Nama: Harun Abdulkarim Khafid

NIM: 121140147

Prak PBO RB

RESUME

Modul 1

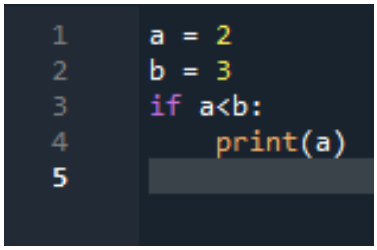
1. **Pengenalan Bahasa pemrograman python**

   Dibuat oleh Guido Van Rossum (1980) di Centrum Wiskunde & Informatica Belanda. Mementingkan readability seperti space untuk memisahkan blok-blok kode

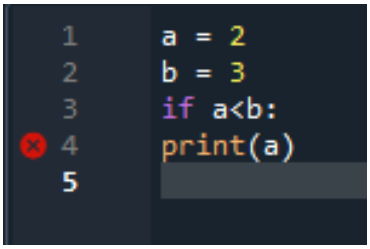
2. Dasar Pemrograman Python

   Baris baru menandakan akhir dari statement tapi bisa terdiri dari beberapa baris menggunakan backslash (\)

   perbedaan spasi atau indentasi sangat mempengaruhi hasil codingan, contohnya



Benar



salah

   Jumlah spasi harus sama sebagai grouping blok, tidak perlu memakai kurung kurawal

   Deklarasi variable perlu diketahui tipe datanya

   - Bool = True / False
   - Int = bilangan bulat (1, 2, 3, 4)
   - Float = bilangan real (5.8 ; 9.99)
   - String = Teks (“semangka”,”kucing”)

   Di python tipe datanya akan otomatis sesuai dengan nilai yang dideklarasikan

2.1 Operator

2.1.1 Aritmatika

Digunakan untuk operasi matematika

- Penjumlahan (+)
- Pengurangan (-)
- Perkalian (*)
- Pembagian (/)
- Pemangkatan (**)
- Pembaian tanpa koma (//)
- Modulus (%)

```
#operasi Matematika

a = 10
b = 6

print(a+b)  #penjumlahan
print(a-b)  #Pengurangan
print(a*b)  #Perkalian
print(a/b)  #pembagian
print(a**b) #Pemangkatan
print(a//b) #Pembagian tanpa koma
print(a%b)  #Modulus

```

```
In [9]: runcell(0, 'C:/Users/
16
4
60
1.6666666666666667
1000000
1
4
```

2.1.2 perbandingan

Membandingkan 2 nilai

- Lebih besar (>)
- Lebih kecil (<)
- Sama dengan (==)
- Tidak sama dengan (!=)
- Lebih besar sama dengan (>=)
- Lebih kecil sama dengan (<=)

```
#operasi Perbandingan

a = 10
b = 6

print(a>b)  #lebih besar
print(a<b)  #lebih kecil
print(a==b) #sama dengan
print(a!=b) #tidak sama dengan
print(a>=b) #lebih besar ssama dengan
print(a<=b) #Lebih kecil sama dengan

```

```
In [10]: runcell(0, 'C:/U
True
False
False
True
True
False
```

2.1.3 Penugasan

Pemberian sebuah nilai

- C = a+b
- C += a
- C-=a
- C *=a
- C /=a
- C **=a
- C //a
- C %=a

```
a = 9
b = 20
c = a+b # 20 + 9 = 29
print(c)
c -= a # 29 - 9 = 20
print(c)

```

2.1.4 Logika

Untuk operasi logika

- And
- Or
- Not

Adapun beberapa jenis operator lainnya seperti

- Bitwise = melakukan operasi bit terhadap operand
- Identitas = memerika dua buah nilai berada pada lokasi yang sama
- Keanggotaan = memeriksa apakah suatu nilai ada di dalam suatu data

2.2 Tipe data bentukan

- List = Menyimpan berbagai tipe data dan isinya bisa diubah
- Tuple = Menyimpan berbagai tipe data yang isinya tidak bisa diubah
- Set = Element data yang disimpan harus unik
- Dictionary = Menyimpan data berupa pasangan penunjuk dan nilai

```
1  a = [4,3,2,1] , ["bogor","depok","Jakarta"] #list
2  b = ('xyz', 1, 3.14)                        #Tuple
3  c = {'alpha','beta','gamma'}                #set
4  d = {'nama' : 'abdul'}                      #dictionary
5
```

2.3 Percabangan

Terdapat 4 jenis percabangan

- Percabangan IF = digunakan Ketika hanya ada satu kondisi
- Percabangan IF-ELSE = digunakan Ketika ada dua kondisi
- Percabangan IF-ELSE-IF = digunakan Ketika ada lebih dari dua kondisi
- Nested IF = digunakan untuk percabangan di dalam percabangan

```
1
2  a = int(input("masukkan nilai: "))
3  if a > 60:
4      print("anda lulus")
5      if a > 80:
6          print("nilai anda A")
7      elif a > 70:
8          print("nilai anda B")
9      else:
10         print("nilai anda C")
11 else:
12     print("anda tidak lulus")
13
```

