

# MODUL I : PENGENALAN PHP

## 1.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengetahui dan mengenal bahasa pemrograman PHP.
2. Mengetahui cara instalasi software yang diperlukan.
3. Memahami tipe data, variabel, konstanta dan dapat mengimplementasikannya dalam pemrograman PHP.
4. Memahami dan mengimplementasikan operator dan variabel array.

## 1.2 Pengenalan PHP

### 1.2.1 Sejarah Singkat PHP

PHP adalah bahasa pemrograman *script server-side* yang didesain untuk pengembangan web. Selain itu, PHP juga bisa digunakan sebagai bahasa pemrograman umum. PHP dikembangkan pada tahun 1995 oleh **Rasmus Lerdorf**, dan sekarang dikelola oleh The PHP Group. Situs resmi PHP beralamat di <http://www.php.net>.

PHP disebut bahasa pemrograman **server side** karena PHP diproses pada komputer server. Hal ini berbeda dibandingkan dengan bahasa pemrograman *client-side* seperti JavaScript yang diproses pada web browser (client).

Pada awalnya PHP merupakan singkatan dari Personal Home Page. Sesuai dengan namanya, PHP digunakan untuk membuat website pribadi. Dalam beberapa tahun perkembangannya, PHP menjelma menjadi bahasa pemrograman web yang powerful dan tidak hanya digunakan untuk membuat halaman web sederhana, tetapi juga website populer yang digunakan oleh jutaan orang seperti wikipedia, wordpress, joomla, dll.

Saat ini PHP adalah singkatan dari **PHP: Hypertext Preprocessor**, sebuah kepanjangan rekursif, yakni permainan kata dimana kepanjangannya terdiri dari singkatan itu sendiri: **PHP: Hypertext Preprocessor**.

PHP dapat digunakan dengan gratis (free) dan bersifat Open Source. PHP dirilis dalam lisensi PHP License, sedikit berbeda dengan lisensi GNU General Public License (GPL) yang biasa digunakan untuk proyek Open Source.

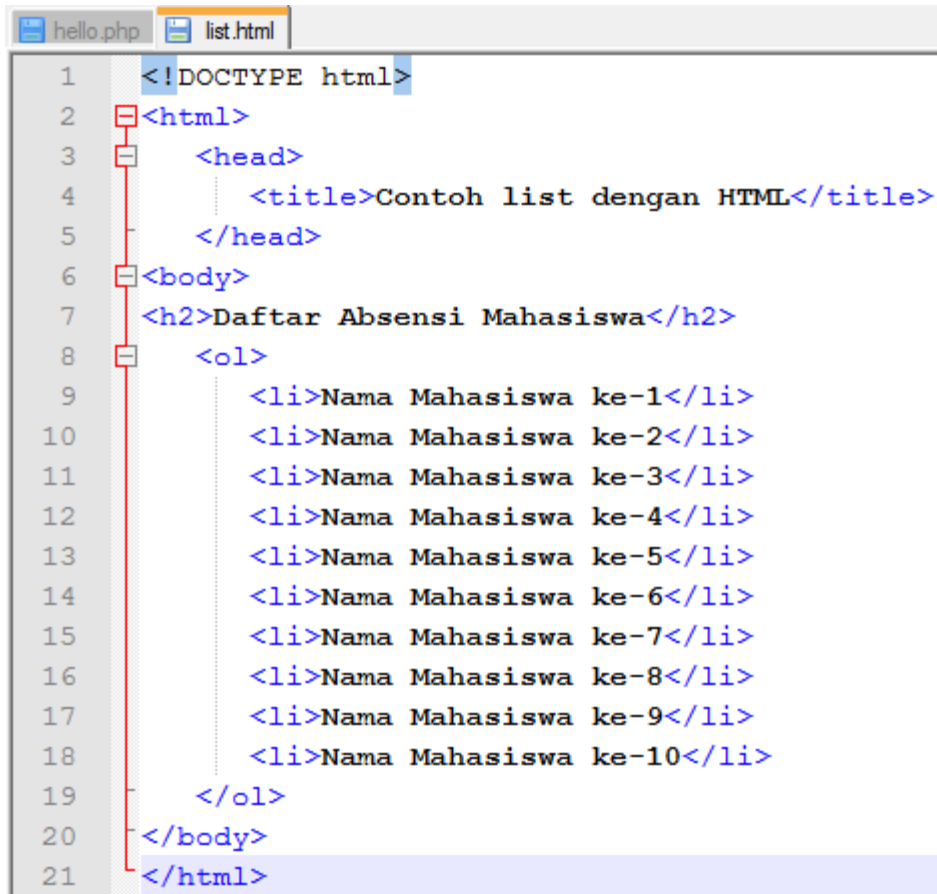
### 1.2.2 Fungsi PHP dalam Pemrograman Web

Untuk membuat halaman web, sebenarnya PHP bukanlah bahasa pemrograman yang wajib digunakan. Kita bisa saja membuat website hanya menggunakan HTML saja. Web yang dihasilkan dengan HTML (dan CSS) ini dikenal dengan website statis, dimana konten dan halaman web bersifat tetap.

Sebagai perbandingan, website dinamis yang bisa dibuat menggunakan PHP adalah situs web yang bisa menyesuaikan tampilan konten tergantung situasi. Website dinamis juga bisa menyimpan data ke dalam database, membuat halaman yang berubah-ubah sesuai input dari user, memproses form, dll.

Untuk pembuatan web, kode PHP biasanya di sisipkan kedalam dokumen HTML. Karena fitur inilah PHP disebut juga sebagai Scripting Language atau bahasa pemrograman script.

Sebagai contoh penggunaan PHP, misalkan kita ingin membuat list dari nomor 1 sampai nomor 10. Dengan menggunakan HTML murni, kita bisa membuatnya secara manual seperti kode berikut ini:



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Contoh list dengan HTML</title>
5   </head>
6   <body>
7     <h2>Daftar Absensi Mahasiswa</h2>
8     <ol>
9       <li>Nama Mahasiswa ke-1</li>
10      <li>Nama Mahasiswa ke-2</li>
11      <li>Nama Mahasiswa ke-3</li>
12      <li>Nama Mahasiswa ke-4</li>
13      <li>Nama Mahasiswa ke-5</li>
14      <li>Nama Mahasiswa ke-6</li>
15      <li>Nama Mahasiswa ke-7</li>
16      <li>Nama Mahasiswa ke-8</li>
17      <li>Nama Mahasiswa ke-9</li>
18      <li>Nama Mahasiswa ke-10</li>
19    </ol>
20  </body>
21 </html>
```

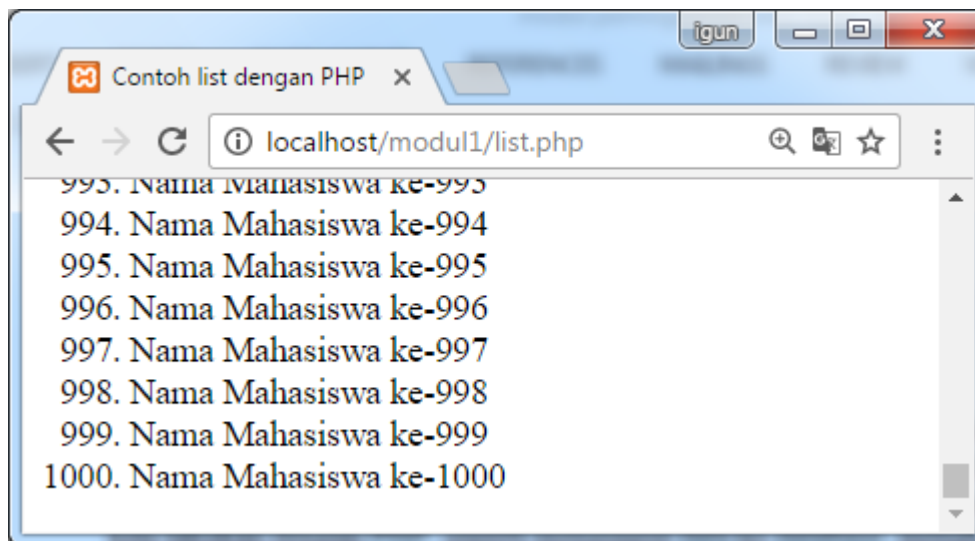
Halaman HTML tersebut dapat dibuat dengan mudah dengan cara men-copy-paste tag <li> sebanyak 10 kali dan mengubah sedikit angka-angka no urut di belakangnya. Namun jika yang kita inginkan adalah menambahkan list tersebut menjadi 100 atau 1000 list, cara copy-paste tersebut menjadi tidak efektif.

Jika menggunakan PHP, kita tinggal membuat perulangan for sebanyak 1000 kali dengan perintah yang lebih singkat seperti berikut ini:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Contoh list dengan PHP</title>
5  </head>
6  <body>
7      <h2>Daftar Absensi Mahasiswa</h2>
8      <ol>
9          <?php
10             for ($i= 1; $i <= 1000; $i++)
11             {
12                 echo "<li>Nama Mahasiswa ke-$i</li>";
13             }
14          ?>
15      </ol>
16  </body>
17  </html>

```



Dengan menggunakan kode baris yang bahkan lebih sedikit, kita dapat membuat list tersebut menjadi 1000 kali, bahkan 100.000 kali dengan hanya mengubah sebuah variabel \$i.

