# **BAB V**

# **FUNCTION DAN METHOD**

## **5.1. Tujuan**

1. Mengenal dan mengetahui perbedaan penggunaan beberapa *IDE* atau aplikasi pemrograman dan beberapa jenis bahasa pemrograman.
2. Memahami pengertian dan konsep dari *Function* dan *Method*.
3. Mengetahui penggunaan *Function* dan *Method* dengan bahasa pemrograman Python, C#, Java, dan PHP.
4. Mengetahui perbedaan *Function* dan *Method* dalam bahasa pemrograman Python, C#, Java, dan PHP.
5. Mampu mengimplementasikan *Function* dan *Method* dalam bahasa pemrograman Python, C#, Java, dan PHP.
6. Memahami pengertian dan konsep penggunaan dari *Github*.

## **Dasar Teori**

### ***Function***

*Function* adalah sebuah struktur, pengelompokkan yang mengandung sekelompok pernyataan yang akan dilaksanakan oleh program. Setiap program pasti memiliki satu function utama, misalnya *int main()*. *Function* tersebut merupakan *function* utama yang menjadi kepala dari program dimana semua eksekusi pernyataan berawal.

*Function* berisi sejumlah pernyataan yang dikemas dalam sebuah nama yang selanjutnya dapat dipanggil beberapa kali dibeberapa tempat dalam program. Tujuannya adalah untuk memudahkan dalam mengembangkan program dan menghemat ukuran program. Setiap fungsi dari bahasa pemograman berbeda cara pemanggilannya misalnya, *fungsions, methods, sub-routines, procedurs*, dll.

Secara umum ada 2 tipe *function*, yaitu:

1. Mengembalikan nilai (*Return Type* atau *Non-Void Type*)

Fungsi ini mengembalikan suatu nilai baru dari proses yang dilakukannya. Ciri utama dari fungsi ini yaitu terdapat *keyword* *return* di dalam *block code*-nya. Nilai yang dikembalikan oleh *return-type function* akan sesuai dengan tipe data yang digunakan pada saat mendeklarasikan fungsi tersebut.

2. Tidak mengembalikan nilai (*Non-Return Type* atau *Void Type*)

Dideklarasikan dengan tipe data *void* yang berarti fungsi ini tidak akan mengembalikan suatu nilai baru setelah pekerjaannya selesai dilakukan sehingga fungsi tipe *void* tidak memerlukan *keyword return* dalam penggunaannya.

Sumber: belajarcpp.com

### ***Method***

*Method* adalah sebuah aksi yang terdapat di dalam *class*. *Method* biasa digunakan dalam dalam Pemograman Berorientasi Objek. Penulisan *method* pada *class* adalah dengan menuliskan *syntax function* di awalnya, lalu diikuti dengan nama method tersebut. Kita dapat memberikan nilai pada parameter *method* dan juga dapat mengembalikan nilainya.

Sumber: malasngoding.com

### **Penggunaan *Function***

1. Variabel Memiliki Ruang Lingkup

Variabel adalah sebuah wadah untuk menyimpan nilai dengan mereferensikannya dengan sebuah nama yang didefinisikan. Variabel memiliki “kawasan” yang bisa mengaksesnya, ini disebut Ruang Lingkup. Menciptakan sebuah variabel didalam fungsi X tidak menjadikan anda dapat mengakses variabel tersebut dari fungsi Y.

1. Parameter atau Argumen

Dalam konsep fungsi dikenal istilah *argument* atau parameter. Parameter adalah variabel yang menyimpan nilai untuk diproses di dalam fungsi. Gunanya adalah sebagai cara untuk mengirim nilai ke fungsi yang dituju. Dalam penamannya, parameter boleh sama dengan nama variabel yang didalam fungsi *main*.

1. Tipe data

Mendefinisikan jenis fungsi (*return/non-return*) dan tipe data nilai kembalian dari fungsi tersebut (jika *return type*).

1. Ruang lingkup penggunaan

Ruang lingkup atau disebut *access modifier*/*access specifier* digunakan pada PBO dimana akan menentukan *object*/*class* mana saja yang dapat menggunakan *variable* maupun *function/method* yang berada di *class* tersebut.

Dikenal 3 jenis *modifier*:

1. *Public*

*Modifier public* mengindikasikan bahwa *class*/*variable*/ *function*/*method* dengan *modifier* ini akan dapat digunakan oleh semua *class* yang ada

1. *Private*

*Modifier private* mengindikasikan bahwa *class*/*variable*/ *function*/*method* dengan *modifier* ini akan dapat digunakan hanya oleh *class* itu sendiri (*class* dimana *variable*/*method* tersebut berada)

1. *Protected*

*Modifier protected* mengindikasikan bahwa *class*/*variable*/ *function*/*method* dengan *modifier* ini akan dapat digunakan oleh *class* itu sendiri dan oleh kelas turunannya (*sub-class*)

Suatu *variable* atau fungsi yang dideklarasikan tanpa menuliskan *modifier* maka secara otomatis memiliki *modifier* *private*.

