SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

STMIK AMIKOM YOGYAKARTA

MODUL
PEMROGRAMAN WEB LANJUT

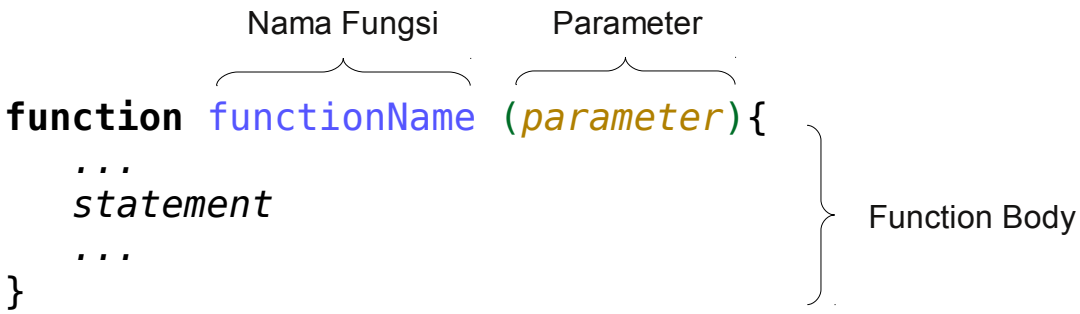
JURUSAN S1 TEKNIK INFORMATIKA
STMIK AMIKOM YOGYAKARTA TAHUN AJARAN 2015/2016

FUNCTION

Fungsi adalah sebuah blok program yang merupakan sekumpulan *statement* yang bertujuan untuk menyelesaikan suatu tugas tertentu. Sebuah fungsi dibuat untuk membantu mengerjakan tugas yang kompleks secara efektif dan efisien. Karena setelah satu fungsi dibuat, ia dapat dipanggil dibagian program manapun untuk menyelesaikan suatu tugas secara-berulang ulang.

Pada bab-bab sebelumnya, kita mengenal beberapa fungsi *built-in* pada PHP, seperti : `date(), print_r(), count(), sin()` dan lain-lain.

1. Struktur Umum



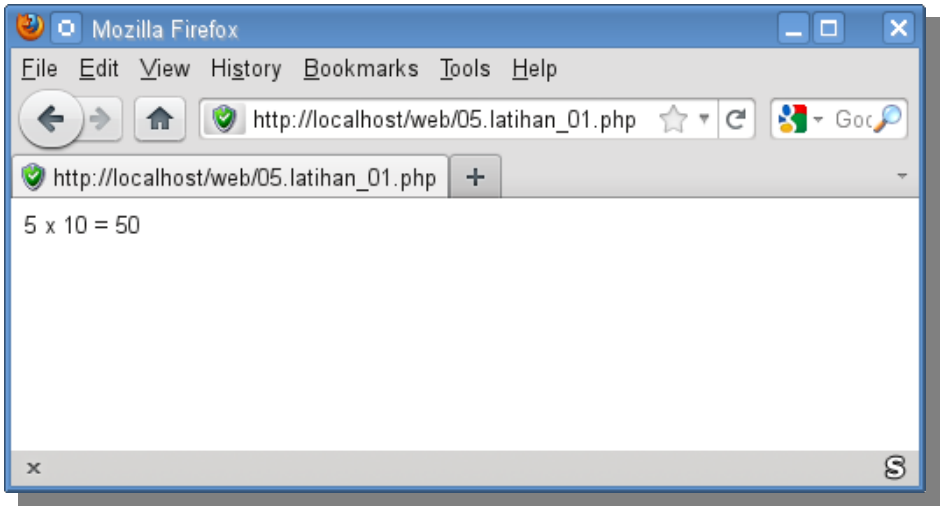
Struktur umum dari fungsi diatas dapat dijelaskan sebagai berikut:

a. Nama fungsi merupakan deklarasi nama fungsi yang akan kita buat. Dalam deklarasinya, fungsi harus memenuhi syarat-syarat sebagai berikut:
   - Tidak boleh sama dengan fungsi yang sudah ada didalam PHP. Ini dikarenakan PHP tidak mendukung fasilitas *overloading* yaitu kondisi dimana fungsi yang dideklarasikan dapat menggunakan nama yang sama namun memiliki hasil keluaran yang berbeda dengan syarat parameter pada fungsi tersebut berbeda.
   - Hanya terdiri dari huruf, angka dan garis bawah (*underscore*).
   - Tidak boleh diawali dengan angka.
b. Parameter adalah data / nilai masukan yang akan diolah oleh fungsi.
c. Function body adalah *statements* / perintah yang akan dikerjakan oleh fungsi.