PHP tidak hanya dapat melakukan pengulangan tersebut, masih banyak hal lain yang bisa kita lakukan dengan PHP, seperti menginput data ke database, menghasilkan gambar, mengkonversi halaman text menjadi PDF, management cookie dan session, dan hal lainnya.

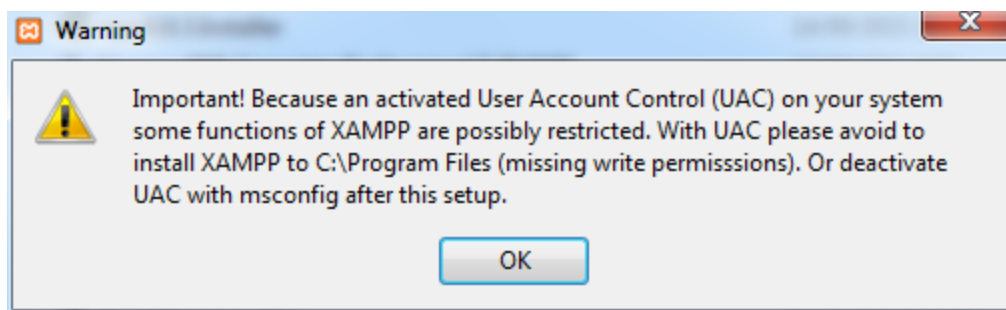
Database yang didukung PHP antara lain : MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC. PHP adalah software Open Source, bebas untuk diunduh dan digunakan.

### 1.3 Instalasi XAMPP

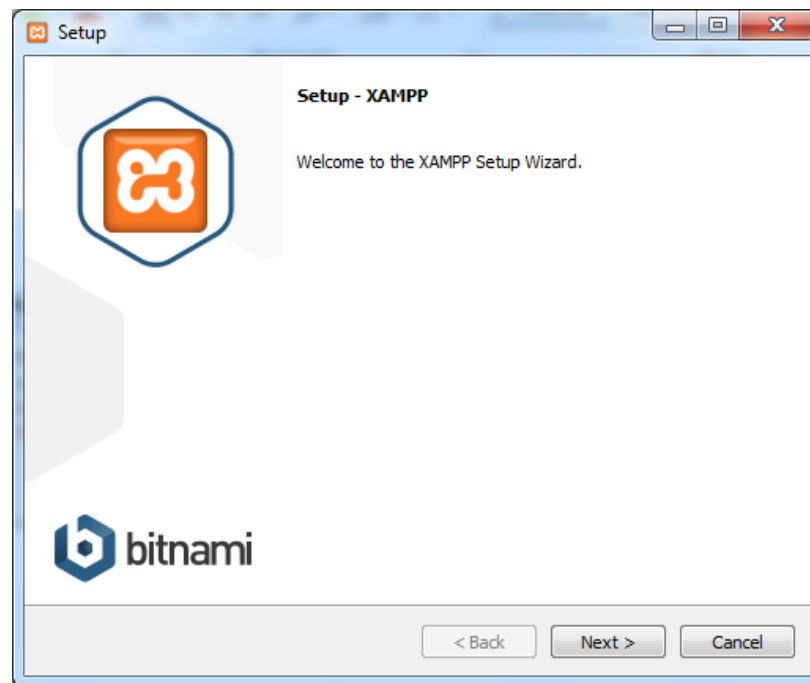
Aplikasi **XAMPP** adalah aplikasi yang *membundle* banyak aplikasi lain yang dibutuhkan dalam pengembangan web. Nama **XAMPP** merupakan singkatan dari aplikasi utama yang ada didalamnya: **X** (huruf X berarti *cross-platform*, dimana aplikasi XAMPP tersedia untuk banyak Sistem Operasi), **A** (*Apache* web server), **M** (*MySQL*), **P** (*PHP*), dan **P** (*Perl*). Selain aplikasi tersebut, XAMPP juga menyertakan modul lain seperti **OpenSSL** dan **phpMyAdmin**. Untuk dapat menggunakan XAMPP, kita harus mendownloadnya di situs resmi XAMPP: <https://www.apachefriends.org>.

Setelah file aplikasi XAMPP tersedia, kita akan mulai menginstall aplikasi ini. Silahkan *double klik* file XAMPP anda. Pada contoh ini, file XAMPP yang digunakan adalah **xampp-win32-5.6.23-0-VC11-installer**.

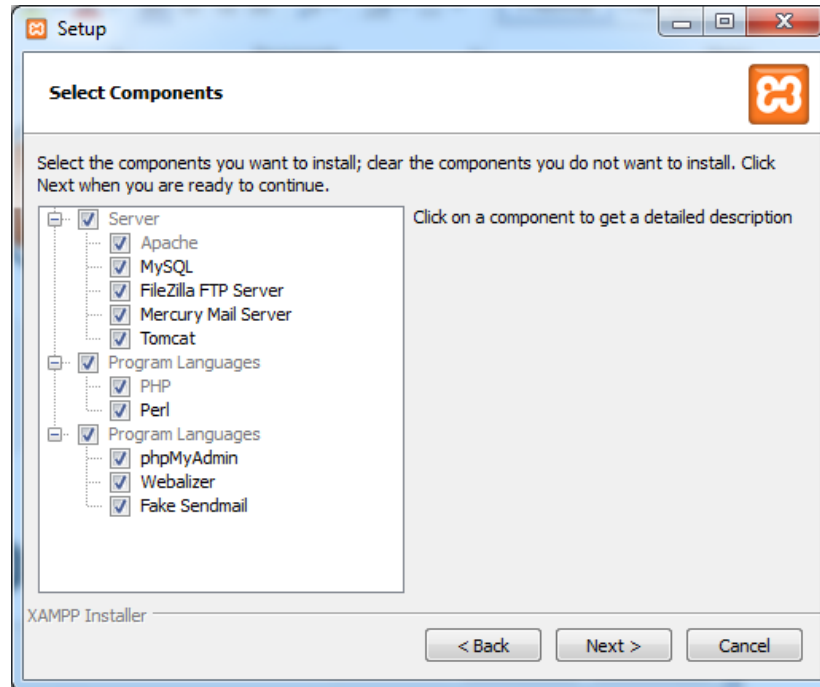
Jendela peringatan berikutnya adalah tentang **UAC (User Account Control)**:



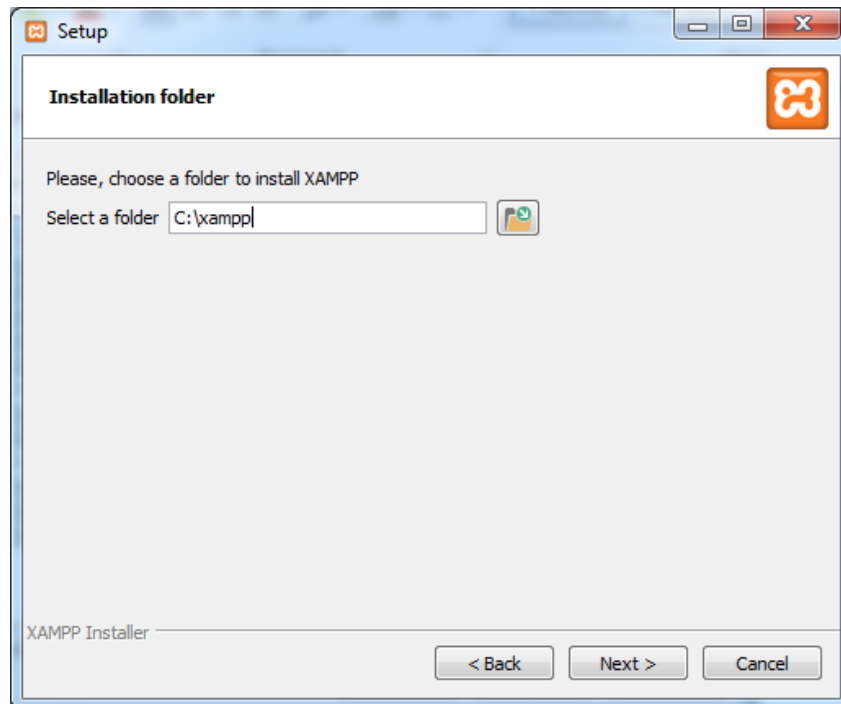
Silahkan klik tombol **OK** untuk melanjutkan. Jendela awal instalasi akan muncul, Klik **Next**.



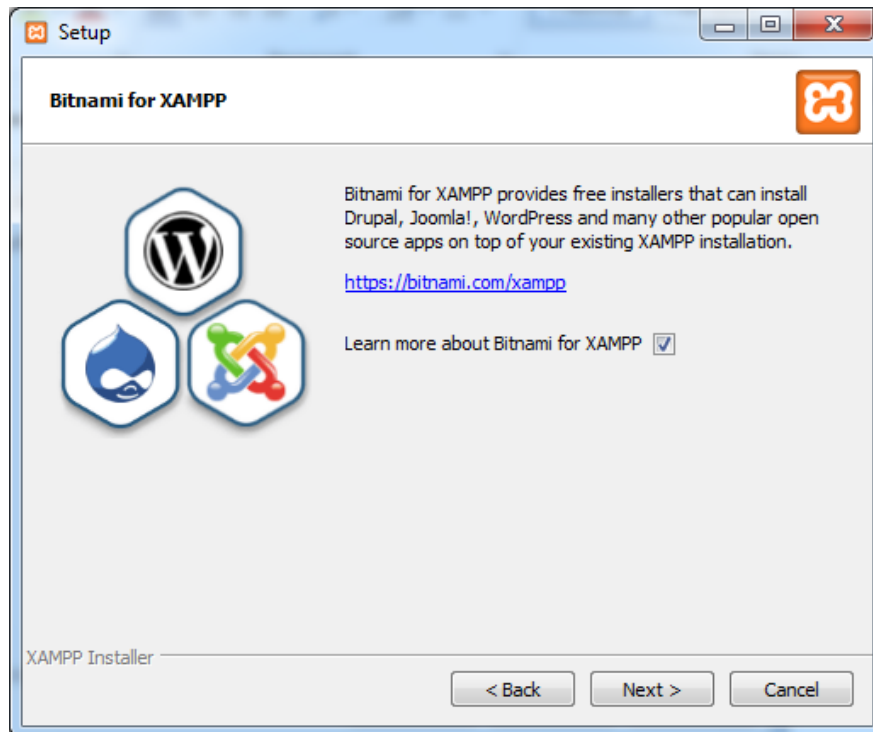
Jendela berikutnya adalah “**Select Component**”. Pada bagian ini kita bisa memilih aplikasi apa saja yang akan diinstall. Dalam tahap ini kita akan membiarkan semua pilihan. Klik **Next** untuk melanjutkan.



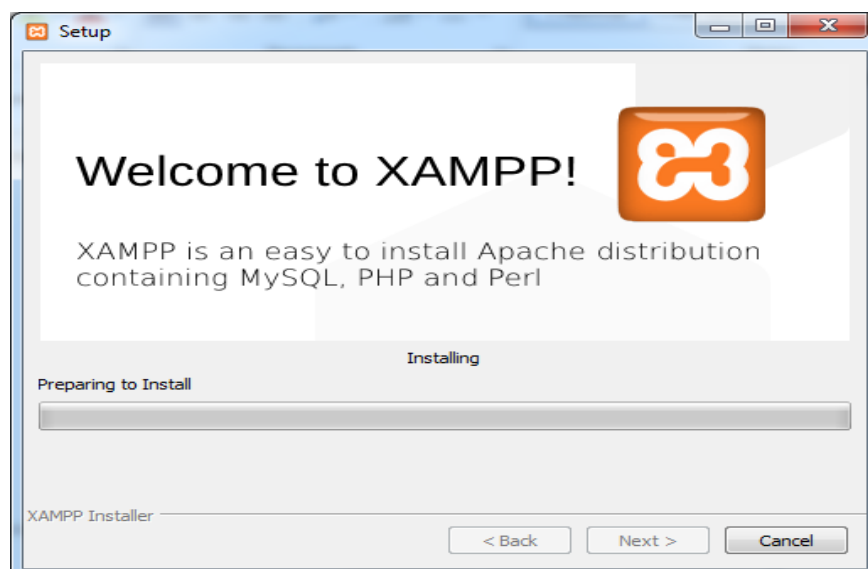
Jendela selanjutnya adalah “**Installation Folder**”. Dalam bagian ini, kita bisa mengubah lokasi dimana file-file XAMPP akan disimpan. Anda bebas menentukan lokasi ini. Sebagai contoh, kita akan meletakkan file XAMPP di **c:\xampp**. Lanjutkan dengan men-klik tombol **Next**.



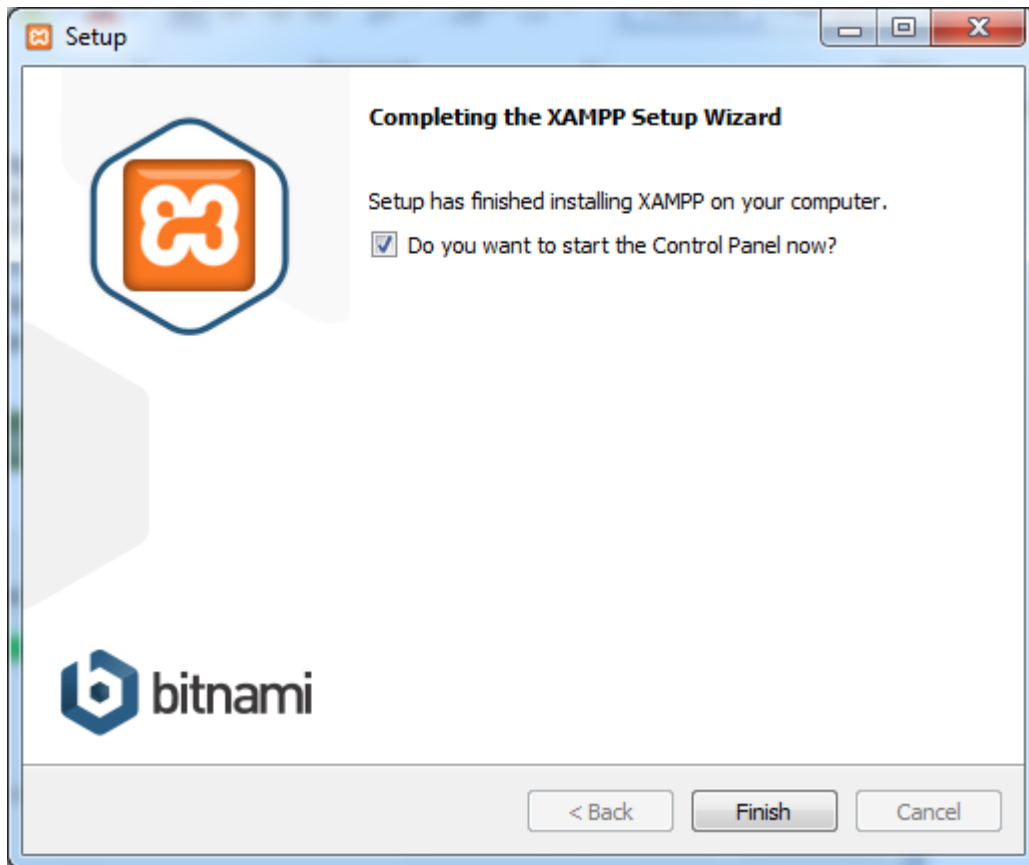
Tampilan berikutnya adalah “**Bitnami for XAMPP**”. XAMPP menawarkan **Bitnami** sebagai cara cepat menginstall CMS seperti *wordpress*, *joomla* dan *drupal*. Tetapi kita akan menginstall wordpress secara manual, sehingga hapus pilihan “*learn more about Bitnami for XAMPP*”, kemudian klik **Next**.



