

**RANGKUMAN MODUL PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**



**Disusun Oleh:
Fransiskus Xaverius Gunawan
121140010
RB**

**PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2023**

DAFTAR ISI

DAFTAR ISI	2
MODUL 1	
Pengenalan dan Dasar Pemrograman Python	4
1.1. Pengenalan Bahasa Pemrograman Python	4
2.1. Dasar Pemrograman Python	4
2.1.1. Sintaks Dasar	4
2.2. Variable dan Tipe Data Primitive	5
2.3. Operator	5
2.3.1. Operator Aritmatika	5
2.3.2. Operator Perbandingan	6
2.3.3. Operator Penugasan	6
2.4. Tipe Data Bentuk	6
2.5. Percabangan	6
2.5.1. Percabangan IF	7
2.5.2. Percabangan IF-ELSE	7
2.5.3. Percabangan IF-ELSE IF	7
2.5.4. Nested IF	7
2.6. Perulangan	8
2.6.1 For Loop	8
2.6.2. While Loop	8
2.7. Fungsi	9
MODUL 2	
Objek dan Kelas	10
1 . Class	10
1.1. Property/Atribut	10
1.2. Method	11
2. Objek	11
3. Magic Method	11
4. Konstruktor	12
5. Destruktor	12
6. Setter & Getter	13
7. Decorator	13
MODUL 3	
ABSTRAKSI DAN ENKAPKULASI	14
1.Abstraksi	14
2.Enkapulasi	14
2.1. Public Access Modifier	14
2.2. Protected Access Modifier	15
2.3. Private Access Modifier	15
MODUL 4	
Pewarisan dan Polimorfisme	16

1. Inheritance	16
1.1. Inheritance Identik	17
2. Polymorphism	17
3. Override	18
4. Casting	18
4.1. Down Casting	18
4.2. Upcasting	19
5. Multiple Inheritance	19

MODUL 1

Pengenalan dan Dasar Pemrograman Python

1.1. Pengenalan Bahasa Pemrograman Python

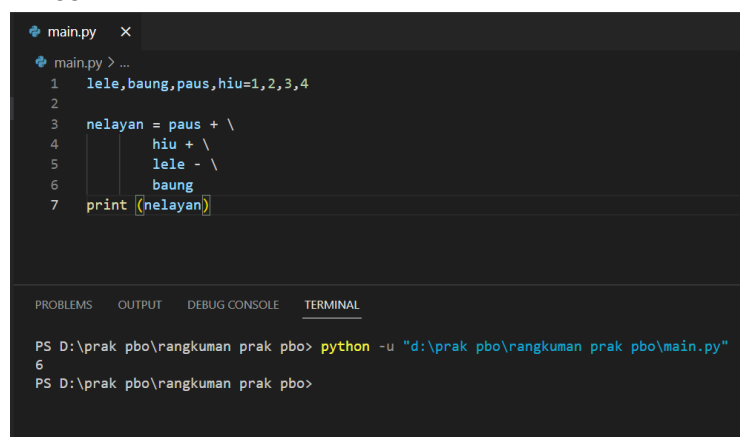
Bahasa python diciptakan oleh Guido Van Rossum pada tahun 1980. Bahasa pyhton mendukung pemrograman object oriented,functional dan struktural. Python dapat digunakan dalam pemrograman desktop,mobile,IoT,web,automatisasi,hacking,robotika,dan lain lain.

2.1. Dasar Pemrograman Python

2.1.1. Sintaks Dasar

Statement

Pada python pada setiap perintah adalah baris baru atau newline. Dapat membuat perintah dari beberapa baris menggunakan backslash (\)



```
main.py x
main.py > ...
1 lele,baung,paus,hiu=1,2,3,4
2
3 nelayan = paus + \
4         hiu + \
5         lele - \
6         baung
7 print (nelayan)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\prak pbo\rangkuman prak pbo> python -u "d:\prak pbo\rangkuman prak pbo\main.py"
6
PS D:\prak pbo\rangkuman prak pbo>
```