2. Implementasi Fungsi

05.latihan_01.php

```php
<?php

function perkalian($bil_1, $bil_2){
   $hasil = $bil_1 * $bil_2;
   return $hasil;
}

echo "5 x 10 = ".perkalian(5, 10);

?>
```



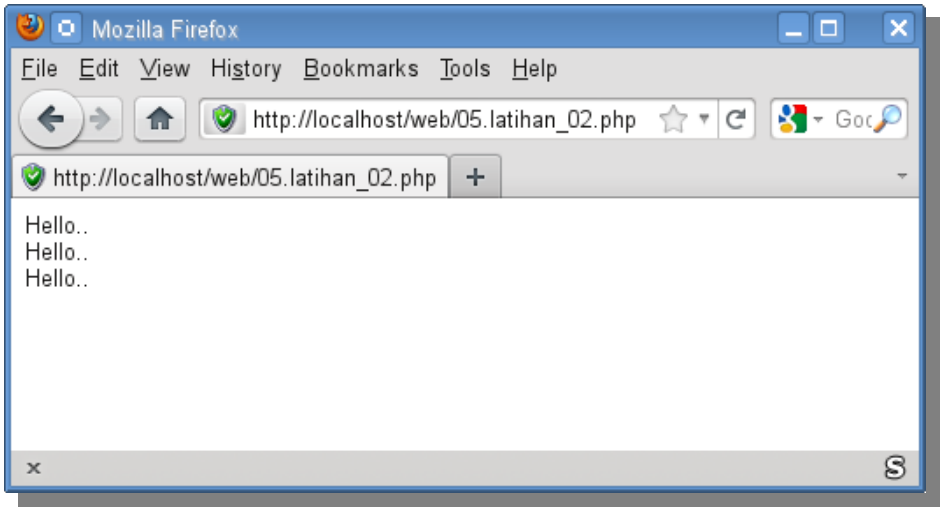
Program diatas dapat dijelaskan sebagai berikut:

a. Deklarasi sebuah fungsi dengan nama **perkalian** dengan parameter input `$bil_1` dan `$bil_2`.
b. Pekerjaan yang dilakukan oleh fungsi **perkalian** adalah mengalikan antara `$bil_1` dan `$bil_2` kemudian hasilnya disimpan pada variabel `$hasil`.
c. Kemudian variabel `$hasil` tersebut akan dijadikan sebagai nilai kembalian / output dari pemanggilan fungsi **perkalian**. Dalam hal ini nilai 50 akan diberikan sebagai umpan balik dari pemanggilan fungsi **perkalian**`(5, 10)`.

Tidak setiap fungsi yang dibuat harus dilengkapi dengan parameter dan mempunyai kembalian. Seperti yang ditunjukkan dalam skrip latihan berikut ini:

05.latihan_02.php

```php
<?php

function sayHello(){
   echo "Hello..<br />";
}

sayHello();
sayHello();
sayHello();

?>
```



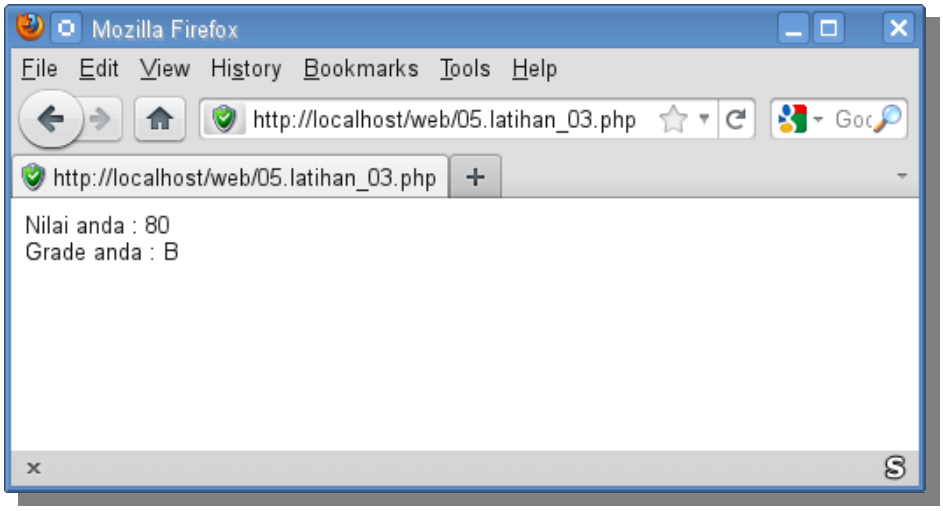
Pada skrip latihan diatas, tidak terdapat parameter yang menjadi masukan pada fungsi `sayHello()`. Demikian pula dengan nilai kembalian dari fungsi yang juga tidak ditentukan. Satu-satunya *statement* yang dikerjakan oleh fungsi tersebut adalah `echo "Hello..<br />";`.

Ketika fungsi `sayHello()` dipanggil / dijalankan sebanyak tiga kali, maka string `Hello..` juga akan ditampilkan sebanyak tiga kali.