Sumber: medium.com, petanikode.com, Modul 4

### **Penggunaan *Method***

Saat anda ingin mencetak sesuatu ke *console* maka anda akan menggunakan sintaks *System.out.println();.* Nah *println();* merupakan salah satu *method* pada java. Cara menggunakan atau membuat *method* adalah sebagai berikut:

1. *Modifier*, berfungsi ntuk menentukan hak akses terhadap *method*.
2. *ReturnType*, ada dua jenis *method* yaitu: *method void* dan *method return*.
3. NamaMethod, ini adalah nama *method*. Sesuai keinginan anda.
4. Parameter, berfungsi untuk meng-oper sebuah nilai. Bersifat opsional.

Sumber: kodedasar.com

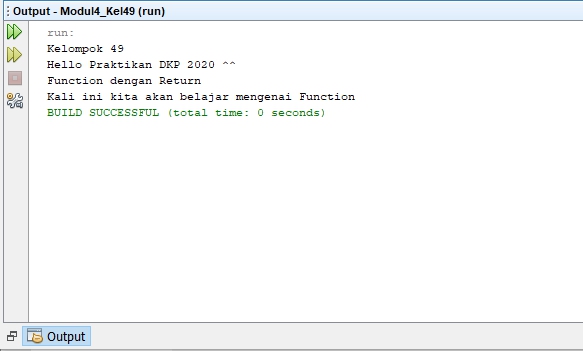
## **Hasil Percobaan dan Analisis**

### ***Function* dalam Java**

|  |
| --- |
| public class Modul4\_Kel49 {  // non-return type dengan parameter  static void cetak(String kelompok){  System.out.println(kelompok);  }  // non-return type tanpa parameter  static void cetak1(){  System.out.println("Hello Praktikan DKP 2020 ^^ ");  }  // return type dengan parameter  static String cetak2(String text){  return text;  }  // return type tanpa parameter  static String cetak3(){  return "Kali ini kita akan belajar mengenai Function";  }  public static void main(String[] args) {  //penggunaan function non-return type dengan megisikan argument untuk parameter kelompok  cetak("Kelompok 49");  //penggunaan function non-return type tanpa parameter  cetak1();  //penggunaan function return type dengan mengisikan nilai pada parameter text  System.out.println(cetak2("Function dengan Return"));  //penggunaan function return type tanpa parameter  System.out.println(cetak3());  }  } |

Pertama tama kita buat public class(bukan *void*) lalu didalam public class ini kita dapat membuat *function* yang diinginkan, disini kita membuat 2 jenis *function* yaitu function non return dan return dengan *function* tersebut diberi nama yang berbeda beda. Dalam pembuatan *function* kita awali dengan jenis *function* yaitu void (bila non *return*) atau static *+* data type (bila *return*) dan diikuti dengan nama *function.* Selanjutnya kita membuat kelas kita yaitu public class void, kelas ini kita masukkan ke *class* yang diatasnya. Selanjutnya didalam class void ini kita panggil nama *function* di *class* sebelumnya dan memberi variabel masukan bila *function* tersebut memerlukan variabel masukan.

*Output* Hasil:



Gambar 5. 1. Output Function dalam Java

### ***Method* dalam Java**

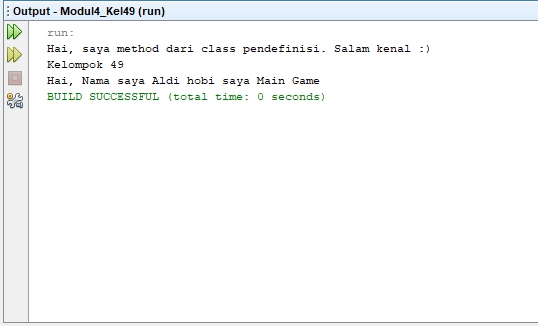
***Class* Pendefinisi *Method***

|  |
| --- |
| public class pendefinisi {  public void greeting(){  System.out.println("Hai, saya method dari class pendefinisi. Salam kenal :)");  }  public void kelompok(String kelompok){  System.out.println(kelompok);  }  public String kenalan (String nama, String hobi){  return "Hai, Nama saya " + nama + " hobi saya " + hobi;  }  } |

***Class* Pemanggil *Method***

|  |
| --- |
| public static void main(String[] args) {  pendefinisi objek = new pendefinisi();    objek.greeting();  objek.kelompok("Kelompok 49");  String print = objek.kenalan("Aldi", "Main Game");  System.out.println(print);  } |