Baris dan Indentasi

Python tidak memakai kurung kurawal sebagai grouping namun menggunakan tab

```
main.py X
main.py > ...
1 for i in range(5):
2     print(i+1)
3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS D:\prak pbo\rangkuman prak pbo> python -u "d:\prak pbo\rangkuman prak pbo\main.py"
1
2
3
4
5
PS D:\prak pbo\rangkuman prak pbo>
```

2.2. Variable dan Tipe Data Primitive

Variable adalah lokasi penyimpanan suatu nilai. Untuk mendeklarasikan variabel, diperlukan tipe data agar nilai dalam variabel tersebut dapat di definisikan. Terdapat beberapa tipe data, yaitu int, string, float, bool.

```
main.py
main.py > ...
1 variable1=False #boolean
2 variable2=5 #integer
3 variable3=5.5 #float
4 variable4='lima' #string
```

2.3. Operator

Terdapat 3 jenis operator dalam python, yaitu operator aritmatika, perbandingan, dan penugasan.

2.3.1. Operator Aritmatika

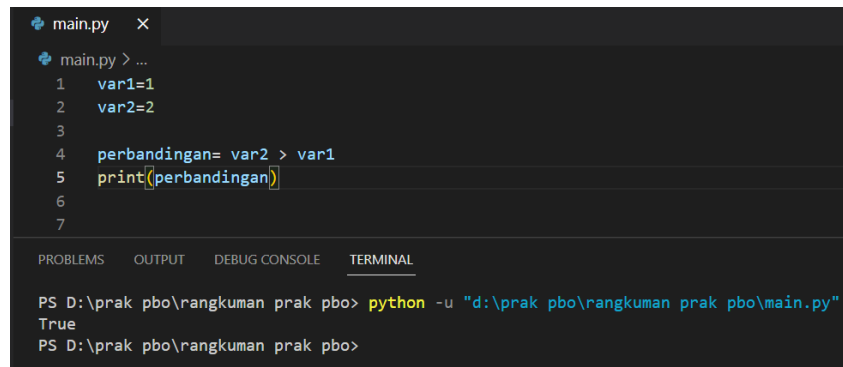
Operator aritmatika seperti penjumlahan, pengurangan, perkalian, pembagian, modulus

```
main.py X
main.py > ...
1 var1=1
2 var2=2
3
4 penjumlahan=var1+var2
5 print(penjumlahan)
6
7 pengurangan=var2-var1
8 print(pengurangan)
9
10 pembagian=var2/var1
11 print(pembagian)
12
13 perkalian=var2*var1
14 print(perkalian)
15
16

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS D:\prak pbo\rangkuman prak pbo> python -u "d:\prak pbo\rangkuman prak pbo\main.py"
3
1
2.0
4
PS D:\prak pbo\rangkuman prak pbo>
```

