**MODUL 4**

**FUNCTION DAN METHOD**

* 1. **Tujuan**

1. Praktikan dapat membedakan konsep dasar function dan method
2. Praktikan dapat memahami dasar penggunaan function dan method
3. Praktikan dapat mengimplementasikan function dan method dalam bahasa pemrograman C#, Python, Java dan PHP
   1. **Dasar Teori**
      1. **Function**

Function merupakan sekumpulan instruksi yang dikelompokan 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 code 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 dimana 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.

1. 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.

* + 1. **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 pasa suatu class yang mana dapat diimplementasikan oleh class lain. Karena method merupakan suatu function maka cara pendefinisannya 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.

* + 1. **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();

* + 1. **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 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*.

* 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 :

1. Java dan C#

|  |
| --- |
| <tipe\_data> <nama\_fungsi>(parameter1, parameter2) {  Proses1;  Proses2;  return;  } |

1. Python

|  |
| --- |
| def <nama\_fungsi>(param1, param2) :  proses1  proses2  return |

1. PHP

|  |
| --- |
| function <nama\_fungsi>(param1, param2) {  proses1;  proses2;  return;  } |

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

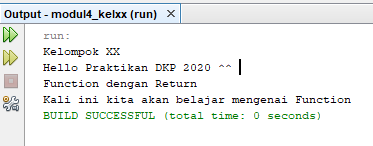
nama\_fungsi(argumen1, argumen2);

* 1. **Percobaan**
     1. **Function pada Java**

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**

|  |
| --- |
| 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 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 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());  }  } |



Gambar 5. 1 Output Function pada Java

Keterangan :

1. Fungsi cetak()

* Fungsi tersebut tidak membalikan nilai
* Fungsi ini merupakan ***non-return type***, ditandai dengan tipe data ***void***
* Fungsi tersebut bernama **cetak**
* Fungsi tersebut memiliki sebuah paramater dengan tipe data string, dengan variabelnya kelompok

1. Fungsi cetak1()

* Fungsi tersebut tidak membalikan nilai
* Fungsi ini merupakan ***non-return type***, ditandai dengan tipe data ***void***
* Fungsi tersebut bernama **cetak1**
* Fungsi tersebut tidak memiliki parameter

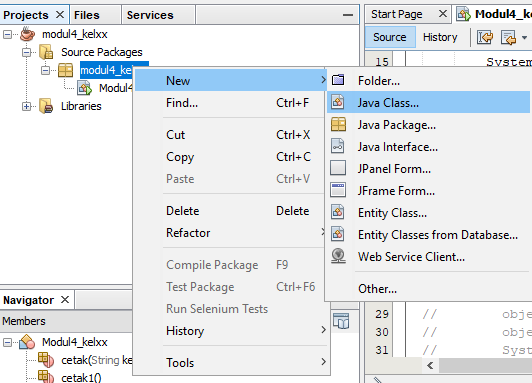
1. Fungsi cetak2()

* Fungsi tersebut membalikan nilai, nilai yang dibalikan bertipe data String
* Fungsi ini merupakan **return type**, ditandai dengan tipe data **non-void (String)**
* Fungsi tersebut bernama **cetak2**
* Fungsi tersebut memiliki parameter yang bertipe data String dengan variabel text

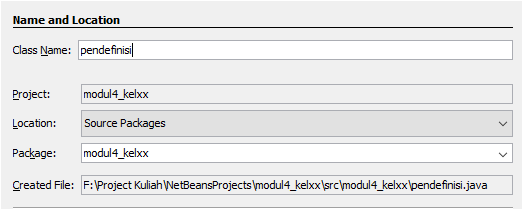
1. Fungsi cetak3()

* Fungsi tersebut membalikan nilai, nilai yang dibalikan bertipe data String
* fungsi ini merupakan return type, ditandai dengan tipe data non-void (String)
* Fungsi tersebut bernama cetak3
* Fungsi tersebut tidak memiliki parameter
  + 1. **Method pada Java**

Percobaan 2 , buat sebuah class dengan nama **pendefinisi**, seperti gambar dibawah ini.