05.latihan_03.php

```php
<?php

function hitungGrade($value){
   if($value >= 85 && $value <= 100){
      $grade = "A";
   } else if($value >= 70){
      $grade = "B";
   } else if($value >= 50){
      $grade = "C";
   } else if($value >= 30){
      $grade = "D";
```

```
    } else {
        $grade = "E";
    }

    return $grade;
}

$nilai = 80;

echo "Nilai anda : ".$nilai;
echo "<br />";
echo "Grade anda : ".hitungGrade($nilai);

?>
```



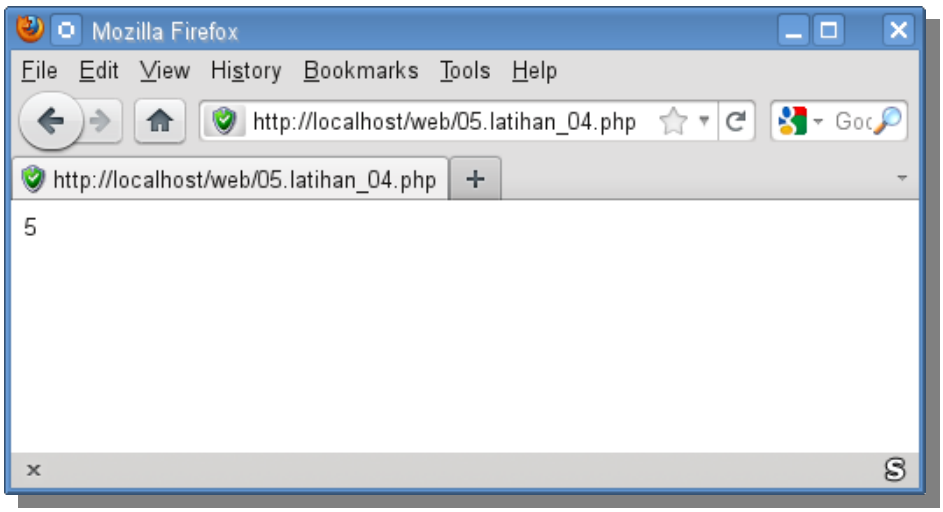
3. Scope

3.1. Local Scope

Setiap fungsi memiliki ruang lingkup kerja sendiri, sehingga variabel yang ada didalam fungsi, tidak akan mempengaruhi variabel yang ada diluar fungsi. Lebih jelas mengenai *local scope* lihatlah contoh latihan berikut :

05.latihan_04.php

```
<?php

$temp = 5;

function doSomething(){
    $temp = 0;
}

doSomething();
echo $temp;

?>
```



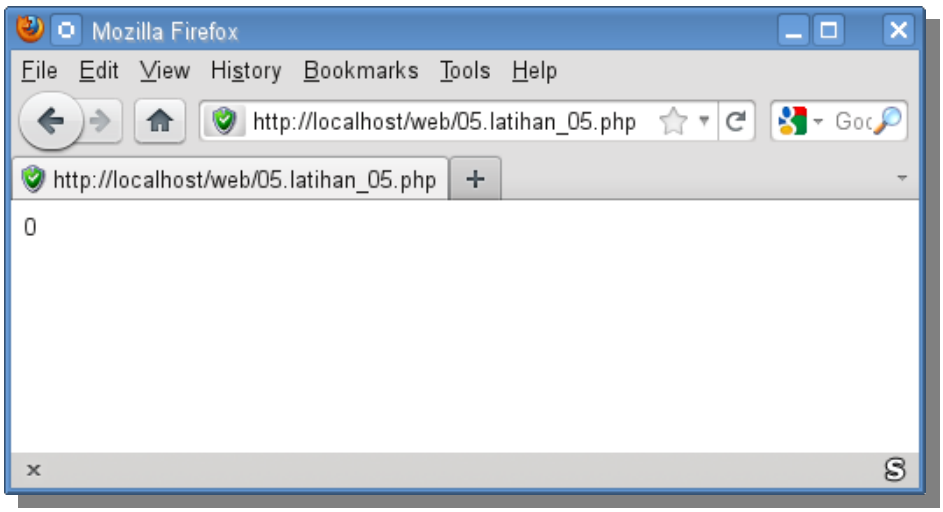
Skrip latihan diatas menunjukkan kepada kita bahwa variabel `$temp` yang telah dideklarasikan sebelumnya tidak akan terpengaruh oleh operasi yang dilakukan oleh fungsi `doSomething()`. Terbukti bahwa variabel `$temp` tetap bernilai `5` walaupun fungsi `doSomething()` telah dijalankan. Ini karena variabel `$temp` pada blok fungsi `doSomething()` hanya bersifat lokal, yaitu hanya berjalan pada function body.