2.4 Perulangan for

Digunakan untuk iterasi pada urutan list, tuple atau string

Sintaks dasar pada python adalah

for i in (kondisi):

```
a = [4,3,2,1]  #list

for i in a:
    print(i)

```

Bisa menggunakan indeks atau range

for i in range(x,y,z)

- 1 parameter = perulangan akan dilakukan sampai indeks x
- 2 parameter = perulangan akan dilakukan dari indeks x sampai indeks y
- 3 parameter = perulangan akan dilakukan dari indeks x sampai indek y dengan penambahan z

Ada juga penggunaan break dan continue, break digunakan Ketika perulangan sampai dikondisi tertentu maka perulangan akan berhenti, continue Ketika perulangan sampai di kondisi tertentu maka kondisi itu dilewati dan dilanjutkan berikutnya

2.5 While

selama kondisi nya terpenuhi maka perulangan akan dilakukan terus

```
a = input("masukkan password: ")
while(a !="ganteng"):
    a = input("salah, masukkan password: ")

```

2.6 Fungsi

Dapat mengakses suatu blok kode tanpa harus menulis ulang

```
def kerja(nama,umur, gaji):
    if gaji > 3_000_000:
        print(f"{nama} , {umur} tahun memiliki gaji diatas umr")
    else:
        print(f"{nama} , {umur} tahun memiliki gaji dibawah umr")
a1 = kerja("abdul",19,4_000_000)
a2 = kerja("karim",23,2_100_00)
```

Modul 2

Objek dan kelas dalam python

1. Kelas

   Kelas digunakan untuk mendefinisikan atribut/property untuk objek yang akan kita instansiasi nantinya

   Gunakan kata kunci class diikuti nama kelas dan titik dua, untuk membuat kelas harus ada method berupa __init__() yang merupakan konstruktor. Kata kunci pass bisa dipakai untuk membuat kelas kosong

   1.1 atribut

   terdapat 2 jenis yaitu atribut kelas dan atribut objek, atribut kelas adalah sifat yang dimiliki kelas dan nantinya akan dimiliki objeknya juga sedangkan atribut objek sifat yang hanya dimiliki objek tersebut saja

```
1   class Robot:
2       jumlah_turn = 0
3       def __init__(self, nama, health, damage): #atribut kelas
4           self.nama = nama
5           self.health = health
6           self.damage = damage
7
8       def lakukan_aksi(self,nama):    #atribut Objek
9           if nama == "Antares":
10              if self.jumlah_turn % 3 == 0:
11                  damage = self.damage * 1.5
12              else:
13                  damage = self.damage
14
```

   1.2 method

   method diibaratkan sebagai proses yang dapat dilakukan sebuah objek seperti gambar diatas method lakukan_aksi

   pemanggilan

```
106 p1 = Antares("Antares",50000,5000)
107 aksi1 = p1.lakukan_aksi(p1.nama)
```

2. objek

   bisa dibilang pengganti untuk pemanggilan sebuah kelas

3. magic method

method yang diawali dan diakhiri dengan double underscore, method ini dipanggil sistem secara internal, tujuannya untuk mengubah sifat bawaah dari suatu objek

4. konstruktor

   pada saat objek dibuat method ini sudah pasti secara otomatis dijalankan, dapat menerima argument Ketika objek dibuat

5. Destruktor

   fungsi untuk menghapus objek(otomatis), tujuannya melakukan final cleaning up contohnya __del__()

6. Setter dan getter

   Enkapsulasi agar tidak terjadi perubahan data. Setter untuk menetapkan nilai sedangkan getter untuk mengambil nilai

7. Decorator

   Melalui property ini kita tidak perlu membuat fungsi lagi dengan nama yang berbeda beda

Modul 3

Abstraksi dan Enkapsulasi

1. **Abstraksi**

   Dalam pembuatan abstraksi user hanya mengetahui apa yang akan objek tersebut lakukan tetapi tidak mementingkan mekanisme yang terjadi di belakang. Wujud nyata suatu kelas dinamakan instance contoh: str adalah kelas dan “python” adalah objek

2. **Enkapsulasi**

   Bedanya dengan abstraksi, enkapsulasi menyembunyikan atribut suatu entitas serta metode untuk melindungi infromasi dari luar. Maka dari itu terdapat 3 acces modifier yaitu