Pertama tama kita buat *class* pada *file* yang berbeda, setelah kita buat *class* lalu kita dapat membuat *method* yang diinginkan. disini kita membuat 2 jenis *method* yaitu method non return dan return dengan tersebut diberi nama yang berbeda beda. Dalam pembuatan *method* kita awali dengan jenis *method* yaitu *void* (bila *non return*) atau *static+data type* (bila *return*) dan diikuti dengan nama *method*. Setelah itu pada *file* utama kita panggil *class* pendefinisi di *file* sebelah, setelah kita panggil *class*-nya kita baru bisa memanggil *method* yang berada pada *class* sebelah. Kita panggil *method* di kelas sebelah dengan menulis variabel yang diperlukan, bila tidak ada variabel yang diperlukan maka dapat kita kosongi.

*****Output* Hasil:

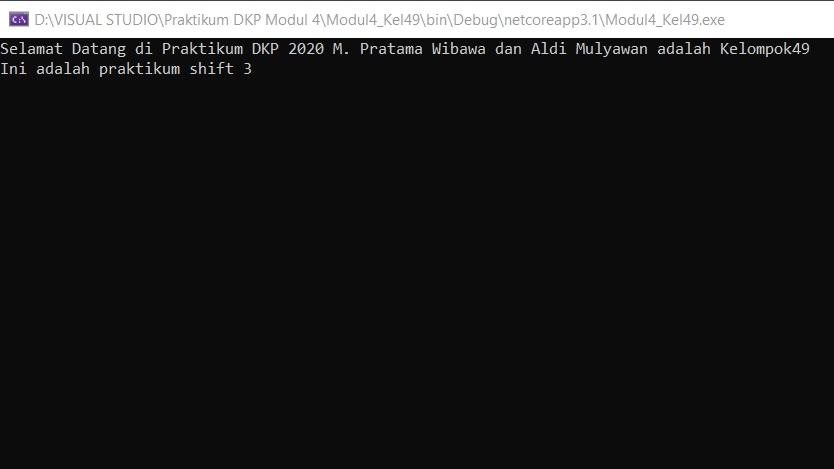
Gambar 5. 2. Output Method dalam Java

### ***Function* dalam C#**

|  |
| --- |
| class Program  {  static void non\_return(String a, String b, String c)  {  Console.WriteLine("Selamat Datang di Praktikum DKP 2020 {0} dan {1} adalah {2}", a, b, c);  }  static int return\_func(int a)  {  if (a > 0 && a < 3) {  return a \* 3;  } else  {  return a \* 0;  }    }  static void Main(string[] args)  {  non\_return("M. Pratama Wibawa", "Aldi Mulyawan", "Kelompok49");  Console.WriteLine("Ini adalah praktikum shift {0}", return\_func(1));  Console.ReadKey();  }  } |

Pertama tama kita buat *public class* (bukan *void*) lalu didalam *public class* ini kita dapat membuat *function* yang diinginkan, disini kita membuat 2 jenis *function* yaitu function non return dan return dengan *function* tersebut diberi nama yang berbeda beda. Dalam pembuatan *function* kita awali dengan jenis *function* yaitu *void*(bila non *return*) atau *static+data type* (bila *return*) dan diikuti dengan nama *function.* Selanjutnya kita membuat kelas kita yaitu public class void, kelas ini kita masukkan ke *class* yang diatasnya. Selanjutnya didalam *class void* ini kita panggil nama *function* di *class* sebelumnya dan memberi variabel masukan bila *function* tersebut memerlukan variabel masukan.

*Output* Hasil:

****

Gambar 5. 3. Output Function dalam C#

### ***Method* dalam C#**

***Class* Modul\_kel49**

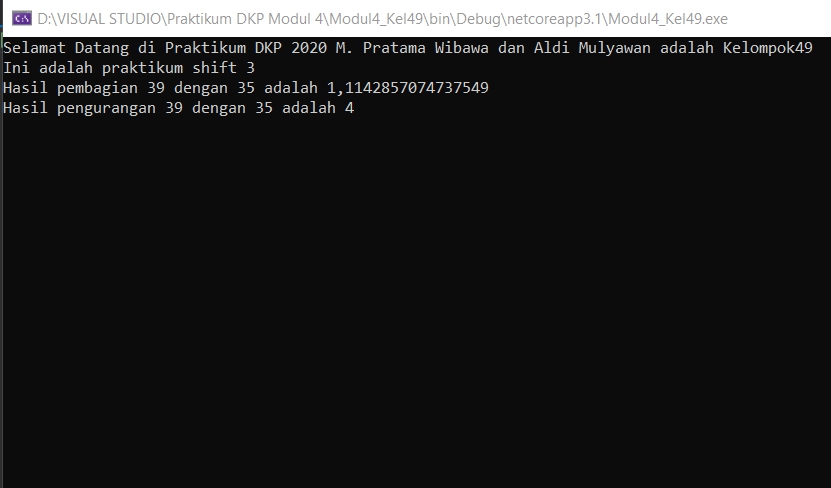
|  |
| --- |
| class Program  {  static void non\_return(String a, String b, String c)  {  Console.WriteLine("Selamat Datang di Praktikum DKP 2020 {0} dan {1} adalah {2}", a, b, c);  }  static int return\_func(int a)  {  if (a > 0 && a < 3) {  return a \* 3;  } else  {  return a \* 0;  }    }  static void Main(string[] args)  {  non\_return("M. Pratama Wibawa", "Aldi Mulyawan", "Kelompok49");  Console.WriteLine("Ini adalah praktikum shift {0}", return\_func(1));  percobaan2 objek = new percobaan2();  objek.pembagian(39, 35);  Console.WriteLine("Hasil pengurangan {0} dengan {1} adalah {2}", 39, 35, perkalian(39, 35));  Console.ReadKey();  }  } |