3.2. Global Scope

Bagaimanapun PHP juga menyediakan mekanisme agar variabel yang dideklarasikan di luar function body dapat dikenali dan dioperasikan / diubah nilainya melalui function body. Perhatikan 05.latihan_05.php dibawah ini yang merupakan modifikasi dari latihan sebelumnya.

05.latihan_05.php

```php
<?php

$temp = 5;

function doSomething(){
   global $temp;
   $temp = 0;
}

doSomething();
echo $temp;

?>
```



Variabel `$temp` pada program diatas akan bernilai `0` setelah pemanggilan fungsi `doSomething()`. Dengan kata lain, variabel `$temp` terkena pengaruh operasi yang dilakukan `doSomething()`.

Ini dikarenakan adanya *statement* `global $temp;` yang dijalankan pada function body `doSomething()`. S*tatement* tersebut memberitahukan kepada PHP interpreter bahwa variabel `$temp` pada function body tersebut adalah sebuah variabel global. Dengan demikian variabel `$temp` yang sudah dideklarasikan sebelumnya akan dikenali dan dapat diubah nilainya melalui fungsi tersebut.

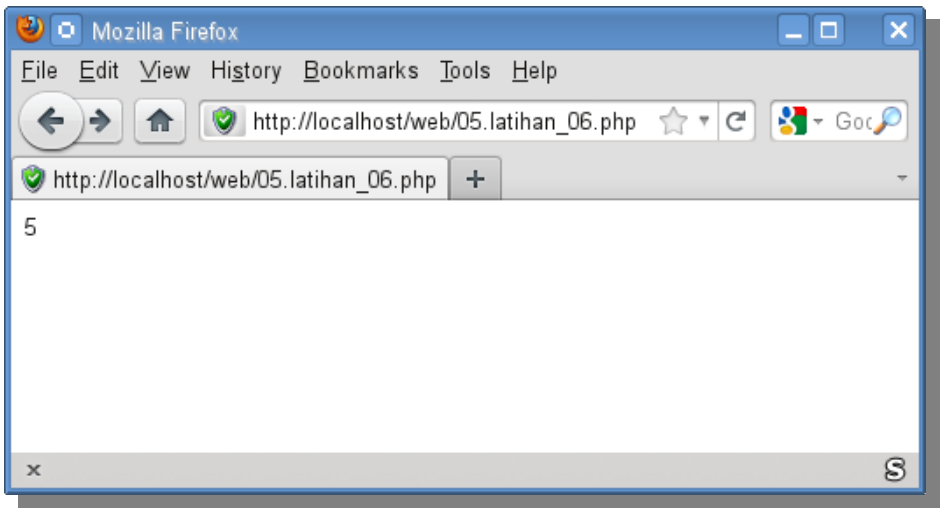
4. Passing Parameter

4.1. By Value

Passing by value ialah teknik memasukkan paramater ke dalam sebuah fungsi dengan cara membuat *copy* dari variabel asli, sehingga variabel asli tidak terpengaruh. Untuk lebih jelasnya, perhatikan contoh latihan berikut :

05.latihan_06.php

```php
<?php

function tambahSatu($value){
   $value++;
}

$a = 5;
tambahSatu($a);
echo $a;

?>
```



Contoh program diatas dapat dijelaskan sebagai berikut:

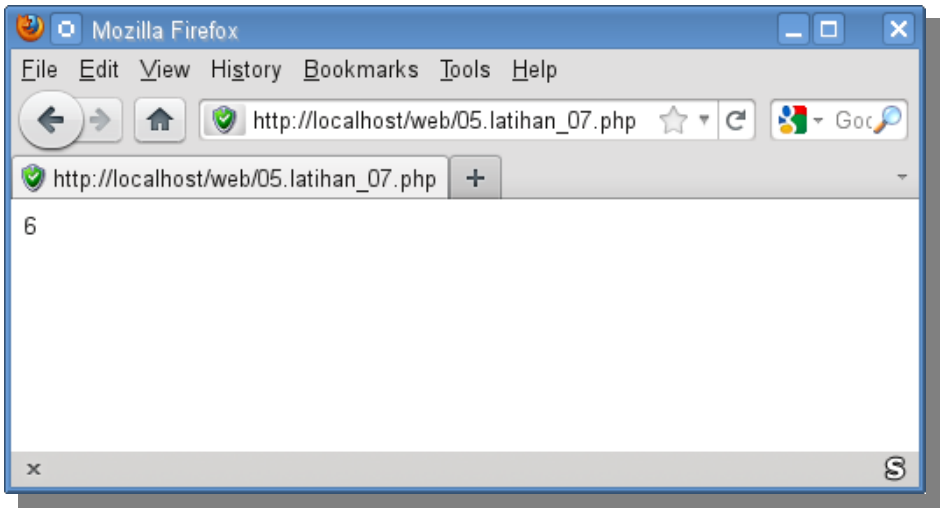
1. Mula-mula variabel `$a` ditugasi untuk menyimpan nilai 5.
2. Kemudian nilai variabel `$a` dimasukkan ke fungsi `tambahSatu()` sebagai parameter untuk diolah lebih lanjut.
3. Didalam fungsi `tambahSatu()`, variabel `$a` akan diproses dan ditangani sebagai variabel `$value`. Dimana data / nilai yang terkandung didalam variabel `$a` juga dimiliki (*copy*) oleh variabel `$value`.
4. *Statement* yang dikerjakan didalam fungsi `tambahSatu()` adalah melakukan *increment* / menambah variabel `$value` dengan nilai satu melalui perintah `$value++;`.
5. Kemudian program mengerjakan *statement* `echo $a;` dimana nilai dalam variabel `$a` tetap bernilai 5. Ini terjadi dikarenakan penambahan nilai (`$value++;`) hanya bekerja didalam fungsi, dan tidak memiliki efek apapun terhadap variabel lainnya diluar tubuh fungsi `tambahSatu()`.

