
MODUL 4

FUNCTION DAN METHOD

5.1. Tujuan

1. Praktikan dapat memahami konsep dasar function dan method.
2. Praktikan dapat memahami perbedaan function dan method.
3. Praktikan dapat memahami fungsi dari function dan method.
4. Praktikan dapat mengimplementasikan function dan method dalam bahasa pemrograman C++, Python, Java dan PHP.

5.2. Dasar Teori

5.2.1. Function

Function merupakan sekumpulan instruksi yang dikelompokkan secara khusus untuk mengerjakan suatu tugas tertentu. Suatu fungsi dapat dikatakan sebagai min-program dari suatu program utuh yang letaknya dipisahkan (berbeda blok) dari bagian program yang dijalankan (program utama) dan dapat digunakan secara berulang-ulang.

Dalam pemrograman suatu fungsi diperlukan untuk menghindari penulisan baris kode yang sama secara berulang-ulang sehingga dapat meningkatkan efisiensi dalam pengerjaan suatu program.

Menggunakan fungsi berarti memecah program ke dalam blok-blok yang lebih kecil sehingga program akan terlihat lebih rapi dan mudah dipahami.

Penggunaan fungsi dalam suatu program juga akan mempermudah *programmer* dalam melakukan proses *debugging* di mana akan lebih mudah mengidentifikasi *error-code* dengan melihat *block-code* tertentu saja.

Dalam bahasa pemrograman secara umum dikenal 2 tipe fungsi yaitu:

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

   Fungsi jenis ini akan 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*)

   Fungsi jenis ini 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.

5.2.2. Method

Method merupakan sebuah fungsi. Penyebutan method lebih erat dikaitkan dengan konsep pemrograman berorientasi objek (PBO), suatu metode pemrograman yang menitikberatkan pada penggunaan *object* dan *class* (dipelajari pada modul 5).

Method pada dasarnya merupakan suatu function yang berada pada suatu *class* yang mana dapat diimplementasikan oleh *class* lain. Karena method merupakan suatu function maka cara pendefinisiannya pun sama dengan pendefinisian function sesuai dengan bahasa pemrograman yang digunakan. Adapun cara menggunakan method yakni tidak langsung memanggil dengan nama fungsi tersebut melainkan harus melalui sebuah objek yang merupakan *instance* (perwujudan) dari *class* yang mendefinisikan function/method tersebut.

5.2.3. Method vs Fucntion

Perbedaan paling mudah diamati dari function dan method yakni pada cara pemanggilannya. Suatu fungsi dipanggil dengan langsung menyebutkan nama fungsi dan menyertakan argumen (jika diperlukan). Sedangkan suatu method dipanggil melalui *instance object* yang merupakan perwujudan dari suatu *class* dengan menggunakan (umumnya) tanda titik (.).

Sebagai contoh, untuk memanggil fungsi bernama **cetak**, cukup menuliskan `cetak();`. Sementara itu, untuk memanggil method **cetak** dibutuhkan suatu objek ‘x’ (misal objek bernama **printer**) dari kelas pendefinisi method tersebut, maka *code* yang dituliskan yaitu `printer.cetak();`

5.2.4. Penulisan dan penggunaan Function

Secara umum, suatu fungsi akan memiliki:

- Ruang lingkup penggunaan (opsional)

  Ruang lingkup atau disebut *access modifier*/*access specifier* digunakan pada PBO di mana akan menentukan *object*/*class* mana saja yang dapat

menggunakan *variable* maupun *function/method* yang berada di *class* tersebut.0

Dikenal 3 jenis *modifier*:

a. *Public*

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

b. *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)

c. *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*.

- Tipe data

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

- Nama Fungsi

  Nama diberikan untuk keperluan penggunaan/pemanggilan. Fungsi dapat digunakan dengan memanggil nama fungsi serta dengan menyertakan satu atau lebih argumen (jika diperlukan)

- Parameter dan Argumen (opsional)

Parameter merupakan *local variable* dari sebuah fungsi. *Local variable* ini menyimpan nilai secara sementara dan hanya digunakan pada proses yang ada di dalam fungsi yang bersangkutan.