***Class* Percobaan2**

|  |
| --- |
| class percobaan2  {  public void pembagian(float a, float b)  {  double c = a / b;  Console.WriteLine("Hasil pembagian {0} dengan {1} adalah {2}", a, b, c);  }  public int pengurangan(int a, int b)  {  int c = a - b;  return c;  }  } |

Pertama tama kita buat *class* pada *file* yang berbeda, setelah kita buat *class* lalu kita dapat membuat *method* yang diinginkan. disini kita membuat 2 jenis *method* yaitu method non return dan return dengan tersebut diberi nama yang berbeda beda. Dalam pembuatan *method* kita awali dengan jenis *method* yaitu *void*(bila *non return*) atau *static+data type* (bila *return*) dan diikuti dengan nama *method*. Setelah itu pada *file* utama kita panggil *class* pendefinisi di *file* sebelah, setelah kita panggil *class*-nya kita baru bisa memanggil *method* yang berada pada *class* sebelah. Kita panggil *method* di kelas sebelah dengan menulis variabel yang diperlukan, bila tidak ada variabel yang diperlukan maka dapat kita kosongkan.

*Output* Hasil:



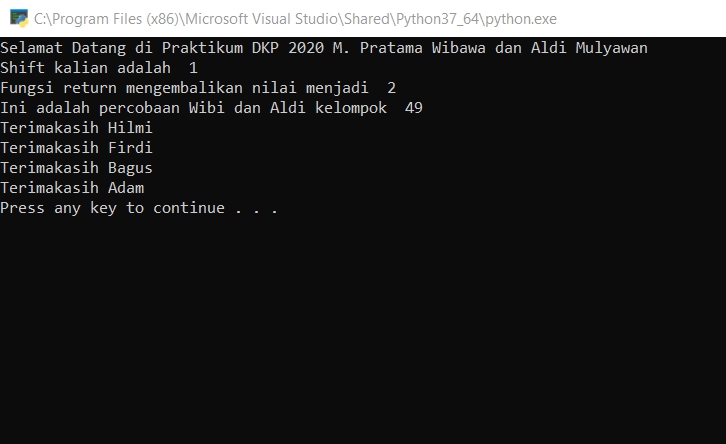
Gambar 5. 4. Output Method dalam C#

### ***Function* dalam Phyton**

|  |
| --- |
| import percobaan2  #Function dengan non return type  def non\_return\_func(praktikan1, praktikan2):  print(f"Selamat Datang di Praktikum DKP 2020 {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("M. Pratama Wibawa", "Aldi Mulyawan")  return\_func(1)  anonim\_func("Wibi", "Aldi", 49)  arbitrary\_func("Hilmi", "Firdi", "Bagus", "Adam") |

Pada *python* pertama tama kita langsung membuat *function* yang kita inginkan, kita awali dengan defdan diikuti dengan nama *function*. Perbedaan dari *function* *return* dan *non return* pada *python* adalah bila *function return* maka diakhir *function* akan ada *code “*return*”,* untuk *non return* tidak ada kata *return* karena tidak ada yang dikembalikan. Di *python* juga ada *arbitary function* yang memiliki format penulisan yang sama dengan *function return* dan *non return.* *Python* juga memiliki *Anonim function* yang memiliki format nama variabel lalu diikuti dengan sama dengan (=) dan *lambda.* Selanjutnya kita panggil *function* diatas dengan menulis *code* nama variabel dan diikuti variabel yang diperlukan.