4.2. By Reference

Passing parameter by reference memungkinkan kita untuk melakukan manipulasi terhadap variabel yang menjadi parameter melalui sebuah fungsi. Untuk lebih jelasnya, perhatikan contoh latihan berikut yang merupakan modifikasi dari skrip program sebelumnya :

05.latihan_07.php

```php
<?php

function tambahSatu(&$value){
   $value++;
}

$a = 5;
tambahSatu($a);
echo $a;

?>
```



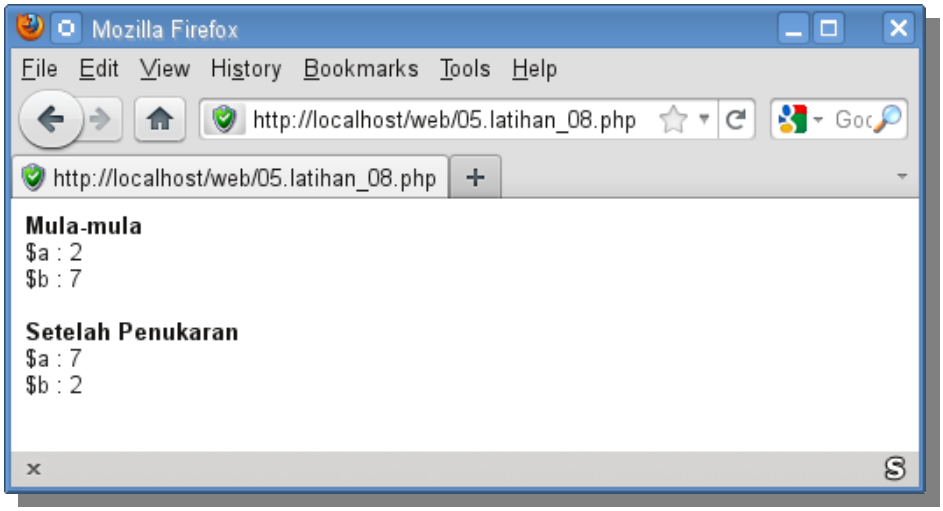
Perbedaan antara skrip 05.latihan_06.php dan 05.latihan_07.php terletak pada cara penanganan parameter. Dimana pada skrip program 05.latihan_07.php menggunakan tanda "&" didepan parameter `$value`. Yang akan memberitahukan PHP interpreter bahwa variabel `$value` tersebut akan diperlakukan sebagai *reference variable*. Yang berarti, bahwa operasi yang dilakukan melalui fungsi `tambahSatu()` pada variabel `$value` akan turut mempengaruhi nilai variabel aslinya, yaitu variabel `$a`.

Ketika program mengerjakan perintah `echo $a;`, maka variabel `$a` bernilai 6. Ini terjadi dikarenakan penambahan nilai (`$value++;`) pada fungsi `tambahSatu()` turut mempengaruhi pada variable `$a` yang menjadi parameter pada fungsi.

Contoh lain yang sering kita jumpai pada dunia pemrograman berkaitan dengan fungsi adalah penukaran nilai antara dua buah variabel. Seperti yang akan ditunjukkan pada latihan berikut:

05.latihan_08.php

```php
<?php

function tukar(&$bil_1, &$bil_2){
   $temp  = $bil_1;
   $bil_1 = $bil_2;
   $bil_2 = $temp;
}

$a = 2;
$b = 7;

echo "<b>Mula-mula</b><br />";
echo "\$a : ".$a;
echo "<br />";
echo "\$b : ".$b;

tukar($a,$b);

echo "<br /><br /><b>Setelah Penukaran</b><br />";
echo "\$a : ".$a;
echo "<br />";
echo "\$b : ".$b;

?>
```