Parameter dituliskan di dalam tanda kurung (()) setelah nama fungsi dan bersifat opsional. Pada sebuah fungsi dapat dideklarasikan beberapa parameter sekaligus dengan memisahkannya menggunakan tanda koma (,).

Argumen merupakan nilai yang diberikan saat pemanggilan fungsi. Argumen akan menjadi nilai dari parameter untuk selanjutnya diproses oleh fungsi yang bersangkutan. Banyaknya argumen yang diberikan bergantung pada banyaknya parameter yang diminta oleh fungsi yang dipanggil.

Ada pula *default-argument* yang merupakan nilai dari parameter yang langsung diberikan saat pembuatan fungsi. Nilai ini akan dipakai oleh fungsi yang bersangkutan saat *user* tidak menyertakan argumen ketika memanggil fungsi tersebut. Pemanggilan fungsi ber-parameter tanpa menyertakan argumen akan menimbulkan *error*. Namun hal ini tidak terjadi ketika telah didefinisikan *default-argument*

- *Keyword return* (kondisional)

  Keyword ini hanya digunakan ketika mendefinisikan fungsi dengan tipe data *non-void* (*return type function*).

Fungsi dapat dituliskan sebagai berikut :

a. Java dan C++

```
<tipe_data> <nama_fungsi>(parameter1, parameter2) {
 Proses1;
 Proses2;
 return;
```

b. Python

```
def <nama_fungsi>(param1, param2) :
   proses1
```


```
    proses2
    return
```

c. PHP

```
function <nama_fungsi>(param1, param2) {
   proses1;
   proses2;
   return;
}
```

Untuk menggunakan fungsi dapat dilakukan dengan menuliskan nama fungsi beserta argumen yang diperlukan

```
nama_fungsi(argumen1, argumen2);
```


5.3. Percobaan

5.3.1. Function pada Java

Untuk percobaan 1, buat *project* dengan nama **,** XX diganti dengan nomor kelompok. Setelah itu masukkan *source code* ke dalam editornya.

Sou

rce Code untuk **modul4_kelxx**

```
public class Modul4_kelxx {

    // 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 2022 ^^ ");
    }

    // 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 mengisikan
      argument untuk parameter kelompok
        cetak("Kelompok XX");

        //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());
    }
}
```


```
Kelompok XX
Hello Praktikan DKP 2022 ^^
Function dengan Return
Kali ini kita akan belajar mengenai Function

Process finished with exit code 0
```

Gambar 5. 1 Output Function pada Java

Keterangan :

a. Fungsi `cetak()`
   - Fungsi tersebut tidak membalikkan nilai
   - Fungsi ini merupakan ***non-return type***, ditandai dengan tipe data ***void***
   - Fungsi tersebut memiliki sebuah parameter dengan tipe data *string*, dengan variabelnya kelompok

b. Fungsi `cetak1()`
   - Fungsi tersebut tidak membalikkan nilai
   - Fungsi ini merupakan ***non-return type***, ditandai dengan tipe data ***void***
   - Fungsi tersebut tidak memiliki parameter

c. Fungsi `cetak2()`
   - Fungsi tersebut membalikkan nilai, nilai yang dibalikkan bertipe data ***String***
   - Fungsi ini merupakan ***return type***, ditandai dengan tipe data ***non-void*** **(*String*)**
   - Fungsi tersebut memiliki parameter yang bertipe data *String* dengan variabel *text*

d. Fungsi `cetak3()`
   - Fungsi tersebut membalikkan nilai, nilai yang dibalikkan bertipe data ***String***
   - fungsi ini merupakan ***return type***, ditandai dengan tipe data ***non-void*** (***String***)
   - Fungsi tersebut tidak memiliki parameter

5.3.2. Method pada Java

Percobaan 2 , buat sebuah *class* dengan nama **pendefinisi.** Setelah itu isikan editor dengan *source code* seperti pada di bawah ini.