Gambar 5. Membuat Java Class



Gambar 5. Memberi Nama pendefinisi

Setelah mengisi *class name*, klik *finish.* Setelah itu isikan editor dengan source code seperti pada dibawah ini.

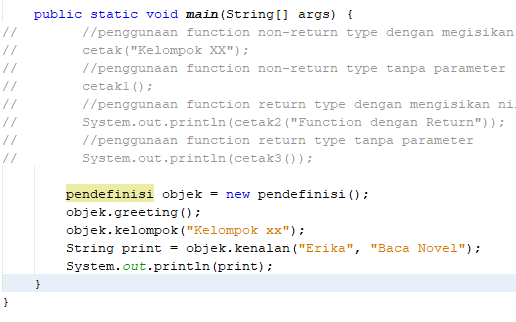
**Class Pendefinisi Method**

|  |
| --- |
| 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, tambahkan komentar (//) pada tiap baris kode dibawah public static void main(String[] args) {, kemudian tambahkan source code pada file modul4\_kelxx seperti gambar berikut.

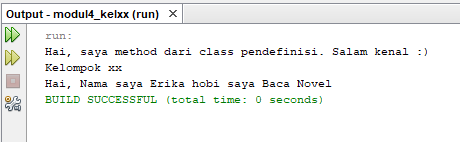
**Class Pemanggil Method**

|  |
| --- |
| public static void main(String[] args) {  pendefinisi objek = new pendefinisi();    objek.greeting();  objek.kelompok("Kelompok xx");  String print = objek.kenalan("Erika", "Baca Novel");  System.out.println(print);  } |



Gambar 5. Source Code pada file modul4\_kelxx

Gambar di bawah adalah hasil running program percobaan 2



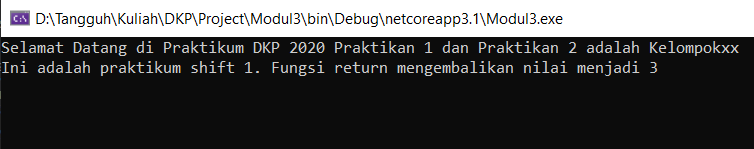
Gambar 5. Output Method Pada Java

* + 1. **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**

|  |
| --- |
| 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("Praktikan 1", "Praktikan 2", "Kelompokxx");  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();  }  } |