Jendela berikutnya adalah konfirmasi untuk mulai menginstall XAMPP, klik **Next**, dan XAMPP akan memulai proses instalasi beberapa saat.



Jika jendela “**Completing the XAMPP Setup Wizard**” telah tampil, maka proses instalasi XAMPP telah selesai. Pada bagian ini kita akan langsung mencoba aplikasi XAMPP, sehingga biarkan pilihan check list “*Do you want to start the Control Panel now?*”, kemudian klik **Finish**.



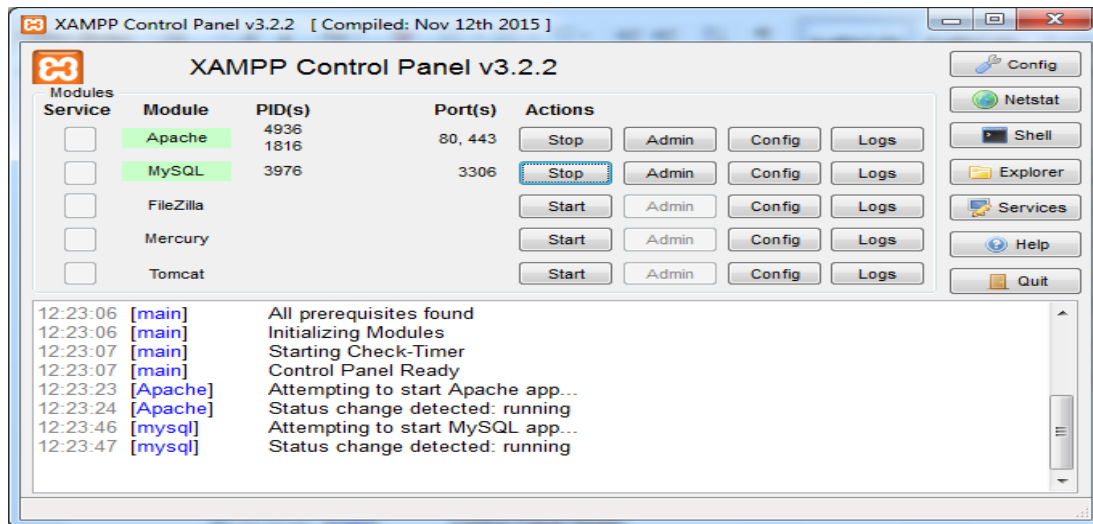
Klik Save untuk memilih bahasa yang akan digunakan.

## 1.4 Menguji Instalasi XAMPP

Jika anda membiarkan pilihan “*Do you want to start the Control Panel now?*” pada jendela terakhir proses instalasi XAMPP, maka akan tampil jendela **XAMPP Control Panel**.

Sesuai dengan namanya, Jendela **XAMPP Control Panel** ini adalah jendela yang digunakan untuk mengontrol apa saja modul XAMPP yang akan atau sedang dijalankan

ntuk menguji instalasi XAMPP, silahkan klik tombol **START** pada modul **Apache** dan **MySQL**. Jika tidak ada masalah, akan tampil warna hijau pada bagian modul ini, seperti tampilan dibawah:

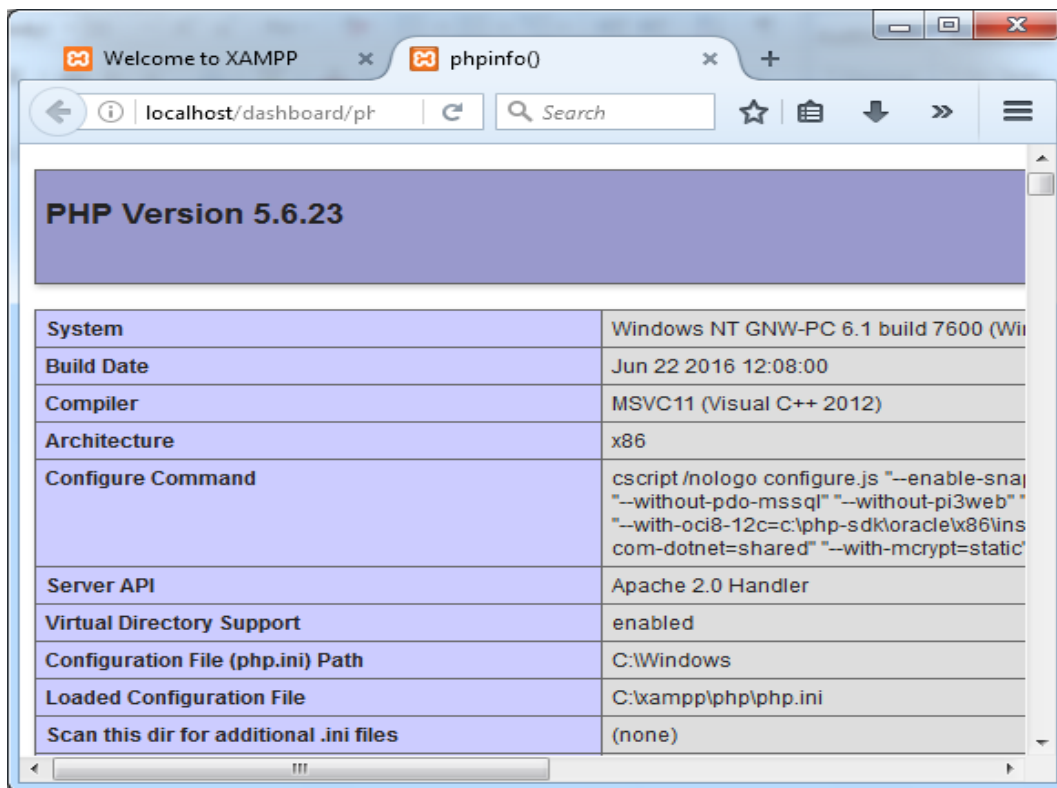


Selanjutnya, buka web browser dan ketikkan alamat **localhost** pada *address bar*, kemudian tekan **enter**. Jika tampil jendela pembuka XAMPP, maka semuanya telah terinstall dengan baik.

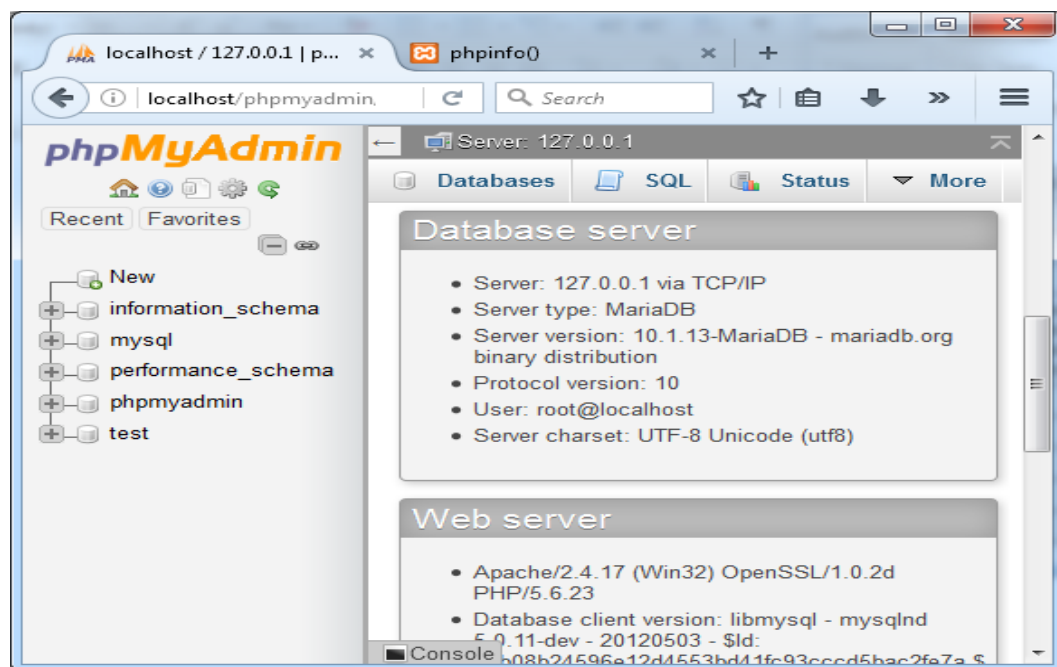


Klik link `phpinfo()` untuk melihat versi PHP.