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;
    }
```

Setelah itu, klik *file* modul4_kelxx yang tadi digunakan pada percobaan 1 dan tambahkan *source code* pada *file* modul4_kelxx.

Class Pemanggil Method

```
public static void main(String[] args) {
        ...
        pendefinisi objek = new pendefinisi();

        objek.greeting();
        objek.kelompok("Kelompok xx");
        String print = objek.kenalan("Faqih", "Tidur");
        System.out.println(print);
        ...
    }
```

```
public static void main(String[] args) {
    //penggunaan function non-return type dengan mengisikan argument untuk parameter kelompok
    cetak( kelompok: "Kelompok XX");

    //penggunaan function non-return type tanpa parameter
    cetak1();

    //penggunaan function return type dengan mengisikan nilai pada parameter text
    System.out.println(cetak2( text: "Function dengan Return"));

    //penggunaan function return type tanpa parameter
    System.out.println(cetak3());

    pendefinisi objek = new pendefinisi();

    objek.greeting();
    objek.kelompok("Kelompok xx");
    String print = objek.kenalan( nama: "Faqih",  hobi: "Tidur");
    System.out.println(print);

}
```

Gambar 5. 4 *Source Code* pada *file* modul4_kelxx

Gambar di bawah adalah hasil *running program* percobaan 2

```
Kelompok XX
Hello Praktikan DKP 2022 ^^
Function dengan Return
Kali ini kita akan belajar mengenai Function
Hai, saya method dari class pendefinisi. Salam kenal :)
Kelompok xx
Hai, Nama saya Faqih hobi saya Tidur

Process finished with exit code 0
```

Gambar 5. 5 *Output* Method Pada Java

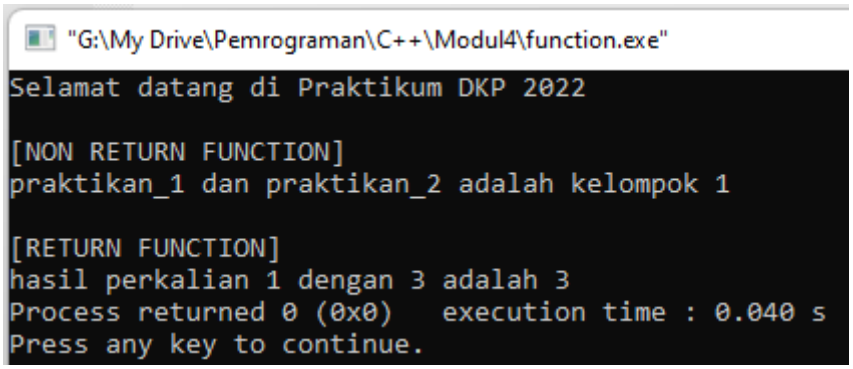
5.3.3. Function pada C++

Untuk percobaan 1, buat *project* dengan nama **modul4_kelXX ,** XX diganti dengan nomor kelompok. Setelah itu masukkan *source code* ke dalam editornya.

Source Code untuk **modul4_kelxx**

```
#include <iostream>
#include <string>
```

```
using namespace std;

void  non_return_func(string  praktikan1,  string  praktikan2,  int
      kelompok) {

      cout  <<  praktikan1  <<  "  dan  "  <<  praktikan2  <<  "  adalah
      kelompok " << kelompok << endl;
}

int return_func(int perkalian) {

    if (perkalian > 0 || perkalian < 3) {
        return perkalian * 3;
    }
    else {
        return perkalian * 0;
    }
}