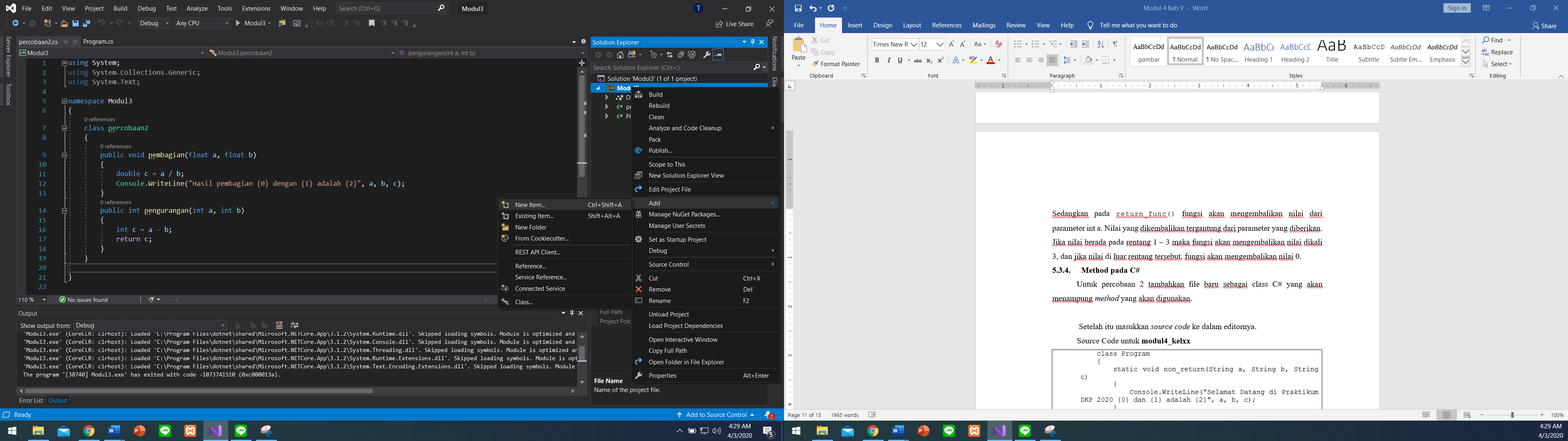


Gambar 5. 6 Output Function pada C#

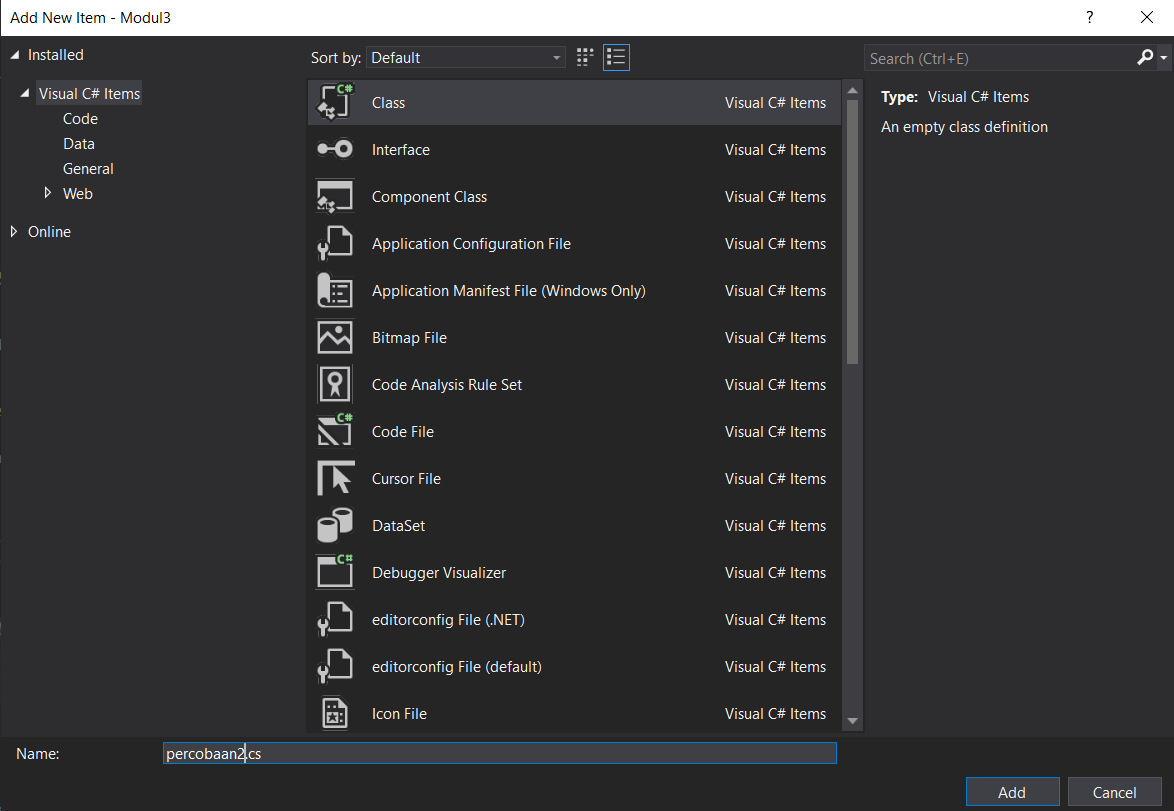
Pada program diatas, terdapat 2 *function* yang digunakan yaitu non\_return() dan return\_func(). Pada fungsi non\_return() fungsi akan mencetak string ke konsol dengan parameter berupa 3 string yaitu a,b, dan c. Sedangkan pada return\_func() fungsi akan mengembalikan nilai dari parameter int a. Nilai 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.

* + 1. **Method pada C#**

Untuk percobaan 2 tambahkan file baru sebagai class C# yang akan menampung *method* yang akan digunakan. Caranya seperti gambar di bawah.