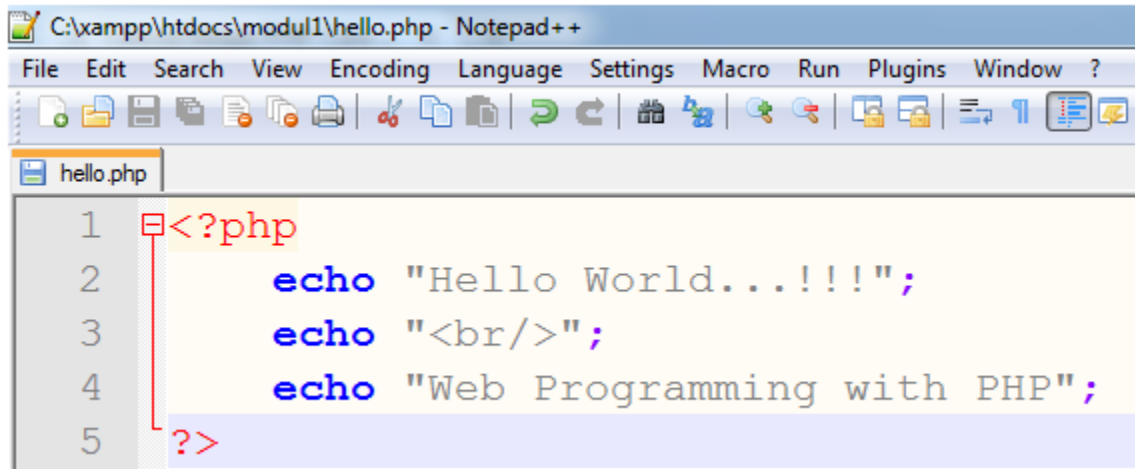


Klik link phpmyadmin untuk melihat versi DBMS yang kita gunakan.



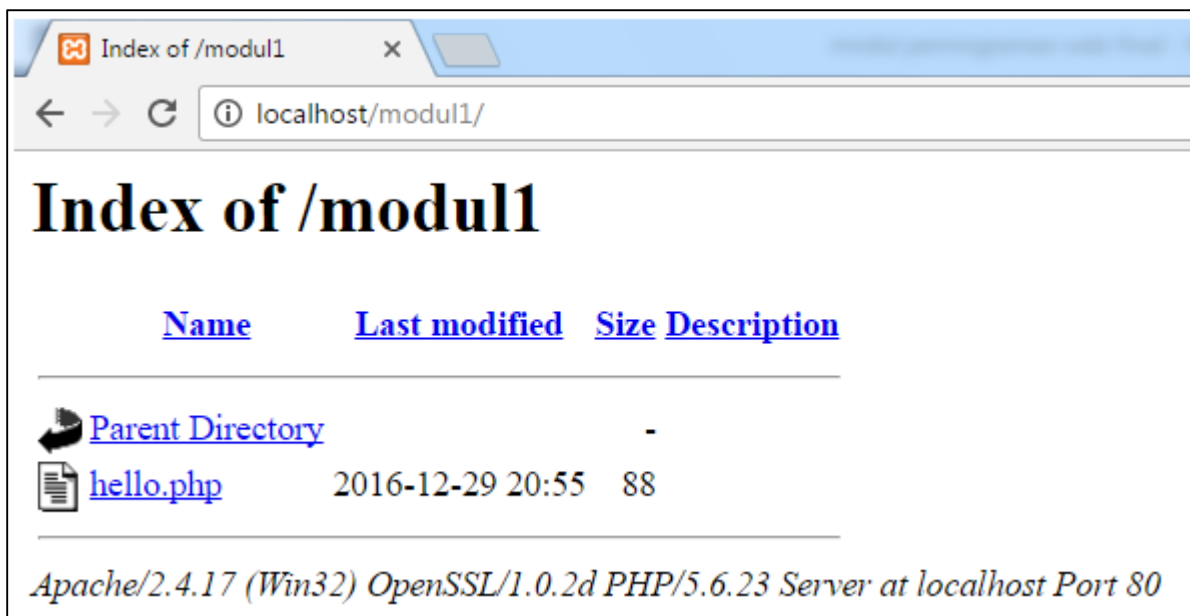
## 1.5 Menyiapkan folder dan Membuat Program Sederhana

Secara default Anda harus menyimpan program PHP di `c:\xampp\htdocs`, buatlah folder dengan nama **modul1**. Gunakan editor, misalnya menggunakan Notepad++ atau editor yang lain lalu ketik kode program PHP di bawah ini, simpan dengan nama **hello.php**.

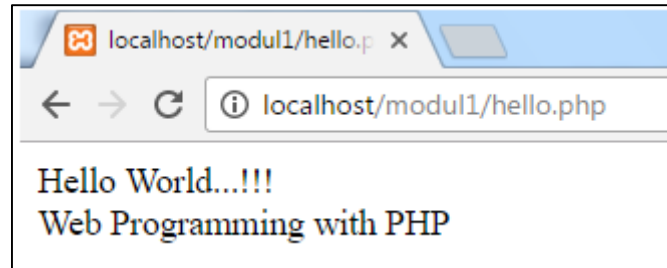


```
C:\xampp\htdocs\modul1\hello.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
hello.php
1 <?php
2     echo "Hello World...!!!";
3     echo "<br/>";
4     echo "Web Programming with PHP";
5 ?>
```

Sebelum menjalankan file PHP, pastikan web server dalam keadaan running, jalankan file `hello.php` melalui web browser dengan mengetikkan URL seperti di bawah ini:



Klik file `hello.php`, hasilnya seperti di bawah ini.



## 1.6 Variabel dan Konstanta

Sama seperti **variabel** dalam bahasa pemrograman lainnya, variabel dalam PHP digunakan untuk menampung nilai inputan dari user, atau nilai yang kita definisikan sendiri. Namun PHP memiliki beberapa aturan tentang cara penggunaan dan penulisan **variabel**.

### 1.6.1 Aturan Penulisan Variabel dalam PHP

#### 1. Penulisan variabel harus diawali dengan tanda \$

**Variabel** di dalam PHP harus diawali dengan **dollar sign** atau **tanda dollar** (\$).

Setelah tanda \$, sebuah **variabel** PHP harus diikuti dengan karakter pertama berupa huruf atau *underscore* (\_), kemudian untuk karakter kedua dan seterusnya bisa menggunakan huruf, angka atau *underscore* (\_). Dengan aturan tersebut, variabel di dalam PHP *tidak bisa* diawali dengan angka.

Berikut adalah contoh penulisan variabel yang benar dalam PHP:

```
1 <?php
2     $i;
3     $nama;
4     $Umur;
5     $_lokasi_memori;
6     $ANGKA_MAKSIMUM;
7 ?>
```

#### 2. Variabel dalam PHP bersifat case sensitif

PHP membedakan variabel yang ditulis dengan huruf besar dan kecil (bersifat **case sensitif**), sehingga **\$belajar** tidak sama dengan **\$Belajar** dan **\$BELAJAR**, ketiganya akan dianggap sebagai variabel yang berbeda.