*Output* Hasil:

****

Gambar 5. 5. Output Function dalam Phyton

### ***Method* dalam Phyton**

***Class* Pemanggil *Method***

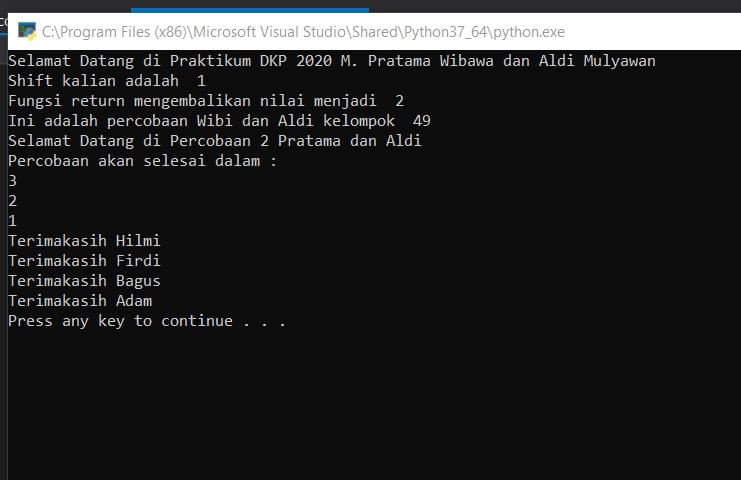
|  |
| --- |
| import percobaan2  #Function dengan non return type  def non\_return\_func(praktikan1, praktikan2):  print(f"Selamat Datang di Praktikum DKP 2020 {praktikan1} dan {praktikan2}")  #Function dengan return type  def return\_func(shift):  print(f"Shift kalian adalah ", shift)  if (shift == 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("M. Pratama Wibawa", "Aldi Mulyawan")  return\_func(1)  anonim\_func("Wibi", "Aldi", 49)  arbitrary\_func("Hilmi", "Firdi", "Bagus", "Adam")  #pembuatan object untuk mengggunakan method  p1 = percobaan2.contoh\_method("Pratama", "Aldi")  #pemanggilan method dengan self parameter  p1.mulai()  #pemanggilan method dengan parameter  p1.selesai(3) |

***Class* Pendefinisi Method**

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

Pertama tama kita buat *class* pada *file* yang berbeda, setelah kita buat *class* lalu kita dapat membuat *method* yang diinginkan. Dalam *method* di python ini kita harus memasukkan *method* init untuk inisialisasi variabel apa saja yang akan digunakan pada *method*-*method* dibawahnya. Pembuatan *method* pada python sama halnya dengan pembuatan *function* yaitu penambahan def diawalan. Selanjutnya pada *file* utama kita panggil method di *file* sebelah dengan memanggil nama *filenya* dulu dan selanjutnya kita buat variabel untuk mewakili kelas yang berisi *method*. Selanjutnya kita dapat memanggil *method* yang kita inginkan dengan menulis variabel diikuti nama *method* dan variabel *input* bila diperlukan.

*Output* Hasil:



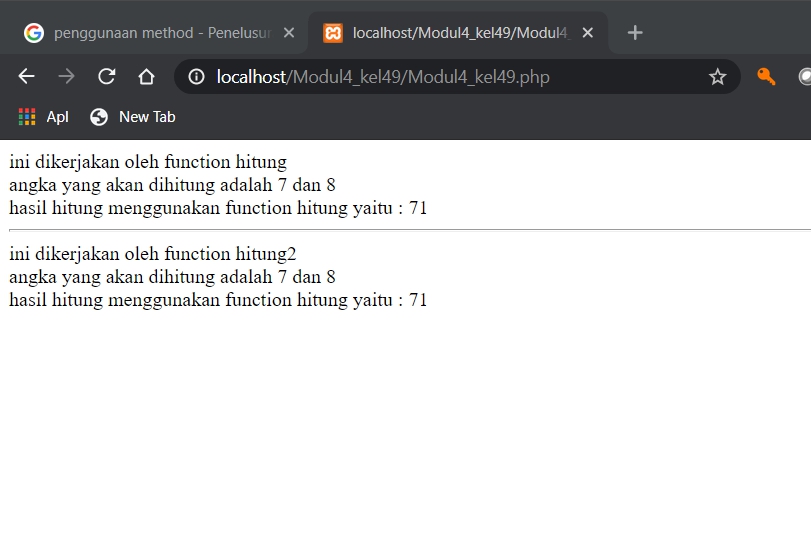
Gambar 5. 6. Output Method dalam Phyton

### ***Function* dalam PHP**

|  |
| --- |
| <?php  //ini return type  function hitung ($bil1, $bil2) {  return $bil1 + $bil2 \* $bil2;  }  $bil1 = 7;  $bil2 = 8;  $hasil = hitung($bil1, $bil2);  echo "ini dikerjakan oleh function hitung <br>";  echo "angka yang akan dihitung adalah $bil1 dan $bil2 <br>";  echo "hasil hitung menggunakan function hitung yaitu : $hasil";    echo "<hr>";  //ini non return type  function hitung2 ($bil1, $bil2) {  $hasil = hitung($bil1, $bil2);  echo "angka yang akan dihitung adalah $bil1 dan $bil2 <br>";  echo "hasil hitung menggunakan function hitung yaitu : $hasil";  }  echo "ini dikerjakan oleh function hitung2 <br>";  hitung2($bil1, $bil2);  ?> |