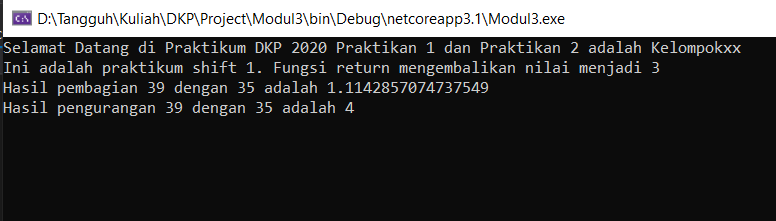
Klik kanan pada nama project yang ada pada panel *Solution Explorer*. Kemudian pilih *add* -> *new item*.



Pilih tipe file *Class,* kemudian beri nama percobaan2.cs. Pemberian nama ada dibagian bawah. Kemudian klik *add.* Setelah itu masukkan *source code* ke dalam editornya. **Hilangkan komen yang ada pada file pertama.**

Source Code untuk **modul4\_kelxx**

|  |
| --- |
| 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;  }  } |



Gambar 5. 7 Output Method pada C#

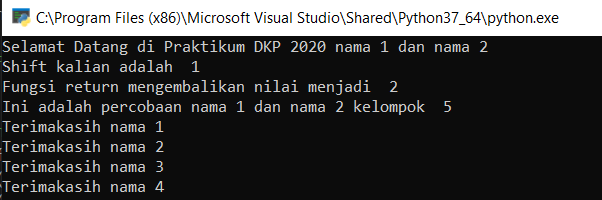
Pada program diatas, terdapat digunakan 1 *class* baru yang digunakan untuk menampung *method.* Di dalam *class* tersebut, terdapat method pembagian()dengan parameter berupa *float,* dan method pengurangan() dengan parameter integer. Untuk memanggil *method* pada *main class*, *class* yang mengandung *method* tersebut harus dideklarasikan sebagai objek. Kemudian, untuk menggunakan *method –* nya, dapat dilakukan dengan memanggil nama objek.nama method. percobaan2 objek = new percobaan2(); digunakan untuk menginisiasikan objek baru yang merujuk pada *class* percobaan2. objek.pembagian(39, 35); digunakan untuk memanggil *method* pembagian yang ada pada *class* percobaan2. Kemudian, objek.pengurangan(39, 35) digunakan untuk memanggil *method* pengurangan.

* + 1. **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 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("nama 1", "nama 2")  return\_func(1)  anonim\_func("nama 1", "nama 2", 5)  arbitrary\_func("nama 1", "nama 2", "nama 3", "nama 4") |

