan incremen 2
    print(k)
```

  - Perulangan while : perulangan ini akan terus dilakukan selama kondisi while masih terpenuhi.

```
i=1
while(i<=10): #akan terus dilakukan sampai while terpenuhi
    print("halo.")
    i+=1
```

- Fungsi : fungsi adalah bentuk kode yang berperan sebagai kerangka suatu perintah yang dapat dipanggil berulang ulang sehingga kita tidak perlu menghabiskan waktu membuat perintah yang sama berkali kali.

```python
def mahasiswa (nama, nilai):
    if nilai >= 60:
        return f"{nama}, kamu telah lulus ujian."
    else:
        return f"{nama}, kamu gagal ujian."

list_mhs = [
    ["Basuki", 90], ["Andi", 30], ["Damian", 80]
    ]

for i in list_mhs:
    print(mahasiswa(i[0], i[1]))
```

- Fungsi dengan parameter otomatis (*args) : parameter fungsi (*args) dianggap sebagai list.

```python
def ini_fungsi (*args): #deklarasi fungsi
    print(args[0] + args[1])

ini_fungsi (1, 2)
```

- Fungsi dengan parameter otomatis (*kwargs) : parameter (*kwargs) dianggap sebagai dictionary.

```python
def ini_fungsi(**kwargs):
    print(kwargs['a'] + kwargs['b'])

ini_fungsi(a='a', b='b')
```

- Kelas : kelas merupakan blueprint dari objek yang ingin kita buat. Satu buah kelas dapat membuat banyak objek dan kelas sendiri tidak bisa langsung digunakan, harus diimplementasikan menjadi sebuah objek dulu, dapat disebut Instansiasi.
  - Konstruktor kelas : method yang pasti akan dijalankan saat kelas dibuat
  - Atribut : variable yang mendefinisikan kelas.

- Method : fungsi dalam kelas. Setiap method memiliki atribut dan dapat mewarisi atribut objek dengan parameter (self).

```
class Mahasiswa:
    manusia = "manusia hidup" #atribut
    def __init__(self, nama, NIM): #konstruktor
        self.nama = nama #atribut objek
        self.nim = NIM

    def cetak_data(self):    #method
        print(self.nama + " dengan NIM : " + str(self.nim))

Andre = Mahasiswa("Andre", 121140017) #Andre merupakan objek
print(Andre.manusia) #print "manusia hidup"
print(Andre.nama)    #print "Andre"
Andre.cetak_data()
```

- Magic Method : adalah metode yang diawali dan diakhiri dengan double underscore. 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.

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

    def __add__(self, objek):    #magic method
        return self.angka + objek.angka

angka1 = Angka(5)
angka2 = Angka(6)

print(angka1 + angka2)
```

- Abstraksi : adalah konsep dalam PBO/OOP dimana model yang dibuay hanya memperlihatkan atribut yang esensial dan menyembunyikan detai-detail lain yang dianggap tidak penting bagi user. Sehingga user hanya perlu tahu apa yang objek dapat lakukan tanpa perlu mengetahui bagaimana objek melakukannya.

Contoh paling umum ada pada mobil. Pengendara tahu bahwa mobil dapat menyala, berjalan, berhenti. Namun pengendara tidak tahu mekanisme keseluruhan bagaimana hal itu terjadi.

```python
class Mobil:
    def __init__(self, warna, merk, tahun):
        self.warna = warna
        self.merk = merk
        self.tahun = tahun

    def berjalan(self):
        print("mobil " + self.merk + " bergerak maju.")
    def mundur(self):
        print("mobil " + self.merk + " bergerak mundur.")
    def berhenti (self):
        print("mobil " + self.merk + " berhenti.")

toyota = Mobil("merah", "avanza", 2004)
toyota.berhenti()
```

