1. **Method**

Method adalah sebuah blok program terpisah (diluar program utama) yang kita gunakan untuk menyelesaikan masalah khusus.

Berikut adalah struktur dari sebuah method:

|  |
| --- |
| modifier returnType namaMethod (Parameter List) {  // method body  } |

Penjelasan singkat:

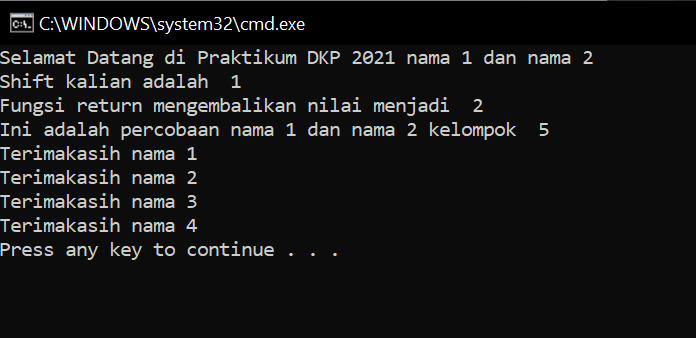
* modifier - Untuk menentukan hak akses terhadap method.
* returnType - Ada dua jenis method yaitu: method void dan method return.
* namaMethod - Ini adalah nama method. Sesuai keinginan anda.
* Parameter - Untuk meng-oper sebuah nilai. Bersifat opsional.

(Sumber: https://kodedasar.com/method-java/)

1. **Function pada Python**

*Source Code:*

|  |
| --- |
| #Function dengan non return type  def non\_return\_func(praktikan1, praktikan2):  print(f"Selamat Datang di Praktikum DKP 2021 {praktikan1} dan {praktikan2}")  #Function dengan return type  def return\_func(shift):  print(f"Shift kalian adalah ", shift)  if (shift == 1) or (shift == 2) :  return print(f"Fungsi return mengembalikan nilai menjadi ", shift \* 2)  else:  return print("Tidak ada shift tersebut")  #Function dengan Arbitrary Type  def arbitrary\_func(\*penutup):  for nama in penutup:  print("Terimakasih", nama)  #Anonymous Function  anonim\_func = lambda praktikan1, praktikan2, kelompok: print(f"Ini adalah percobaan {praktikan1} dan {praktikan2} kelompok ", kelompok )  #Pemanggilan Fungsi  non\_return\_func("nama 1", "nama 2")  return\_func(1)  anonim\_func("nama 1", "nama 2", 5)  arbitrary\_func("nama 1", "nama 2", "nama 3", "nama 4") |

****

Gambar 5.3.5.1 Output Function pada Python

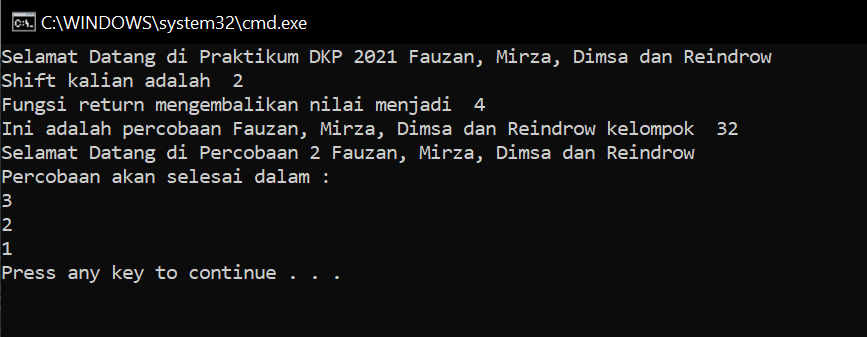
Analisis:

Pertama deklarasikan fungsi jenis *Non-Return Type* atau *Void Type* dengan *variable* “praktikan1” dan “praktikan2”. Kemudian dibaris selanjutnya, terdapat print dengan f-string yang akan mencetak "Selamat Datang di Praktikum DKP 2021 {praktikan1} dan {praktikan2}". Lalu pendeklarasian fungsi jenis *Return Type* dengan *variable* *shift*. Dibaris selanjutnya terdapat print dengan f-string yang akan mencetak *shift* berapa para praktikan. Setelah itu, deklarasikan pengkondisian *if-else* dengan *if* jika pada return\_func() adalah 1 atau 2, maka *return print* akan mencetak “Shift kalian adalah 1” atau “Shift kalian adalah 2” dan Fungsi return mengembalikan nilai menjadi 2” atau “Fungsi return mengembalikan nilai menjadi 4”. Dan jika *else* pada return\_func() adalah selain 1 dan 2, maka *return* *print* akan mencetak “Tidak ada shift tersebut”. Selanjutnya fungsi dengan *Arbitrary Type* untuk menyebut jumlah *named* argumen fungsi agar tidak berubah-ubah. Lalu fungsi *Anonymous* atau sering disebut *lambda* *function,* dan lambda disini merupakan ekspresi. Selanjutnya Pemanggilan Fungsi untuk memanggil semua jenis fungsi *non-return type*, *return function*, *anonim function* dan *arbitrary function* dari program tadi agar tercetak pada *output.*

1. **Method pada Python**

*Source Code Function:*

|  |
| --- |
| import percobaan2  #Function dengan non return type  def non\_return\_func(praktikan1, praktikan2, praktikan3, praktikan4):  print(f"Selamat Datang di Praktikum DKP 2021 {praktikan1}, {praktikan2}, {praktikan3} dan {praktikan4}")  #Function dengan return type  def return\_func(shift):  print(f"Shift kalian adalah ", shift)  if (shift == 1) or (shift == 2) :  return print(f"Fungsi return mengembalikan nilai menjadi ", shift \* 2)  else:  return print("Tidak ada shift tersebut")  #Function dengan Arbitrary Type  def arbitrary\_func(\*penutup):  for nama in penutup:  print("Terimakasih", nama)  #Anonymous Function  anonim\_func = lambda praktikan1, praktikan2, praktikan3, praktikan4, kelompok: print(f"Ini adalah percobaan {praktikan1}, {praktikan2}, {praktikan3} dan {praktikan4} kelompok ", kelompok )  #Pemanggilan Fungsi  non\_return\_func("Fauzan", "Mirza", "Dimsa", "Reindrow")  return\_func(2)  anonim\_func("Fauzan", "Mirza", "Dimsa", "Reindrow",32)  #membuat object  p1=percobaan2.contoh\_method("Fauzan","Mirza","Dimsa","Reindrow")  #memanggil method dengan self parameter  p1.mulai()  #memanggil method dengan parameter  p1.selesai(3)  #arbitrary\_func("nama 1", "nama2", "nama 3", "nama 4") |

*Source Code Method:*

|  |
| --- |
| class contoh\_method:  #init method  def \_\_init\_\_(self, praktikan1, praktikan2, praktikan3, praktikan4):  self.praktikan1 = praktikan1  self.praktikan2 = praktikan2  self.praktikan3 = praktikan3  self.praktikan4 = praktikan4  #self parameter  def mulai(self):  print(f"Selamat Datang di Percobaan 2 {self.praktikan1}, {self.praktikan2}, {self.praktikan3} dan {self.praktikan4}")  #method dengan parameter  def selesai(self, waktu):  print("Percobaan akan selesai dalam :")  while waktu > 0:  print(waktu)  waktu -= 1 |

Gambar 5.3.6.1 Output Percobaan dengan Method pada Python

Analisis:

Pertama add new item, lalu membuat class baru untuk menampung *method*. Lalu tuliskan *source* *code* dari *function* tetapi tambahkan beberapa kode. Pada *method*, terdapat method init() yang berfungsi untuk mengeksekusi secara otomatis ketika class dari method dipanggil sebuah objek. Kemudian *self* *parameter* ini diharuskan sebagai parameter pertama dari program *method.* Lalu method dengan parameter digunakan untuk menghitung waktu percobaan akan selesai dengan jumlah waktu yang ditentukan.