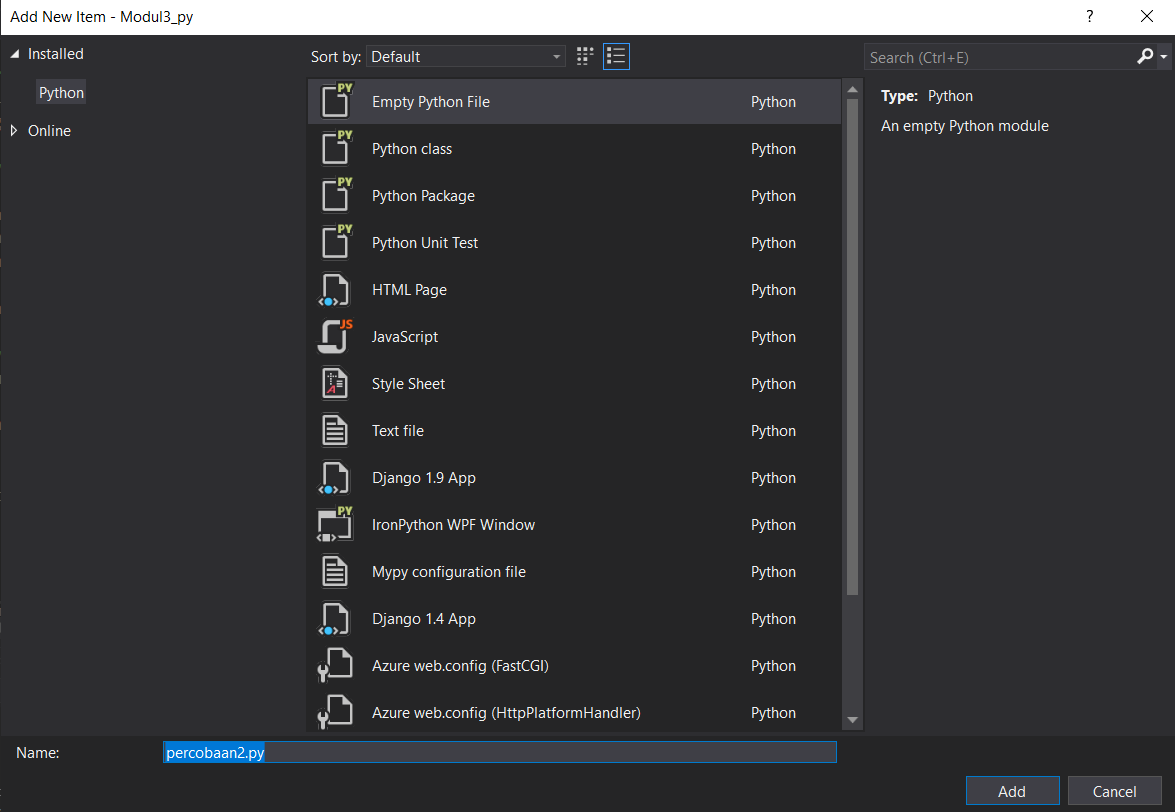


Gambar 5. 8 Output Function pada python

Pada percobaan diatas, terdapat 4 tipe *function* yang digunakan yaitu *non – return function, return function, arbitrary function,* dan *anonymous function. Arbitrary function* merupakan fungsi yang jumlah parameter nya fleksibel. Parameter dideklarasikan dengan menambahkan symbol *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.

* + 1. **Method pada Python**

Untuk percobaan 2 tambahkan file baru yang akan menampung *method* yang akan digunakan. Caranya seperti penambahan pada *class* pada C#.