2.3.2. Operator Perbandingan

Operator yang digunakan untuk membandingkan 2 nilai



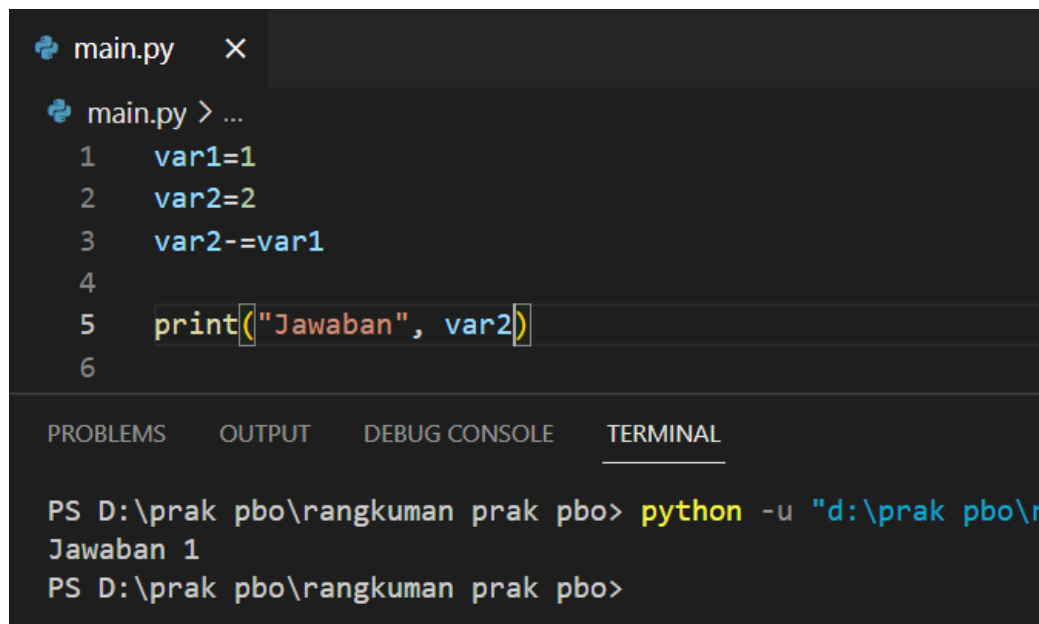
```
main.py X
main.py > ...
1  var1=1
2  var2=2
3
4  perbandingan= var2 > var1
5  print(perbandingan)
6
7

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\prak pbo\rangkuman prak pbo> python -u "d:\prak pbo\rangkuman prak pbo\main.py"
True
PS D:\prak pbo\rangkuman prak pbo>
```

2.3.3. Operator Penugasan

Operator yang digunakan untuk memberikan nilai ke variable, seperti =, +=, -=, dan lain lain



```
main.py X
main.py > ...
1  var1=1
2  var2=2
3  var2-=var1
4
5  print("Jawaban", var2)
6

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\prak pbo\rangkuman prak pbo> python -u "d:\prak pbo\r
Jawaban 1
PS D:\prak pbo\rangkuman prak pbo>
```

2.4. Tipe Data Bentukan

Terdapat 4 tipe data bentukan, yaitu List, Tuple, Set, Dictionary.

- List adalah kumpulan data yang terurut dapat diubah
- Tuple adalah kumpulan data yang terurut tetapi tidak dapat diubah
- Set adalah kumpulan data yang tidak urut, tidak terindeks, dan tidak ada anggota yang sama
- Dictionary adalah kumpulan data yang tidak urut, dapat diubah, dan tidak ada anggota yang sama

2.5. Percabangan

Percabangan adalah suatu kondisi jika terdapat 2 kondisi atau lebih. Percabangan terdiri dari IF, IF-ELSE, IF ELSE IF, dan Nested IF.

2.5.1. Percabangan IF

Percabangan jika terdapat 2 kondisi, dan untuk mengeluarkan output 1 (benar)

```
main.py > ...  
1  var1=1  
2  var2=2  
3  
4  if(var2>var1):  
5      print("Benar")
```

2.5.2. Percabangan IF-ELSE

Percabangan jika terdapat 2 kondisi, dengan terdapat 2 output (benar/salah)

```
main.py > ...  
1  var1=1  
2  var2=2  
3  
4  if(var2>var1):  
5      print("Benar")  
6  else:  
7      print("salah")
```

2.5.3. Percabangan IF-ELSE IF

Percabangan jika terdapat lebih dari 2 kondisi

```
main.py > ...  
1  var1=1  
2  var2=2  
3  
4  if(var2>var1):  
5      print("Benar")  
6  elif(var1<var2):  
7      print("Benar")  
8  elif(var2<var1):  
9      print("salah")
```

2.5.4. Nested IF

Perulangan dalam perulangan, jika terdapat kondisi didalam kondisi, maka menggunakan nested if

```

main.py > ...
1  var1=1
2  var2=2
3
4  if(var2>var1):
5      print("Benar")
6      if(var1<var2):
7          print("benar")
8  else:
9      print("salah")

```

2.6. Perulangan

Perulangan terdiri dari beberapa cara, yaitu dengan while dan for. Program akan terus mengulang sesuai dengan perintah yang di input oleh user.