int main() {
    cout << "Selamat datang di Praktikum DKP 2022\n" << endl;
    cout << "[NON RETURN FUNCTION]" << endl;
    non_return_func("praktikan_1", "praktikan_2", 1);
    cout << "\n[RETURN FUNCTION] " << endl;
        cout  <<  "hasil  perkalian  1  dengan  3  adalah  "  <<
      return_func(1);
}
```



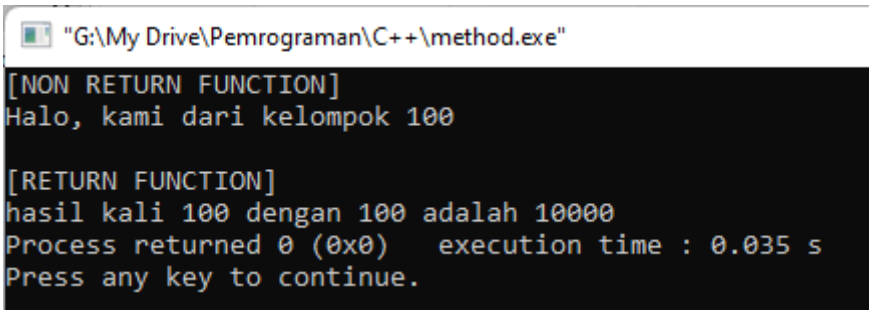
Pada program di atas, terdapat 2 *function* yang digunakan yaitu `non_return_func()` dan `return_func()`. Pada fungsi `non_return()` fungsi akan mencetak *string* ke konsol dengan parameter berupa 3 *string* yaitu praktikan1, praktikan2, dan kelompok. Sedangkan pada `return_func()` fungsi akan mengembalikan nilai dari parameter int perkalian. Nilai yang dikembalikan tergantung dari parameter yang diberikan. Jika nilai berada pada rentang 1 – 3

maka fungsi akan mengembalikan nilai dikali 3, dan jika nilai di luar rentang tersebut, fungsi akan mengembalikan nilai 0. Perlu diingat `main()` harus berada di bawah function

5.3.4. Method pada C++

Source Code

```
#include <iostream>
#include <string>
using namespace std;

class MyClass {
public:
    void non_return_method(string kelompok) {
        cout << "Halo, kami dari kelompok " << kelompok;
    }

    int with_return_method(int angka) {
        return angka*angka;
    }
};

int main() {
    MyClass myObj;
    cout << "[NON RETURN FUNCTION]" << endl;
    myObj.non_return_method("100");
    cout << "\n\n[RETURN FUNCTION]" << endl;
         cout << "hasil kali 100 dengan 100 adalah " <<
      myObj.with_return_method(100);
    return 0;
}
```



Gambar 5. 7 Output Method pada C#

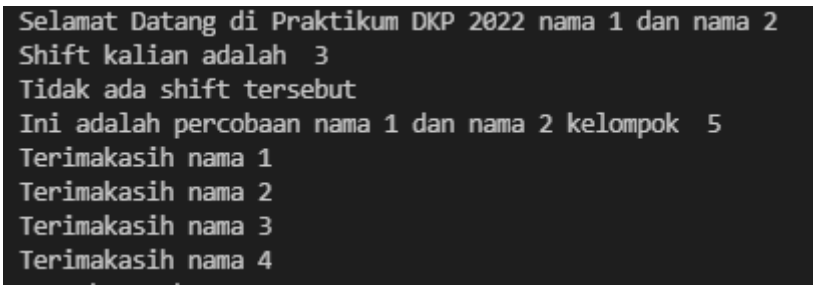
Pada program di atas terdapat *class* di luar dari `main()`. Dibuat objek pada `main()` untuk memanggil *class*, kemudian digunakan untuk memanggil fungsi `non_return_method` menggunakan objek yang sudah dibuat.

5.3.5. Function pada Python

Untuk percobaan 1, buat project dengan nama **modul4_kelXX ,** XX diganti dengan nomor kelompok. Setelah itu masukkan *source code* ke dalam editornya.

Source Code untuk **modul4_kelxx**

```
#Function dengan non return type
def non_return_func(praktikan1, praktikan2):
     print(f"Selamat Datang di Praktikum DKP 2022 {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(3)
anonim_func("nama 1", "nama 2", 5)
arbitrary_func("nama 1", "nama 2", "nama 3", "nama 4")
```

Gambar 5. 8 *Output* Function pada python

Pada percobaan di atas, terdapat 4 tipe *function* yang digunakan yaitu *non – return function, return function, arbitrary function,* dan *anonymous function. Arbitrary function* merupakan fungsi yang jumlah parameternya fleksibel. Parameter dideklarasikan dengan menambahkan simbol *asterisk* (*) pada awal parameter. Fungsi ini dapat digunakan jika kita tidak mengetahui secara pasti parameter yang akan digunakan pada fungsi. Kemudian, *anonymous function,* merupakan fungsi yang dapat dideklarasikan tanpa memberikan nama fungsi. Fungsi ini memanfaatkan *lambda* yang ada pada python.

5.3.6. Method pada Python

Untuk percobaan 2 tambahkan *file* baru yang akan menampung *method* yang akan digunakan kemudian beri nama percobaan2.py. Setelah itu masukkan *source code* ke dalam editornya.

Source Code untuk percobaan2.py