Pada PHP pertama tama kita langsung membuat *function* yang kita inginkan, kita awali dengan *function* dan diikuti dengan nama *function*. Perbedaan dari *function return* dan *non return* pada PHP adalah bila *function return* maka diakhir *function* akan ada *code “*return*”,* untuk *non return* tidak ada kata *return* karena tidak ada yang dikembalikan. Selanjutnya kita dapat langsung memanggil *function* yang telah kita buat dengan mengetik nama *function.* Akhirnya *function* akan mengeluarkan hasil yang diinginkan.

*Output* Hasil:

****

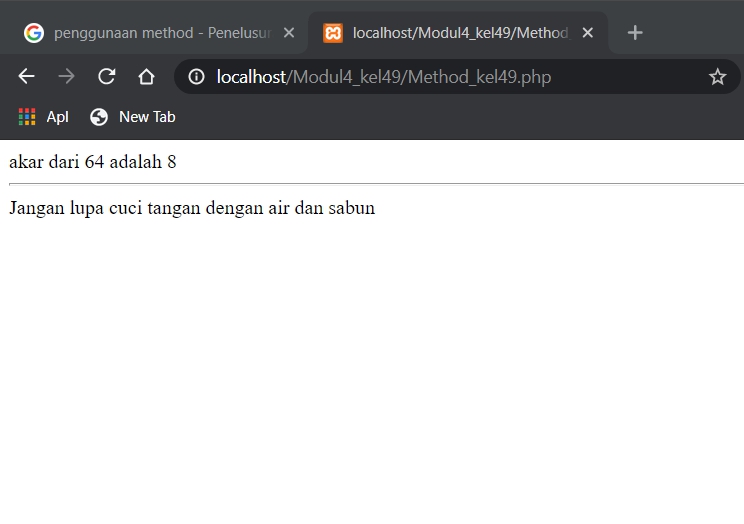
Gambar 5. 7. Output Function dalam PHP

### ***Method* dalam PHP**

|  |
| --- |
| <?php  declare(strict\_types = 1);  //class pendefinisi method  class pendefinisi {  public function akar(float $angka) : float {  return sqrt($angka);  }  public function staysafe (string $noun1, string $noun2) {  echo "Jangan lupa cuci tangan dengan $noun1 dan $noun2";  }  }  //code pemanggil method  $angka = 64;  //$objek merupakan instance dari class pendefinisi  $objek = new pendefinisi();    // OOP pada PHP menggunakan -> bukan .  echo "akar dari $angka adalah ".$objek -> akar($angka);  echo "<hr>";  $objek -> staysafe('air', 'sabun');    ?> |

Pertama tama kita buat *class* pada *file* yang berbeda, setelah kita buat *class* lalu kita dapat membuat *method* yang diinginkan. Pembuatan *Method* pada PHP pertama tama kita awali dengan public function dan kita lanjutkan dengan nama *method*.Selanjutnya pada *file* utama kita panggil *method* di *file* sebelah dengan memanggil nama *method* tetapi sebelumnya harus diketahui dulu variabel *input* yang diperlukan (Bila ada). Akhirnya *method* akan mengeluarkan hasil yang diinginkan.

*Output* Hasil:

****

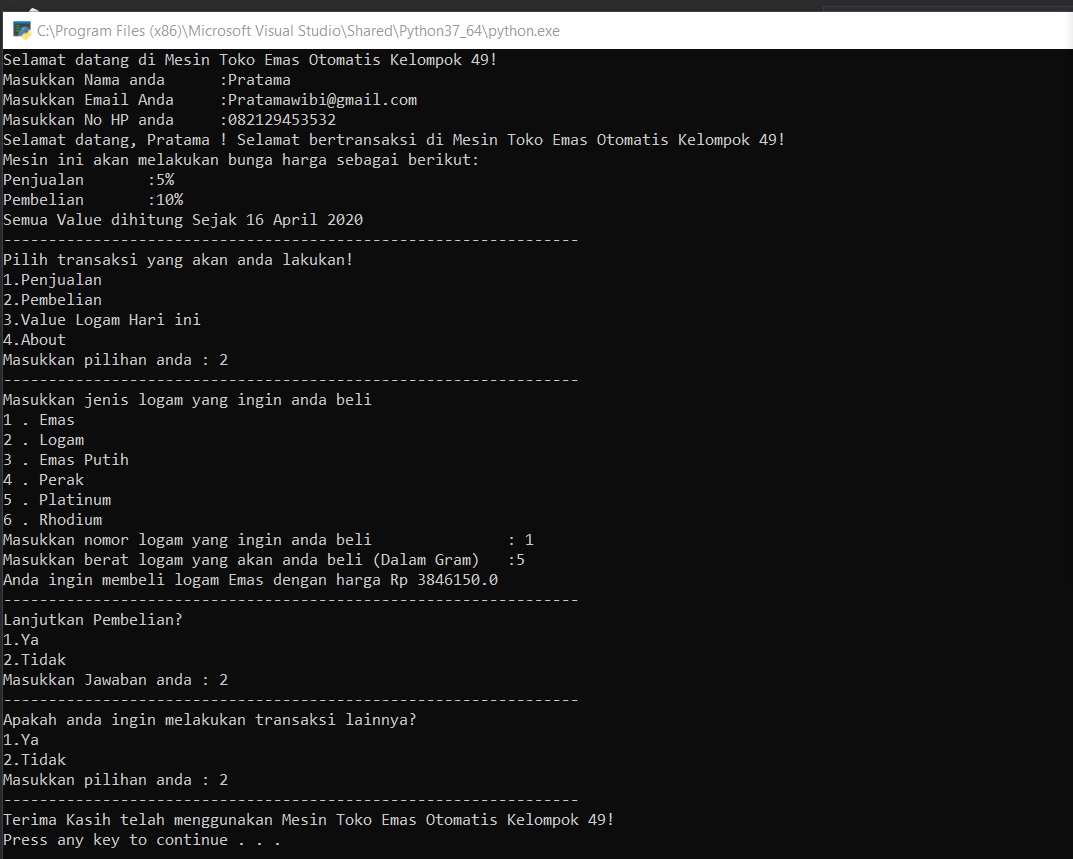
Gambar 5. 8. Output Method dalam PHP

## **Tugas**

|  |
| --- |
| import Ext\_Source\_Code  def creator () :  print("Program ini dibuat oleh kelompok 49 Shift 2")  print("Aldi Mulyawan 21120119120026")  print("Mochamad Pratama Wibawa 21120119130062")  print("Selamat datang di Mesin Toko Emas Otomatis Kelompok 49!")  nama = str(input("Masukkan Nama anda\t:"))  email = str(input("Masukkan Email Anda\t:"))  HP = str(input("Masukkan No HP anda\t:"))  call = Ext\_Source\_Code.main(nama,email,HP)  call.welcome()  print("Semua Value dihitung Sejak 16 April 2020")  print("----------------------------------------------------------------")  choice = 1  while (choice == 1):  print("Pilih transaksi yang akan anda lakukan!")  print("1.Penjualan")  print("2.Pembelian")  print("3.Value Logam Hari ini ")  print("4.About")  a = int(input("Masukkan pilihan anda : "))  print("----------------------------------------------------------------")  if (a == 1):  call.penjualan\_fitur()  b = int(input("Masukkan nomor logam yang ingin anda beli\t\t: "))  c = int(input("Masukkan berat logam yang akan anda beli (Dalam Gram)\t:"))  print("Masukkan Logam ke kotak dibawah.....")  print("Loading......")  print("Scan berhasil,Data benar.")  call.penjualan\_transaksi(b, c)  e = int(input("Masukkan Jawaban anda : "))  if (e == 1):  print("Terima Kasih telah bertransaksi menggunakan Mesin Toko Emas Otomatis Kelompok 49!")  print("Detail transaksi akan dikirimkan ke email ",email," serta no HP ",HP)  print("----------------------------------------------------------------")  elif (a == 2):  call.pembelian\_fitur()  b = int(input("Masukkan nomor logam yang ingin anda beli\t\t: "))  c = int(input("Masukkan berat logam yang akan anda beli (Dalam Gram)\t:"))  call.pembelian\_transaksi(b, c)  e = int(input("Masukkan Jawaban anda : "))  if (e == 1):  print("Terima Kasih telah bertransaksi menggunakan Mesin Toko Emas Otomatis Kelompok 49!")  print("Detail transaksi akan dikirimkan ke email ",email," serta no HP ",HP)  print("----------------------------------------------------------------")  elif (a == 3):  call.about\_fitur()  elif (a == 4):  creator()  else:  print("Mohon masukkan angka yang sesuai!")  print("Apakah anda ingin melakukan transaksi lainnya?")  print("1.Ya")  print("2.Tidak")  choice = int(input("Masukkan pilihan anda : "))  print("----------------------------------------------------------------")  print("Terima Kasih telah menggunakan Mesin Toko Emas Otomatis Kelompok 49!") |