2.6.1 For Loop

Perulangan digunakan untuk iterasi, string.

```

main.py > ...
1  for i in range(5):
2      print(i+1)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS D:\Kuliah\Semester 4\Prak PBO> python -u
1
2
3
4
5
PS D:\Kuliah\Semester 4\Prak PBO>

```

2.6.2. While Loop

Perulangan yang dilakukan sampai suatu kondisi tertentu terpenuhi

```

main.py > ...
1  i = 1
2  while i < 6:
3      print(i)
4      i += 1

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

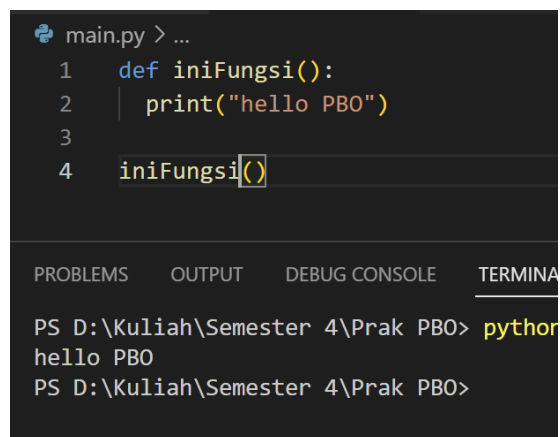
```

PS D:\Kuliah\Semester 4\Prak PBO> py
1
2
3
4
5
PS D:\Kuliah\Semester 4\Prak PBO>

```


2.7. Fungsi

Fungsi adalah blok kode yang hanya berjalan ketika dipanggil. Fungsi dapat dilewati dengan menggunakan pass



```
main.py > ...
1  def iniFungsi():
2      print("hello PBO")
3
4  iniFungsi()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Kuliah\Semester 4\Prak PBO> pythor
hello PBO
PS D:\Kuliah\Semester 4\Prak PBO>
```

MODUL 2

Objek dan Kelas

1 . Class

Class dalam python merupakan konstruktor atau blueprint untuk rancangan yang ingin kita buat. Kita dapat menggunakan untuk objek. Untuk menggunakan kelas, keyword yang dipakai adalah **class** lalu diikuti dengan nama dari class tersebut

```
main.py > meong
1 class meong:
2
3     pass
```

dalam class dapat menggunakan method untuk konstruktor dalam class meong ,seperti `__init__`

```
main.py > meong
1 class meong:
2     def __init__(self,simba,simbi):
3         self.simba=simba
4         self.simbi=simbi
5
6     pass
```

1.1. Property/Atribut

Property dapat di deklarasikan dalam class ataupun objek. Atribut kelas dimiliki oleh kelas, dan atribut objek dimiliki oleh objek.

```

main.py > meong
1 class meong:
2     jumlahSimba=1
3     jumlahSimbi=2
4
5     def __init__(self,simba,simbi):
6         self.simba=simba
7         self.simbi=simbi
8
9     pass

```

1.2. Method

Method adalah fungsi dalam kelas , sama seperti dengan atribut, semua objek yang sama akan memilimi method yang sama

```

main.py > meong
1 class meong:
2     jumlahSimba=1
3     jumlahSimbi=2
4
5     def __init__(self,simba,simbi):
6         self.simba=simba
7         self.simbi=simbi
8
9     def jalan(self):
10        print("majuuuu")
11
12    pass

```

2. Objek

Ojek adalah sesuatu yang mewakili kelas. Objek digunakan untuk pengganti pemanggilan kelas. Untuk membuat sebuah objek, kita dapat dengan dengan memanggil nama kelas lalu ditambah dengan tanda kurung ()

```

12 simba_jalan()

```

3. Magic Method

Magic method adalah metode yang diawali dan diakhir dengan double underscore (__), method ini tidak dipanggil secara langsung, tapi dipanggil oleh sistem. Contoh magic method __add__