```
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
```

Kemudian, pada *file* pertama, tambahkan beberapa kode sehingga menjadi seperti di bawah.

```
from percobaan2 import *

#Function dengan non return type
def non_return_func(praktikan1, praktikan2):
    print(f"Selamat Datang di Praktikum DKP 2022 {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(3)
anonim_func("nama 1", "nama 2", 5)
arbitrary_func("nama 1", "nama 2", "nama 3", "nama 4")

praktikan = contoh_method("nama 1", "nama 2")
praktikan.mulai()
praktikan.selesai(2)
```

```
Selamat Datang di Praktikum DKP 2022 nama 1 dan nama 2
Shift kalian adalah  3
Tidak ada shift tersebut
Ini adalah percobaan nama 1 dan nama 2 kelompok  5
Terimakasih nama 1
Terimakasih nama 2
Terimakasih nama 3
Terimakasih nama 4
Selamat Datang di Percobaan 2 nama 1 dan nama 2
Percobaan akan selesai dalam :
2
1
```

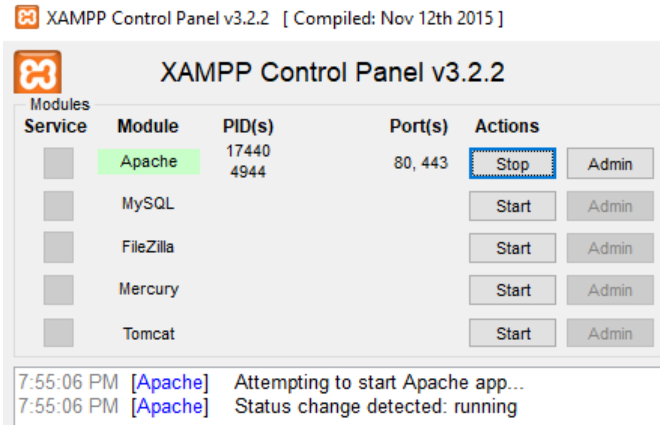
Gambar 5. 9 Output Method pada python

Pada program di atas, terdapat digunakan 1 *class* baru yang digunakan untuk menampung *method.* Pada python, terdapat *method* `init()`. Method ini merupakan *method* yang akan dieksekusi secara otomatis ketika *class* dari *method* tersebut dipanggil pada sebuah objek. Kemudian, terdapat *self parameter* yaitu parameter yang mengacu pada *class* itu sendiri. *Self parameter* ini harus dideklarasikan sebagai parameter pertama dari sebuah *method.* Untuk penamaan,

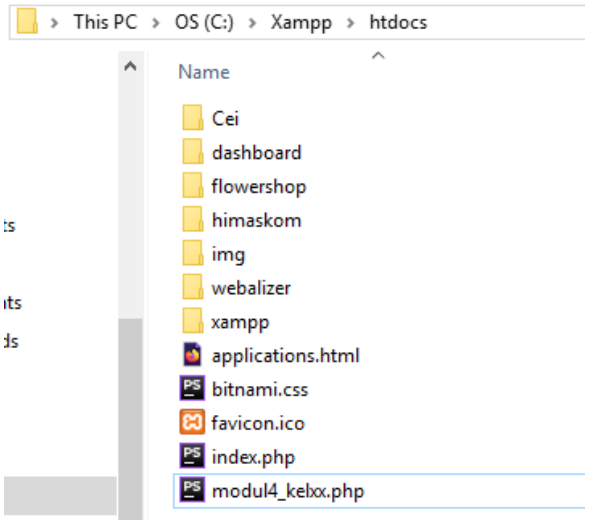
dapat dinamai dengan bebas, hanya parameter ini harus diletakkan sebagai parameter pertama.

5.3.7. Function pada PHP

Nyalakan terlebih dahulu Apache dari Xampp Control Panel



Percobaan 1, membuat *file* bernama modul4_kelxx.php **(xx diganti dengan kelompok kalian)** dalam folder Xampp > htdocs

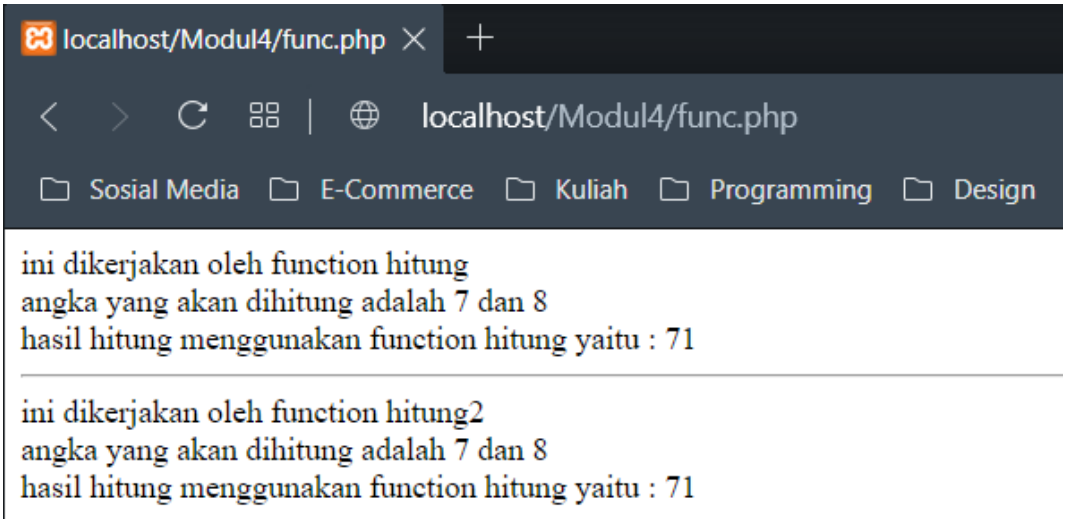


New File dengan nama modul4_kelxx.php di htdocs

Open File tersebut dengan *text editor* lalu masukkan *source code* berikut