Untuk menghindari kesalahan program yang dikarenakan salah merujuk **variabel**, disarankan menggunakan huruf kecil untuk seluruh nama **variabel**.

```

1  <?php
2      $nama="Naufal";
3      echo $Nama; // Notice: Undefined variable: Nama
4  ?>

```

Dalam contoh di atas, PHP mengeluarkan error karena tidak menemukan variabel `$Nama`.

### 3. Cara Memberikan Nilai kepada Variabel

Sama seperti sebagian besar bahasa pemrograman lainnya, untuk memberikan nilai kepada sebuah **variabel**, PHP menggunakan tanda **sama dengan** (`=`). Operator 'sama dengan' ini dikenal dengan istilah *Assignment Operators*.

Perintah pemberian nilai kepada sebuah variabel disebut dengan *assignment*. Jika *variabel* tersebut belum pernah digunakan, dan langsung diberikan nilai awal, maka disebut juga dengan proses *inisialisasi*.

Berikut contoh cara memberikan nilai awal (*inisialisasi*) kepada variabel:

```

1  <?php
2      $nama = "Naufal";
3      $umur = 19;
4      $pesan = "Saya sedang belajar PHP";
5  ?>

```

### 4. Variabel dalam PHP tidak memerlukan deklarasi terlebih dahulu

Jika Anda pernah mempelajari bahasa pemrograman desktop seperti **Pascal**, **C**, **C++**, dan **Java**, di dalam bahasa pemrograman tersebut, sebuah variabel harus dideklarasikan terlebih dahulu sebelum digunakan.

Namun di dalam PHP, **variabel** tidak perlu dideklarasikan terlebih dahulu. Anda bebas membuat variabel baru di tengah-tengah kode program, dan langsung menggunakannya tanpa di deklarasi terlebih dahulu.

```

1  <?php
2      $nama="Naufal";
3      echo $nama;
4  ?>

```

## 5. Variabel dalam PHP tidak bertipe

Dalam kelompok bahasa pemrograman, PHP termasuk **Loosely Type Language**, yaitu jenis bahasa pemrograman yang variabelnya tidak terikat pada sebuah tipe tertentu.

Hal ini berbeda jika dibandingkan dengan bahasa pemrograman desktop seperti **Pascal** atau **C**, dimana jika anda membuat sebuah variabel bertipe **integer**, maka variabel itu hanya bisa menampung nilai angka, dan anda tidak akan bisa mengisinya dengan huruf.

Di dalam PHP, setiap variabel bebas diisi dengan nilai apa saja, seperti contoh berikut:

```
1 <?php
2     $a = 17; // nilai variabel a berisi angka (integer)
3     $a = "aku"; // nilai variabel a diubah menjadi kata (string)
4     $a = 17.42; // nilai variabel a diubah menjadi desimal (float)
5 ?>
```

## 6. Variabel Sistem PHP (Predefined Variables)

**Predefined Variables** atau terjemahan bebasnya **Variabel Sistem PHP**, adalah beberapa variabel yang telah didefinisikan secara sistem oleh PHP, dan kita sebaiknya tidak membuat variabel dengan nama yang sama.

Beberapa contoh **Predefined Variables** dalam PHP adalah:

\$GLOBALS, \$\_SERVER, \$\_GET, \$\_POST, \$\_FILES, \$\_COOKIE, \$\_SESSION, \$\_REQUEST, \$\_ENV, \$php\_errormsg, \$HTTP\_RAW\_POST\_DATA, \$http\_response\_header, \$argc, \$argv, \$this.

Daftar list **Predefined Variables** tersebut diambil dari **manual PHP** di <http://www.php.net/reserved.variables>, di dalam manual tersebut juga dijelaskan bahwa mungkin masih terdapat beberapa variabel sistem PHP selain list di atas, hal ini tergantung dengan jenis web server, versi PHP yang digunakan, dan beberapa faktor lainnya. Namun kebanyakan variabel sistem PHP menggunakan tanda **\$\_** pada awal nama variabel, namun tidak selalu.

### 1.6.2 Cara Menampilkan Nilai Variabel

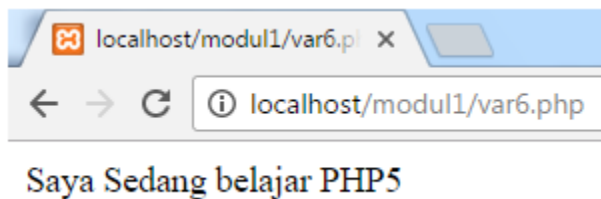
Untuk menampilkan nilai atau isi dari variabel, kita tinggal menampilkannya dengan perintah **echo** atau **print**, seperti berikut ini:

```

1 <?php
2     $a='Saya Sedang belajar PHP';
3     $b=5;
4
5     print $a;
6     echo $b;
7 ?>

```

Hasil yang didapat adalah:



### 1.6.3 Pengertian Konstanta PHP

Dalam bahasa pemrograman, **Konstanta (constant)** adalah suatu lokasi penyimpanan (dalam memory) yang berisikan nilai yang sifatnya tetap dan tidak bisa diubah sepanjang program berjalan. Berbeda dengan **variabel** yang isi/nilainya dapat diubah bahkan dihapus selama program berjalan, sebuah **konstanta** jika telah diberikan nilai, tidak dapat diubah lagi dalam kode program. Hal ini sesuai dengan namanya, yakni **konstant**.

#### Aturan Penulisan Konstanta PHP

##### 1. Cara Pendefinisian Konstanta dalam PHP

Jika **variabel** di dalam PHP dibuat dengan menambahkan tanda dollar, seperti: **\$nama**. Untuk membuat konstanta PHP menyediakan 2 cara:

1. Menggunakan kata kunci (keyword) **const**.
2. Menggunakan fungsi **define**.

Untuk mendefinisikan konstanta dengan kata kunci **const**, caranya mirip dengan menambahkan nilai kepada sebuah variabel, namun didahului kata **const**. Berikut adalah contoh penulisannya:

```

1 <?php
2     const situs = "https://igracias.telkomuniversity.ac.id/";
3     echo situs; // https://igracias.telkomuniversity.ac.id/
4 ?>

```

Jika menggunakan fungsi **define**, fungsi ini membutuhkan 2 nilai, yakni nama konstanta dan nilainya. Seperti contoh berikut ini:

```

1 <?php
2     define("situs", " https://igracias.telkomuniversity.ac.id ");
3     echo situs; // https://igracias.telkomuniversity.ac.id
4 ?>

```

## 1.7 Tipe Data

### 1.7.1 Tipe Data Integer

Tipe data *integer* adalah *tipe data yang berupa angka bulat* seperti: 1, 22, dan -172. Tipe data *integer* umum digunakan untuk data dengan angka bulat, seperti harga barang, jumlah stock dan jumlah mahasiswa. Jika data yang kita miliki kemungkinan akan mengandung pecahan, maka tipe data yang digunakan adalah *float*.

Nilai integer dapat bernilai **positif** (+) maupun **negatif** (-). Jika tidak diberi tanda, maka diasumsikan nilai tersebut adalah **positif**.

Berikut contoh penulisan bilangan integer dalam PHP:

```

1 <?php
2     $umur=21;
3     $harga=15000;
4     $rugi=-500000;
5
6     echo $umur; //21
7     echo "<br />";
8     echo $harga; //15000
9     echo "<br />";
10    echo $rugi; //-500000
11 ?>

```

Untuk variabel dengan angka integer, kita bisa melakukan operasi matematis seperti penambahan, pengurangan, pembagian dan lain-lain, seperti contoh berikut ini:

```

1 <?php
2     $a=14;
3     $b=16;
4     $c= $a + $b;
5     echo $c; // 30
6
7     $d=$a * $b;
8     echo $d; // 224
9 ?>

```

Karena PHP tidak memerlukan pendeklarasian *variabel*, maka ketika sebuah variabel berisi angka bulat, maka secara otomatis variabel tersebut di sebut sebagai *variabel integer*.