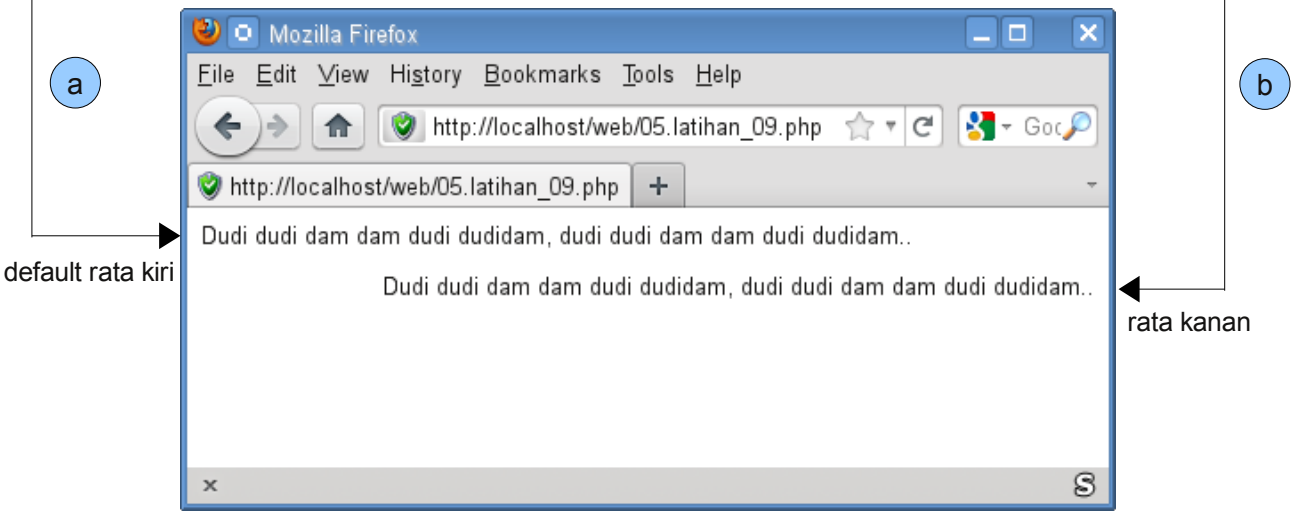


4.3. Default Parameter

Pada suatu kondisi, terkadang kita menginginkan parameter pada sebuah fungsi dapat bersifat optional. Yaitu bilamana sebuah parameter masukan tidak ditentukan oleh programmer, maka nilai parameter tersebut akan diisi oleh *default* nilai yang sudah ditentukan sebelumnya. Untuk lebih jelasnya, perhatikan skrip contoh latihan berikut:

05.latihan_09.php

```php
<?php

function paragraphAlign($str, $align="left"){
   return '<p align="'.$align.'">'.$str.'</p>';
}

$lirik = "Dudi dudi dam dam dudi dudidam, dudi dudi dam dam dudi dudidam..";

echo paragraphAlign($lirik);
echo paragraphAlign($lirik, "right");

?>
```



Fungsi `paragraphAlign()` dideklarasikan dengan dua parameter yaitu `$str` dan `$align`. Yang menjadi fokus perhatikan kita adalah parameter `$align`, dimana `"left"` menjadi default nilai parameter tersebut. Bilamana pada saat pemanggilan fungsi `paragraphAlign()` parameter `$align` tidak ditentukan nilainya, maka secara otomatis parameter `$align` akan diisi / bernilai `"left"`, seperti yang ditunjukkan oleh **proses a**. Sedangkan pemanggilan fungsi `paragraphAlign()` dengan menentukan parameter secara lengkap hasilnya akan ditunjukkan oleh **proses b**.

4. Fungsi Built-in PHP

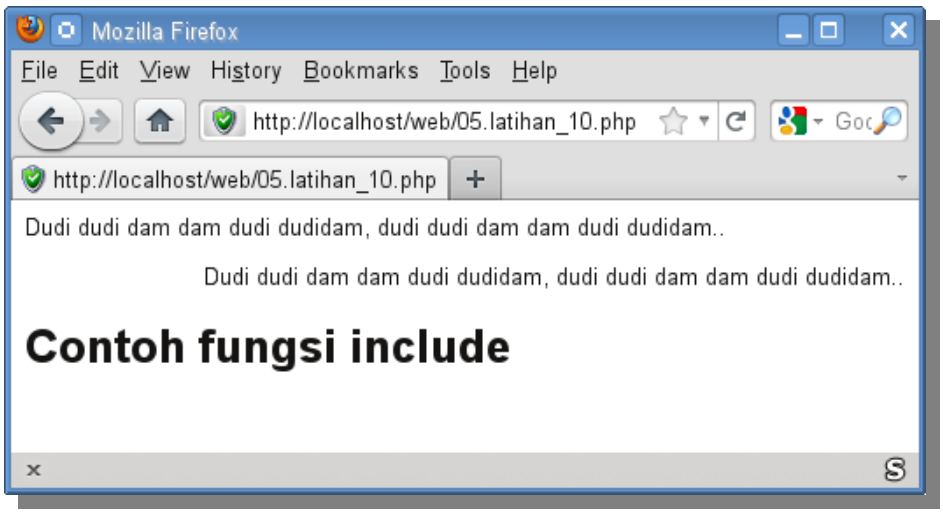
PHP menyediakan fungsi *built-in* yang melimpah. Lebih dari 700 buah fungsi *built-in* yang tersedia pada PHP 5.x yang akan memudahkan kita untuk mengembangkan sebuah sistem. Pada sub bab ini akan dibahas sebagian dari fungsi-fungsi tersebut.