   - Public Acces Modifier

     Saat kita membuat method maka method tersebut otomatis bersifat public

   - Protected Access Modifer

     Hanya dapat diakses oleh kelas turunan darinya

     ```
     1  class bank:
     2
     3      def __init__(self, nama, saldo):
     4          self._nama = nama
     5          self._saldo = saldo
     6
     ```

   - Private acces modifier

     Hanya dapat diakses di dalam kelas itu sendiri

```
1   class bank:
2
3       def __init__(self, nama, saldo):
4           self.__nama = nama
5           self.__saldo = saldo
6
```

Agar dapat diakses dari luar maka diperlukan fungsi setter and getter

Membuat instance objek dan cara mengakses nya

```
6       def prints(self):
7           print(self.__nama)
8   akun = bank("harun",9_000_000)
9   akun.prints()
```

Cara yang lebih baik untuk memodifikasi atribut dari suatu objek dapat menggunakkan fungsi berikut :

- getattr(obj, name[, default]) – Mengakses atribut dari objek
- hasattr(obj, name) – Memeriksa apakah suatu objek memiliki atribut tertentu
- setattr(obj, name, value) – Mengatur nilai atribut. Jika ternyata atribut tidak ada, maka atribut tersebut akan dibuat.
- delattr(obj, name) – Menghapus atribut dari suatu objek.

Modul 4

Pewarisan dan Polimorfisme

1. **Inheritance (pewarisan)**

   Contohnya adalah ada kelas parent lalu kelas turunannya akan memiliki atribut dan method yang sama dengan kelas parent

```python
class mahluk_hidup:
    def __init__(self, nama, umur, jenis):
        self.nama = nama
        self.umur = umur
        self.jenis = jenis
    def prints(self):
        print(self.nama,self.umur,"tahun",self.jenis)

class manusia(mahluk_hidup):
    pass
class hewan(mahluk_hidup):
    pass

orang = manusia("harun", 19, "manusia")
kucing = hewan("kucing",2,"mamalia")
kucing.prints()
```

Jadi tidak perlu memanggil kelas parentnya

Jika ada atribut tambahan pada kelas turunannya maka

```python
class manusia(mahluk_hidup):
    def __init__(self, nama, umur, jenis, alamat):
        super().__init__(nama,umur,jenis)
        self.alamat = "kosong"
```

2. **Polymorphism**

   Dalam Pemrograman Berbasis Objek 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.

3. **Override/overriding**

   Menimpa method yang ada di parents karena Namanya sama, jadi method pada parents tidak berlaku

```python
class anime:
    def tampil(self):
        print("naruto anime terbaik")

class boruto(anime):
    def tampil(self):
        print("boruto anime terbaik")
a = boruto()
a.tampil()
```

a

4. **Overloading**

   Metode overloading mengizinkan sebuah class untuk memiliki sekumpulan fungsi dengan nama yang sama dan argumen yang berbeda.

```python
class naruto:
    def __init__(self,nama):
        self.nama = nama
    def serang(self):
        print("rasengan")

class sasuke:
    def __init__(self,nama):
        self.nama = nama
    def serang(self):
        print("kamui")

def pukul(obj):
    obj.serang()

sasuk = naruto("naruto")
narut = sasuke("sasuke")

pukul(sasuk)
pukul(narut)
```

5. **Multiple inheritance**

   Pewarisan bisa dilakukan dari banyak parents

```python
class naruto:
    def kekuatan1(self):
        print("aku bisa rasengan")
class sasuke:
    def kekuatan2(self):
        print("aku bisa chidori")

class boruto(naruto,sasuke):
    pass

a = boruto()
a.kekuatan1()
a.kekuatan2()
```

6. **Method Resolution Order**

   urutan pencarian metode dalam hirarki class. Jika method tidak ada di kelas pertama maka akan dicari dikelas berikutnya. Digunakan dalam multiple inheritance, jadi pencarian dilakukan dari urutan paling kiri ke paling kanan

7. Dynamic cast

   Ada 2 tipe konvesi yaitu

   - Implisit

     Python secara otomatis mengkonversikan tipe data ke tipe data lainnya tanpa ada campur tangan pengguna.

```
1   a = 3.4
2
3   print(type(a)) #output class float
```

- Eksplisit

  Pengguna mengubah tipe data sebuah objek ke tipe data lainnya dengan fungsi yang sudah ada dalam python seperti int(), float(), dan str(). dapat berisiko terjadinya kehilangan data.

```
1   a = "1123"
2   print("tipe data a adalah",type(a)) #string
3   a = int(a)
4   print("tipe data a adalah",type(a)) #int
```

8. Casting
   - Downcasting = parent mengakses atribut child
   - Upcasting = child mengakses atribut parent
   - Type casting = semua variabel atau instansi di Python pada dasarnya merupakan objek (kelas) yang sifat/perilakunya dapat dimanipulasi jika dibutuhkan