Jangkauan angka integer bergantung kepada kemampuan komputasi komputer, namun biasanya dimulai dari **-2,147,483,648** sampai **+2,147,483,647**, atau **32bit**. Jika terdapat menungkinan angka yang dihasilkan dari kode program kita berada diluar jangkauan ini, sebaiknya menggunakan tipe data *float*.

Untuk mengetahui nilai maksimal tipe data integer pada komputer, PHP menyediakan konstanta **PHP\_INT\_MAX**. Berikut adalah hasil nilai **PHP\_INT\_MAX** yang dijalankan:

```
1 <?php
2     print PHP_INT_MAX; // 2147483647
3 ?>
```

### 1.7.2 Tipe Data Float

PHP mendukung **2 cara** penulisan tipe data **float**, yang pertama yaitu penulisan desimal sehari-hari, seperti 0.17 atau 9.47 dan yang kedua berupa penulisan format **scientific notation**, seperti 0.314E1, atau 12.0E-3.

Berikut contoh penulisan bilangan float dalam PHP:

```
1 <?php
2     $angka_float1= 0.78;
3     $angka_float2= 14.99;
4     $angka_scientific1=0.314E1;
5     $angka_scientific2=0.3365E-3;
6
7     echo $angka_float1; // 0.78
8     echo "<br />";
9     echo $angka_float2; //14.99
10    echo "<br />";
11    echo $angka_scientific1; //3.14
12    echo "<br />";
13    echo $angka_scientific2; //0.0003365
14 ?>
```

Sama seperti tipe data *integer*, variabel dengan tipe data *float* juga dapat melakukan operasi numerik seperti penambahan, pembagian, perkalian, dan lain-lain. Berikut adalah contoh operasi matematis dengan tipe data float:



```

1 <?php
2     $a=10.66;
3     $b=12.4;
4     $c= $a + $b;
5     echo $c; // 23.06
6
7     $d=$a / $b;
8     echo $d; // 0.85967741935484
9 ?>

```

### 1.7.3 Tipe Data String

#### 1. Penulisan Tipe Data String dengan Single Quoted

Penulisan tipe data string menggunakan **single quoted** atau **tanda petik satu** (karakter `'`) merupakan cara penulisan **string** yang paling sederhana. Kita tinggal membuat sebuah kata atau kalimat, dan menambahkan **tanda petik satu** di awal dan akhir kalimat.

Untuk **string** yang didalamnya juga terdapat tanda petik satu, kita harus mendahuluinya dengan karakter **backslash** (`\`) agar tidak dianggap sebagai penutup **string**. Dan jika di dalam **string** anda ingin menulis tanda **backslash**, kita harus menulisnya dengan 2 kali (`\\`).

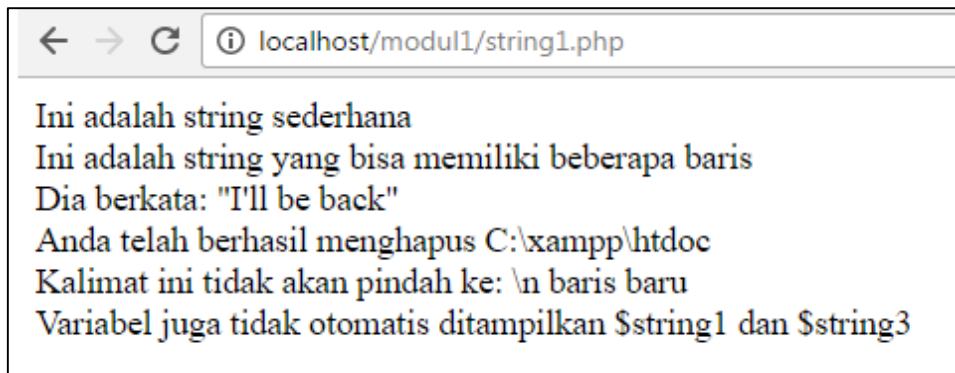
Berikut adalah contoh penulisan tipe data **string** menggunakan metode **single quoted**:

```

1 <?php
2 $string1='Ini adalah string sederhana';
3 $string2='Ini adalah string yang bisa memiliki beberapa baris';
4 $string3='Dia berkata: "I\'ll be back"';
5 $string4='Anda telah berhasil menghapus C:\\xampp\\htdocs';
6 $string5='Kalimat ini tidak akan pindah ke: \n baris baru';
7 $string6='Variabel juga tidak otomatis ditampilkan $string1 dan $string3';
8
9 echo $string1; echo "<br>";
10 echo $string2; echo "<br>";
11 echo $string3; echo "<br>";
12 echo $string4; echo "<br>";
13 echo $string5; echo "<br>";
14 echo $string6;
15 ?>

```

Jika contoh tersebut dijalankan, berikut tampilannya di browser:



Pada contoh di atas terdapat beberapa karakter khusus seperti “, \n, dan variabel yang dimulai dengan tanda dollar (\$). Ketiga karakter khusus ini ditampilkan secara karakter aslinya ke dalam browser.

## 2. Penulisan Tipe Data String dengan Double Quoted

Cara kedua dalam penulisan tipe data **string** dalam PHP adalah dengan menggunakan **Double Quoted** atau **tanda petik dua** (karakter “). Walaupun seperti tidak ada perbedaan dengan menggunakan *single quote*, hasil yang di dapat akan sangat berbeda.

Dengan **double quoted**, PHP akan memproses karakter-karakter khusus seperti *carriage return* (\n), dan karakter tab (\t) dan juga memproses setiap **variabel** (yang ditandai dengan tanda \$ didepan kata).

Di karenakan metode **double quoted** melakukan pemrosesan terlebih dahulu, maka untuk menampilkan karakter khusus seperti *tanda petik* (karakter ‘), *tanda dollar* (karakter \$) dan tanda-tanda khusus lainnya, kita harus menggunakan **backslash** (karakter \).

Sebagai contoh penggunaan **double quoted string**, kita akan menggunakan contoh yang sama dengan **single quoted string**, agar dapat dilihat perbedaannya:

```
1 <?php
2 $string1="Ini adalah string sederhana";
3 $string2="Ini adalah string yang bisa memiliki beberapa baris";
4 $string3="Dia berkata: \"I'll be back\"";
5 $string4="Anda telah berhasil menghapus C:\\xampp\\htdocs";
6 $string5="Kalimat ini akan akan pindah ke: \\n baris baru";
7 $string6="Variabel akan otomatis ditampilkan: $string1 dan $string3";
8
9 echo $string1; echo "<br \>";
10 echo $string2; echo "<br \>";
11 echo $string3; echo "<br \>";
12 echo $string4; echo "<br \>";
13 echo $string5; echo "<br \>";
14 echo $string6;
15 ?>
```

Perhatikan perbedaannya pada hasil **\$string3**, **\$string5** dan **\$string6**.

Pada **\$string3**, kita harus mem-**backslash** tanda petik dua karena itu merupakan karakter khusus dalam **double quoted string**.

Pada **\$string5**, tanda **\n** yang merupakan karakter khusus untuk baris baru, tapi karena kita menampilkannya di browser, karakter ini tidak akan terlihat, tetapi jika kita menulis hasil string ini kedalam sebuah file text, kalimat tersebut akan terdiri dari 2 baris.

Pada **\$string6**, terlihat bahwa string dengan petik dua akan memproses variabel **\$string1** dan **\$string3** sehingga tampil hasilnya di **web browser**. Fitur ini akan sangat bermanfaat jika kita sering menampilkan variabel didalam sebuah **string**.

#### 1.7.4 Tipe Data Boolean

Tipe **data boolean** adalah tipe data paling sederhana dalam PHP dan juga dalam bahasa pemrograman lainnya. Tipe data ini hanya memiliki 2 nilai, yaitu **true** (benar) dan **false** (salah). Tipe data **boolean** biasanya digunakan dalam **operasi logika** seperti kondisi **if**, dan perulangan (**looping**). Untuk penggunaan tipe data boolean akan kita pelajari pada waktu membahas tentang struktur pemrograman PHP.

Cara Penulisan Boolean dalam PHP

Penulisan **boolean** cukup sederhana, karena hanya memiliki 2 nilai, yakni **true** atau **false**. Penulisan **true** atau **false** ini bersifat **non-case sensitif**, sehingga bisa ditulis sebagai **true**, **True** atau **TRUE**.