4.1 include

Fungsi `include` memungkinkan kita untuk memasukkan / menyertakan isi sebuah file kedalam sebuah program.

05.latihan_10.php

```
<?php
include("05.latihan_09.php");
echo "<h1>Contoh fungsi include</h1>";
?>
```



Contoh skrip latihan diatas akan memanggil file latihan kita sebelumnya, yaitu `05.latihan_09.php` melalui fungsi `include`.

4.1.1 include vs require

Terdapat cara lain yang digunakan untuk memasukkan / menyertakan isi sebuah file kedalam sebuah program yaitu `require`. Perbedaan antara keduanya yaitu ketika `include` gagal memanggil sebuah file, PHP akan menampilkan pesan error, namun tetap menjalan program utamanya. Berbeda dengan `require` yang akan langsung berhenti jika terdapat masalah pada saat pemanggilan sebuah file. Jalankan dan amati perilaku kedua contoh skrip berikut :

05.latihan_11.php

```php
<?php
include("somefile.php");
echo "<h1>Contoh fungsi include</h1>";
?>
```

05.latihan_12.php

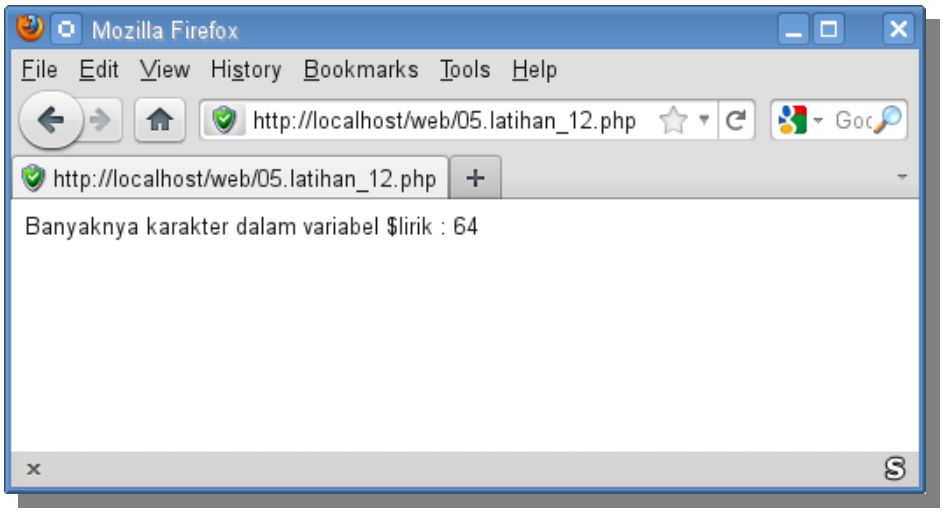
```php
<?php
require("somefile.php");
echo "<h1>Contoh fungsi require</h1>";
?>
```

4.2 strlen

Fungsi `strlen` berguna untuk mengetahui panjang / banyaknya karakter dalam sebuah string / variabel.

05.latihan_12.php

```php
<?php
$lirik = "Dudi dudi dam dam dudi dudidam, dudi dudi dam dam dudi dudidam..";
echo "Banyaknya karakter dalam variabel \$lirik : ".strlen($lirik);
?>
```