|  |
| --- |
| class main:  def \_\_init\_\_ (self, nama, email, HP):  self.nama=nama  self.email=email  self.HP=HP  self.logam=["Emas","Logam","Emas Putih","Perak","Platinum","Rhodium"]  self.harga=[732600,216044,361000,396000,385872,1081241]  def welcome (self):  print(f"Selamat datang, {self.nama} ! Selamat bertransaksi di Mesin Toko Emas Otomatis Kelompok 49!")  print("Mesin ini akan melakukan bunga harga sebagai berikut:")  print("Penjualan\t:5%")  print("Pembelian\t:10%")  def pembelian\_fitur (self):  print("Masukkan jenis logam yang ingin anda beli")  count = 1  for x in self.logam :  print(count,".",x)  count = count + 1  def penjualan\_fitur (self):  print("Masukkan jenis logam yang ingin anda jual")  count = 1  for x in self.logam :  print(count,".",x)  count = count + 1  def about\_fitur (self):  print("Berikut adalah Value Logam pada hari ini (per gram):")  count = 1  for x in self.logam :  print(count,".",x,"\t: Rp ",self.harga[count - 1])  count = count + 1  def pembelian\_transaksi (self,angka1,angka2):  biaya = self.harga[angka1 - 1] \* angka2 \* 105 / 100  print(f"Anda ingin membeli logam {self.logam[angka1-1]} dengan harga Rp {biaya}")  print("----------------------------------------------------------------")  print("Lanjutkan Pembelian?")  print("1.Ya")  print("2.Tidak")  def penjualan\_transaksi (self,angka1,angka2):  biaya = self.harga[angka1 - 1] \* angka2 \* 90 / 100  print(f"Anda ingin menjual logam {self.logam[angka1-1]} dengan harga Rp {biaya}")  print("----------------------------------------------------------------")  print("Lanjutkan Pembelian?")  print("1.Ya")  print("2.Tidak") |

*Output* Hasil:



Gambar 5. 9. Output Tugas Modul 4

Penjelasan *Output* :

Pertama tama kita buat *file python* baru selain *file* utama dan kita namai Ext\_Source\_Code. Selanjutnya kita buat method inisialisasi, disini kita menggunakan 5 variabel inisialisasi dengan 3 variabel yang harus diisi.selanjutnya kita membuat *method* welcome, daftar barang untuk pembelian dan penjualan, logam yang tersedia, dan yang terakhir transaksi pembelian dan penjualan. Selanjutnya kita kembali ke *file* utama,dan pertama tama *import file* sebelumnya. Lalu kita buat *function* info creator, selanjutnya kita buat *inputan* untuk nama,no hp,dan email supaya *method* dapat dijalankan. Setelah variabel memiliki nilai maka kita dapat memanggil *method* dan kita wakili dengan variabel call. Setelah itu kita panggil *method* welcome dan kita buat while supaya transaksi dapat terus berulang. Setelah itu kita buat daftar transaksi yang tersedia dan beri *inputan* supaya pembeli dapat memasukkan fitur yang diinginkan. lalu kita buat if elif dan pemanggilan *method* yang bersangkutan serta else bila *inputan* tidak sesuai. Selanjutnya kita lihat di pembelian dan penjualan, kita panggil *method* penjualan\_transaksi atau pembelian\_transaksi supaya transaksi dapat dihitung dan kita beri pilihan apakah pembeli akan melanjutkan transaksinya. Setelah selesai maka pembeli akan diberi pilihan apakah akan melakukan transaksi lainnya bila iya maka akan diulang dari pemilihan fitur yang tersedia. Bila pembeli menyudahi transaksi maka keluarkan *output* terima kasih telah bertransaksi.

Link Tugas : <https://github.com/SMAMHTN/Praktikum_DKP_Kel49/tree/master/Tugas%20Modul%204%20Kelompok%2049/PythonApplication1>

## **Kesimpulan**

* + 1. *Function* dan *Method* memiliki hubungan yang saling berkaitan.
    2. Penggunaan *Function* dan *Method* betujuan agar menghemat ukuran program dan memudahkan dalam mengembangkannya.
    3. Penggunaan *Function* atau *Method* dipilih berdasarkan kebutuhan dan urgensinya masing-masing.
    4. Bahasa pemrograman yang keluar dalam waktu dekat, akan memiliki kemampuan yang lebih cepat dan *simple* dari pendahulunya. Seperti phyton lebih *simple* dari C#.
    5. Pemograman mempunyai tiga inti dasar yaitu *input*, proses, dan *output*.
    6. Antara bahasa pemrograman satu dengan yang lainnya kadang terdapat perbedaan-perbendaan tertentu, karena memiliki aturannya masing-masing.
    7. Bahasa pemrogaman tidak dapat dibanding-bandingkan, setiap bahasa pemograman bisa dipakai sesuai kebutuhannya masing-masing.
    8. Diperlukan ketelitian dalam mebuat sebuah program, karena kesalahan sedikit akan membuat program tidak berjalan dengan baik.