```

main.py > meong
1 class meong:
2     jumlahSimba=1
3     jumlahSimbi=2
4
5     def __init__(self,simba,simbi):
6         self.simba=simba
7         self.simbi=simbi
8
9     def __add__(self,jalan):
10        return self.simba + jalan.simba
11

```

4. Konstruktor

Konstruktor adalah method yang pasti diajalan secara otomatis pada saat sebuah objek untuk mewakili kelas tersebut. Konstruktor dapat melakukan print,dan menerima argumen

```

main.py > meong
1 class meong:
2     jumlahSimba=1
3     jumlahSimbi=2
4
5     #konstruktor
6     def __init__(self,simba,simbi):
7         self.simba=simba
8         self.simbi=simbi
9
10    #magic method
11    def __add__(self,jalan):
12        return self.simba + jalan.simba
13

```

5. Destruktor

Fungsi yang dipanggil untuk menghapus objek. Fungsi ini bekerja otomatis tanpa perlu pemanggilan

```

main.py > meong
1 class meong:
2     jumlahSimba=1
3     jumlahSimbi=2
4
5     #konstruktor
6     def __init__(self,simba,simbi):
7         self.simba=simba
8         self.simbi=simbi
9
10    #Destructor
11    def __del__(self):
12        print("objek {self.simba} dihapus")
13

```

6. Setter & Getter

Setter & getter berfungsi untuk enkapsulasi agar tidak terjadi perubahan data yang tidak sengaja

```

main.py > meong
1 class meong:
2
3     #konstruktor
4     def __init__(self,simba,simbi):
5         self._simba=simba
6         self.simbi=simbi
7
8     #getter
9     def get_simba(self):
10        return self._simba
11
12    #setter
13    def set_simba(self,a):
14        self._simba=a
15

```

7. Decorator

Berbeda dengan getter dan setter , dengan menggunakan property decorator , kita tidak perlu membuat fungsi lagi dengan nama berbeda, cukup dengan memberikan 1 buah variable.

```

main.py > meong
1 class meong:
2
3     #konstruktor
4     def __init__(self,simba,simbi):
5         self._simba=simba
6         self.simbi=simbi
7
8     @property
9     def get_simba(self):
10        return self._simba
11

```

MODUL 3

ABSTRAKSI DAN ENKAPKULASI

1.Abstraksi

Abstraksi adalah konsep OOP dimana model yang dibuat hanya memperlihatkan atribut yang esensial dan menyembunyikan detail-detail yang tidak penting dari user. gunanya untuk mengurangi kompleksitas.

2.Enkapkulasi

Enkapsulasi adalah sebuah metode yang digunakan untuk mengatur struktur kelas dengan cara menyembunyikan alur kerja dari kelas tersebut,yang dimaksud dengan struktur kelas tersebut adalah property dan method. Terdapat 3 cara untuk menyembunyikan informasi yang tidak dibutuhkan.

2.1. Public Access Modifier

Ketika kita mendeklarasikan suatu variable atau method maka itulah public access modifier, karena setiap varibale atau method secara default itu sudah public access modifier.

```

main.py > meong
1  class meong:
2
3      def __init__(self,simba,simbi):
4          self.simba=simba
5          self.simbi=simbi
6
7
8      def get_simba(self):
9          return self._simba
10

```

2.2. Protected Access Modifier

Ketika method atau variable menggunakan protected, maka variable dan method hanya bisa diakses melalui kelas tersebut. Cara mendeklarasikan nya dengan memberikan 1 underscore sebelum nama var atau method nya (_).

```

main.py > meong > get_simba
1  class meong:
2
3      def __init__(self,simba,simbi):
4          self._simba=simba
5          self.simbi=simbi
6

```

2.3. Private Access Modifier

Ketika var atau method menggunakan private, maka var dan method tersebut hanya bisa diakses dalam kelas itu sendiri. Private access merupakan access modifier yang paling aman, untuk menggunakan private access ,user perlu mendeklarasikan menggunakan double underscore (__).