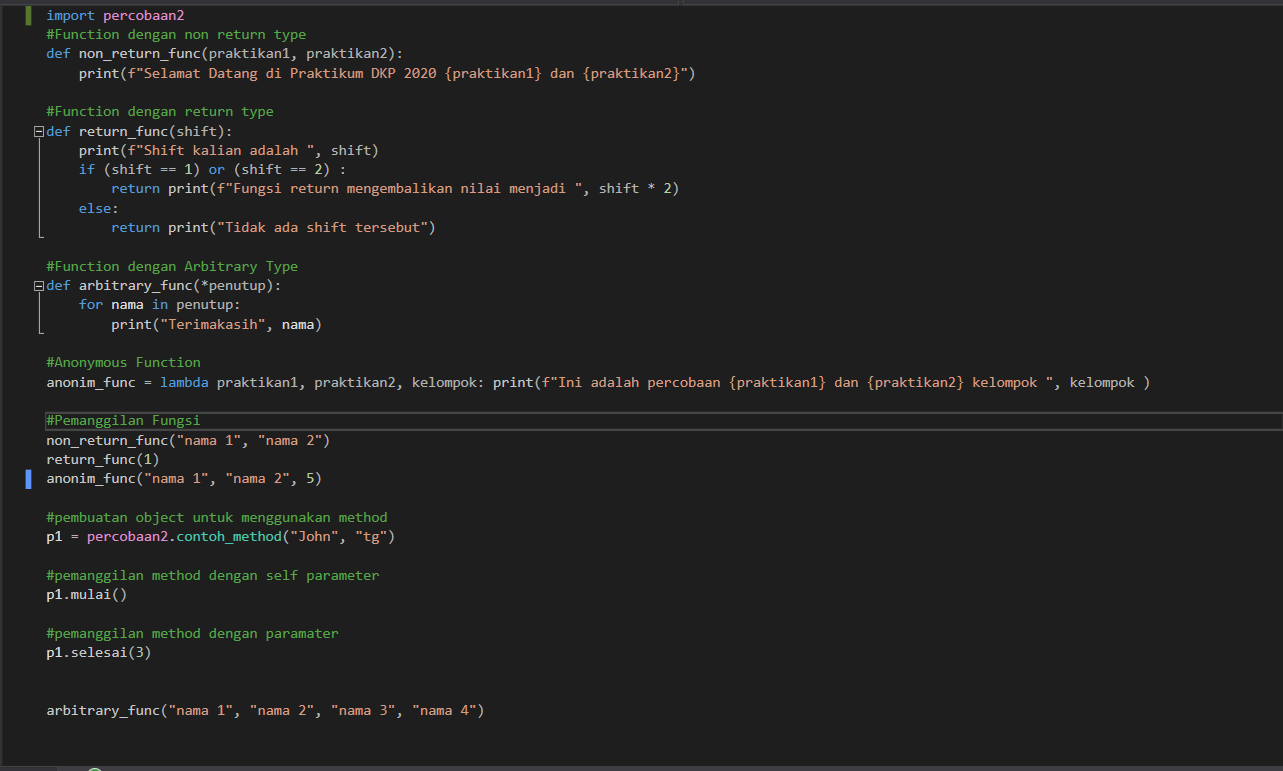


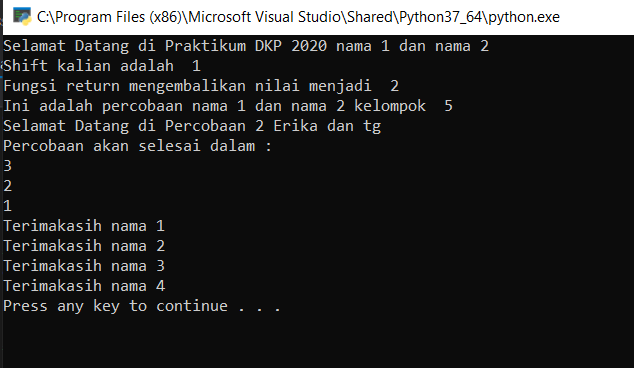
Kemudian pada opsi *add new item,* pilih *empty python file,* kemudian beri nama percobaan2.py. Pemberian nama ada dibagian bawah. Kemudian klik *add.* Setelah itu masukkan *source code* ke dalam editornya.

Source Code untuk **modul4\_kelxx**

|  |
| --- |
| 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 gambar dibawah.



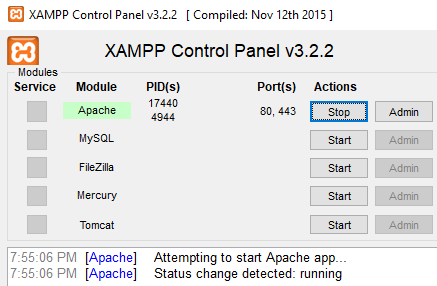


Gambar 5. 9 Output Method pada python

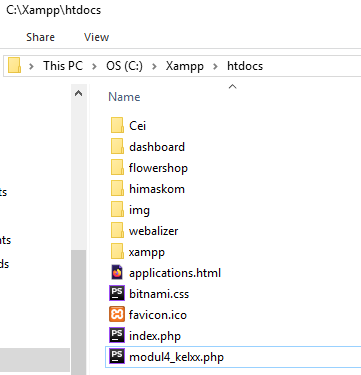
Pada program diatas, terdapat digunakan 1 *class* baru yang digunakan untuk menampung *method.* Pada python, terdapat *method* init(). Method ini merupakan *method* yang akan dieksekusi secara otomatis ketika *class* dari *method* tersebut dipanggil pada sebuah objek. Kemudian, terdapat *self parameter* 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.

* + 1. **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

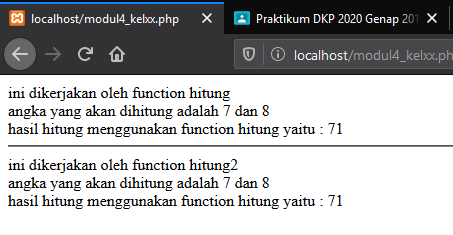


New File dengan nama modul4\_kelxx.php di htdocs

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



* + 1. **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\_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');    ?> |

Berikut adalah hasil dari gambar hasil dari percobaan 2