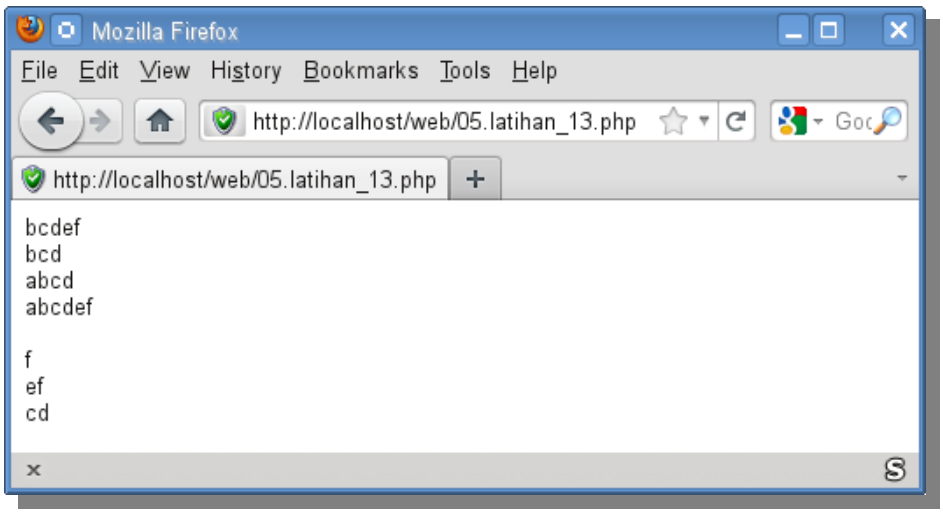


4.3 substr

Fungsi `substr` atau juga disebut fungsi sub string adalah salah satu fungsi manipulasi string yang sangat berguna. Fungsi `substr` digunakan untuk mengambil karakter tertentu pada sebuah string yang dispesifikasikan oleh parameter *awal* dan *panjang*. Perhatikan dan pelajari penggunaan fungsi `substr` pada contoh latihan program dibawah ini:

05.latihan_13.php

```php
<?php

echo substr("abcdef", 1);     // mengembalikan "bcdef"
echo substr("abcdef", 1, 3);  // mengembalikan "bcd"
echo substr("abcdef", 0, 4);  // mengembalikan "abcd"
echo substr("abcdef", 0, 8);  // mengembalikan "abcdef"

echo substr("abcdef", -1);    // mengembalikan "f"
echo substr("abcdef", -2);    // mengembalikan "ef"
echo substr("abcdef", -4, 2); // mengembalikan "cd"

?>
```

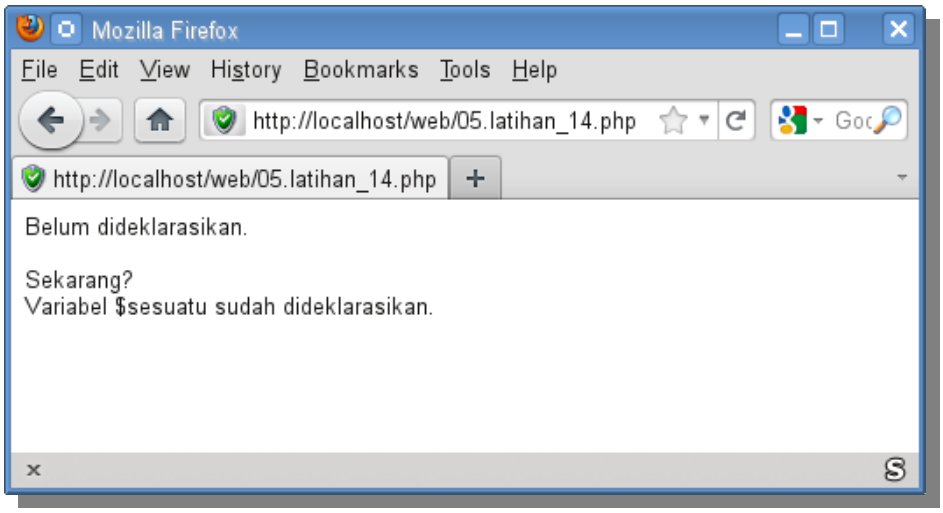


4.4 isset

Fungsi isset digunakan memeriksa apakah sebuah variabel sudah dideklarasikan sebelumnya. Ia akan mengembalikan nilai true jika sebuah variabel sudah dideklarasikan dan false jika belum pernah dideklarasikan.

05.latihan_14.php

```php
<?php

if(isset($sesuatu)){
   echo "Variabel \$sesuatu sudah dideklarasikan.";
} else { echo "Belum dideklarasikan."; }

echo "<br /><br />Sekarang?<br />";

$sesuatu = 14;

if(isset($sesuatu)){
   echo "Variabel \$sesuatu sudah dideklarasikan.";
} else { echo "Belum dideklarasikan."; }

?>
```



Pada contoh latihan program diatas, terdapat dua percabangan. Pada percabangan pertama, kondisi `isset($sesuatu)` akan mengembalikan nilai false karena variabel `$sesuatu` belum dideklarasikan. Untuk percabangan kedua, kondisi `isset($sesuatu)` akan mengembalikan nilai true karena variabel `$sesuatu` sudah dideklarasikan.

DAFTAR PUSTAKA

1. Buzton, Toby. 2002. *PHP By Example*. Indianapolis, Indiana: Que.
2. Muhardin, Endy. 2003. *PHP Programming Fundamental dan MySQL Fundamental*. IlmuKomputer.Com
3. Rasmus, Lerdorf., Dkk. 2003. *PHP 5 Manual*. PHP Documentation Group.