- Enkapsulasi : merupakan metode yang digunakan untuk mengetur struktur kelas dengan cara menyembunyikan data dan akses serta alur kerja kelas tersebut. Ada 3 macam access modifier dalam python yaitu public, private, dan protected.
    - Public Access Modifier : adalah deklarasi default Ketika kita membuat kelas, variable, dan method pada umumya.
    - Protected Access Modifier : deklarasi yang masuk dalam protected access berarti hanya dapat diakses oleh kelas turunannya. Dideklarasikan dengan 1 underscore(_).
    - Private Access Modifier : atribut dan kelas yang masuk dalam private access hanya dapat diakses oleh kelas itu sendiri. Dideklarasikan dengan 2/double underscore (__).

```python
class Mahasiswa:
    def __init__(self,nama, umur, NIM):
        self.nama = nama    #public attribute
        self._umur = umur   #protected atrribute
        self.__nim = NIM    #private attribute

    def lihat_nama(self):   #public method
        print(self.nama)
    def _lihat_umur(self):  #protected method
        print(self._umur)
    def __lihat_nim(self):  #private method
        print(self.__nim)

    def get_nim(self):      #getter method
        return self.__nim
    def set_nim(self, nim_baru):    #setter method
        self.__nim = nim_baru

    @property   #decorator
    def nim(self):
        return self.__nim
    @nim.setter #decorator
    def gaji(self, nim_baru):
        self.__nim = nim_baru
```

- Pemanggilan :

```python
andre = Mahasiswa ("Andre", 21, 1211400) #inisiasi objek
print(andre.nama)   #public dapat diakses di luar kelas
print(andre._umur)  #protectec masih dapat diakses di luar kelas
#print(andre.__nim) #private tidak dapat diakses di luar kelas
print(andre.get_nim())  #dengan metode getter kita dapat mengakses private attribute
print()
andre.lihat_nama()
andre._lihat_umur()
#andre.__lihat._nim()
```

- Inheritance : adalah konsep dalam OOP. Pada inheritance kita dapat menurunkan kelas dari kelas lain untuk saling berbagi method.

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

class anak(ayah):   #kelas anak mewarisi atribut dari ayah
    pass
```

  - Inheritance identik : merupakan bentuk pewarisan dimana kita menambahkan constructor pada class child sehingga child memiliki construcotr sendiri tanpa kehilangan constructor parent classnya. Metode ini dilakukan menggunakan kunci

super().

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

class anjing(hewan):
    def __init__(self, nama, umur, warna):
        super().__init__(nama, umur)    #inheritance identik
        self.warna = warna

river = anjing("river", 12, "coklat")
print(river.nama)
```

- Multiple inheritance : bentuk inheritance dimana kelas anak akan mewarisi sifat lebih dari satu kelas parent. Dalam multiple inheritance method pertamalah yang dilihat.

```
class parent1:
    pass

class parent2:
    pass

class child(parent1, parent2):
    pass
```

- Polymorphism : adalah konsep dalam OOP dimana konsep ini memungkinkan digunakannya suatu interface yang sama untuk memerintah objek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda. Dalam python terdapat method khusus yang disebut method overloading, dan overriding.
    - Overriding : menimpa suatu metode pada parent class dengan mendefinisikan Kembali method dengan nama yang sama pada child class.

```
class ayah:
    def bekerja(self):
        print("ayah akan pergi berkerja")
class kakak(ayah):
    def bekerja(self):
        print("kakak akan pergi ke toko")
class ibu(ayah):
    def bekerja(self):
        print("ibu pergi ke pasar")

keluarga = [ayah(), ibu()]
keluarga[0].bekerja()   #print "ayah akan pergi bekerja"
keluarga[1].bekerja()   #print "ibu pergi ke pasar"
```

Pada contoh di atas semua child class mewarisi sifat class parent ayah yaitu bekerja. Namun setiap child class memiliki definisi yang berbeda beda dalam bekerja sehingga menghasilkan output yang berbeda.

- Overloading : metode dimana sebuah class memiliki fungsi dengan nama yang sama dan argument yang berbeda.

```
class BangunDatar:
    def hitung_luas(self):
        pass

class Rectangle(BangunDatar):
    def hitung_luas(self, panjang, lebar):
        return panjang*lebar
class Triangle(BangunDatar):
    def hitung_luas(self, alas, tinggi):
        return alas * tinggi * 0.5
class Circle(BangunDatar):
    def hitung_luas(self, jari_jari):
        return 3.14 * jari_jari * jari_jari

kotak = Rectangle()
print(kotak.hitung_luas(5, 8))
```

Pada contoh di atas semua child class memiliki method hitung_luas, namun semua child class memiliki definisi yang berbeda beda.