```

main.py > meong
1  class meong:
2
3      def __init__(self,simba,simbi):
4          self.__simba=simba
5          self.__simbi=simbi
6

```

MODUL 4

Pewarisan dan Polimorfisme

1.Inherintance

Pada inheritance, kita dapat menurunkan kelas dari kelas lain untuk hirarki kelas yang saling berbagi atribut dan metode.


```

main.py > woof
1 class meong:
2
3     def __init__(self,simba,simbi):
4         self.__simba=simba
5         self.__simbi=simbi
6
7
8     def get_simba(self):
9         return self.__simba
10
11
12     def set_simba(self,a):
13         self.__simba=a
14
15 class woof(meong):
16     pass
17

```

1.1. Inheritance Identik

Inheritance identik merupakan pewarisan yang menambahkan constructor pada class child sehingga class child memiliki constructornya sendiri tanpa menghilangkan constructor pada class parentnya.

```

main.py > woof > __init__
1 class meong:
2
3     def __init__(self,simba,simbi):
4         self.__simba=simba
5         self.__simbi=simbi
6
7
8     def get_simba(self):
9         return self.__simba
10
11
12     def set_simba(self,a):
13         self.__simba=a
14
15 class woof(meong):
16     def __init__(self,simba,simbi,yibo):
17         super().__init__(simba,simbi)
18         self.yibo=yibo

```

2. Polymorphism

Polymorphism berarti banyak (poly) dan bentuk (morphism), dalam OOP 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.

```

main.py > ...
1  class gajah:
2      def type(self):
3          print("Jumbo")
4      def warna(self):
5          print("Grey")
6
7  class lion:
8      def type(self):
9          print("Predator")
10     def warna(self):
11         print("Yellow")
12 def func(obj):
13     obj.type()
14     obj.warna()
15
16 obj_gajah = gajah()
17 obj_lion=lion()
18 func(obj_gajah)
19 func(obj_lion)

```

3. Override

Pada konsep OOP di python kita dapat menimpa suatu metode yang ada pada parent class dengan mendefinisikan kembali method dengan nama yang sama pada child class . Dengan begitu maka method yang ada parent class tidak berlaku dan yang akan dijalankan adalah method yang terdapat di child class.

```

main.py > ...
1  class hewan():
2      def tampil(self):
3          print("hewan")
4
5  class kucing(hewan):
6      def tampil(self):
7          print("KUCING")
8  p1=kucing()
9  p1.tampil()

```

4. Casting

4.1. Down Casting

Merupakan parent class mengakses atribut yang ada pada kelas bawah (child class)

```

main.py > ...
1  class meong:
2
3      def __init__(self,simba,simbi):
4          self.simba=simba
5          self.simbi=simbi
6
7      def hewan(self):
8          print(f"{self.simba} {self.simbi} {self.yibo}")
9
10
11
12  class woof(meong):
13      def __init__(self,simba,simbi,yibo):
14          super().__init__(simba,simbi)
15          self.yibo=yibo
16
17  buddy=woof("golden retriver","gede","ganteng")
18  buddy.hewan()

```

4.2. Upcasting

Merupakan child class mengakses atribut yang ada pada kelas atas (parent class)

```

main.py > ...
1  class meong:
2      simbi="gede"
3
4      def __init__(self,simba,simbi):
5          self.simba=simba
6          self.simbi=simbi
7
8      def hewan(self):
9          print(f"{self.simba} {self.simbi} {self.yibo}")
10
11  class woof(meong):
12      def __init__(self,simba,simbi,yibo):
13          super().__init__(simba,simbi)
14          self.yibo=yibo
15
16      def hewan(self):
17          print(f"{self.simba} {self.simbi} {self.yibo}")
18
19  buddy=woof("golden retriver","gede","ganteng")
20  buddy.hewan()

```

5. Multiple Inheritance

Kelas dapat mewarisi dari banyak orang tua, contoh :

```

main.py > mbek
1  class meong:
2      pass
3
4  class woof:
5      pass
6
7  class mbek(meong,woof):
8      pass

```