```
<?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);
```

Berikut adalah gambar hasil dari percobaan 1



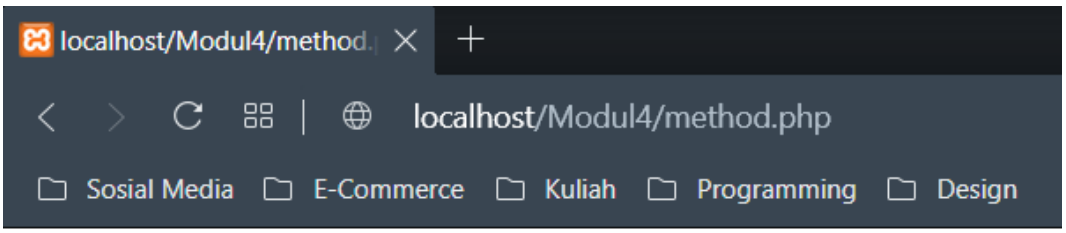
5.3.8. Method pada PHP

Di percobaan 2 sedikit berbeda dengan bahasa pemrograman lain, OOP pada PHP menggunakan notasi berbeda untuk menyatakan *method* yaitu notasi panah (->) bukan titik.

Buat *file* baru yang diberi nama **method_kelxx.php**, setelah itu beri *source code* ke *text* editor seperti berikut

```
<?php

	declare(strict_type s


;ne ak;s; = 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');
```

Berikut adalah hasil dari gambar hasil dari percobaan 2



akar dari 64 adalah 8

Jangan lupa cuci tangan dengan air dan sabun