Berikut adalah contoh penulisan tipe data Boolean:

```
<?php
    $benar=true;
    $salah=false;

    echo "benar = $benar, salah = $salah";
    // hasil output: benar = 1, salah =
?>
```

Jika anda menjalankan contoh kode PHP di atas, variabel **\$benar** akan ditampilkan dengan angka 1, sedangkan variabel **\$false** ditampilkan dengan string kosong (tanpa output). Hal ini karena jika ditampilkan menggunakan echo, tipe data boolean “dipaksa” berganti dengan tipe data string.

#### 1.7.5 Pengertian Tipe Data Array

**Array** (atau *larik* dalam bahasa indonesia) bukanlah tipe data dasar seperti **integer** atau **boolean**, **Array** adalah sebuah tipe data bentukan yang terdiri dari kumpulan tipe data lainnya. Menggunakan array akan memudahkan dalam membuat kelompok data, serta menghemat penulisan dan penggunaan **variabel**.

### 1.7.6 Cara Penulisan Array dalam PHP

PHP mendukung beberapa **cara penulisan array**, salah satunya dengan menggunakan **konstruktor array PHP** (*array language construct*) sebagai berikut:

```
1 | $nama_variabel = array(  
2 |     key => value,  
3 |     key2 => value2,  
4 |     key3 => value3,  
5 |     ...  
6 | )
```

Komponen **array** terdiri dari pasangan kunci (**key**) dan nilai (**value**). **Key** adalah penunjuk posisi dimana **value** disimpan. Perhatikan juga bahwa PHP menggunakan *tanda panah* (**=>**) untuk memberikan nilai kepada key.

Dalam mengakses nilai dari **array**, kita menggunakan kombinasi **\$nama\_variabel** dan nilai **key**-nya, dengan penulisan sebagai berikut:

```
$nama_variabel[key];
```

Berikut adalah contoh pengaksesan array dalam PHP:

```
1 | <?php  
2 | //pembuatan array  
3 | $nama = array(  
4 |     1=>"Andri",  
5 |     2=>"Joko",  
6 |     3=>"Sukma",  
7 |     4=>"Rina",  
8 |     5=>"Sari");  
9 |  
10 | //cara akses array  
11 | echo $nama[1]; //Andri  
12 | echo "<br />";  
13 | echo $nama[2]; //Joko  
14 | echo "<br />";  
15 | echo $nama[3]; //Sukma  
16 | ?>
```

Dalam contoh di atas, menggunakan angka **integer** sebagai **key** (1,2,3...) dan **string** sebagai **value** (Andri, Joko, Sukma, ...).

Selain mendefenisikan key secara langsung, PHP juga memperbolehkan penulisan array **tanpa key**, dan key itu secara otomatis akan diurutkan dari nilai 0, 1, 2, dst.

Berikut adalah contoh pendefinisian array tanpa key:

```
1 <?php
2 // pembuatan array
3 $nama = array("Andri","Joko","Sukma","Rina","Sari");
4
5 // pengaksesan array
6 echo $nama[1]; //Joko
7 echo "<br />";
8 echo $nama[2]; //Sukma
9 echo "<br />";
10 echo $nama[3]; //Rina
11 ?>
```

Perhatikan bahwa sekarang, **index** atau **key** dari array dimulai dari angka **0**, bukan 1. sehingga **\$nama[1]** berisi *Joko*. nama *Andri* berada di **\$nama[0]**. Dalam penggunaan array di dalam PHP, konsep “key” array dimulai dari angka 0 ini sangat penting untuk dipahami

Selain menggunakan angka, **key** dalam PHP dapat berisi **string** atau **boolean**. Sedangkan untuk value dapat menyimpan berbagai tipe data seperti *integer*, *float*, *string*, *boolean*, bahkan *array* lainnya. Array seperti ini disebut juga dengan istilah “**associate array**”.

Berikut contoh penggunaan array dengan kombinasi tipe data.

```
1 <?php
2 // pembuatan array
3 $coba = array (
4     2=>"Andri",
5     "dua"=>"2",
6     'tiga'=>3,
7     true=>true,
8     9=>"sembilan",);
9
10 // pengaksesan array
11 echo $coba[2]; //Andri
12 echo "<br />";
13 echo $coba["dua"]; //2
14 echo "<br />";
15 echo $coba['tiga']; //3
16 echo "<br />";
17 echo $coba[true]; //1 (true di konversi menjadi 1)
18 echo "<br />";
19 echo $coba[9]; // sembilan
20 ?>
```

Dari contoh di atas, kita membuat array **\$coba** dengan menggunakan berbagai tipe data untuk **key** dan **value**, yaitu dengan tipe data **integer**, **string**, dan **boolean**. Namun jika **key** di defenisikan dengan tipe data **boolean** seperti pada baris ke-6, maka secara otomatis PHP akan mengkonversinya menjadi 1.

## 1.8 Operator Aritmatika

Penggunaan **operator aritmatika** di dalam PHP relatif mudah, karena kita telah terbiasa dengan operator ini.

Contoh	Nama Operator	Hasil
+\$a	Positif	Nilai positif dari \$a
-\$a	Negatif	Nilai negatif dari \$a
\$a + \$b	Penambahan	Total dari \$a dan \$b
\$a - \$b	Pengurangan	Selisih dari \$a dan \$b
\$a * \$b	Perkalian	Hasil Kali dari \$a dan \$b
\$a / \$b	Div/Pembagian	Hasil Bagi dari \$a dan \$b
\$a % \$b	Mod/Sisa hasil bagi	Sisa dari pembagian \$a / \$b

Berikut adalah contoh kode program, cara penggunaan operator aritmatika dalam PHP:

```
1 <?php
2     $hasil1= -3;
3     $hasil2=3+5;
4     $hasil3=8-4.5;
5     $hasil4=2*5;
6     $hasil5=3+8/5-3;
7     $hasil6=10 % 4;
8
9     echo "\$hasil1:"; var_dump($hasil1); // $hasil1:int(-3)
10    echo "<br \>";
11    echo "\$hasil2:"; var_dump($hasil2); // $hasil2:int(8)
12    echo "<br \>";
13    echo "\$hasil3:"; var_dump($hasil3); // $hasil3:float(3.5)
14    echo "<br \>";
15    echo "\$hasil4:"; var_dump($hasil4); // $hasil4:int(10)
16    echo "<br \>";
17    echo "\$hasil5:"; var_dump($hasil5); // $hasil5:float(1.6)
18    echo "<br \>";
19    echo "\$hasil6:"; var_dump($hasil6); // $hasil6:int(2)
20 ?>
```

Pada kode program di atas, kita menggunakan fungsi `var_dump()` untuk menampilkan hasil perhitungan, sehingga kita bisa melihat tipe data dari masing-masing variabel.

Dari hasil `var_dump()`, terlihat bahwa variabel `$hasil3` dan `$hasil5` bertipe `float`. Hal ini dikarenakan perhitungan aritmatika pada baris ke-4 dan ke-6 menghasilkan angka *desimal*, sehingga secara otomatis variabel tersebut tidak dapat ditampung sebagai `integer`, melainkan harus `float`.

Namun jika hasil operasi matematis tersebut menghasilkan bilangan bulat, PHP akan menyimpannya sebagai tipe data `int` (`integer`), seperti variabel `$hasil1`, `$hasil2`, `$hasil4` dan `$hasil6`.

Pada perhitungan baris ke-6 yaitu persamaan `$hasil5=3+8/5-3`, hasilnya adalah `1.6`. Hal ini karena operator pembagian memiliki prioritas lebih tinggi daripada operator tambah dan kurang. Operasi `3+8/5-3` dikerjakan oleh PHP sebagai `(3+(8/5))-3`. Namun untuk hal ini, disarankan menggunakan tanda kurung secara tertulis agar memudahkan dalam membaca alur program, dari pada bergantung kepada aturan prioritas operator PHP.

## 1.9 Operator String

Dalam PHP, hanya terdapat 1 jenis operator **String**, yakni operasi penyambungan (**concatenation**) string. Operator ini menggunakan karakter titik (`.`).

Operator penyambungan string ini membutuhkan 2 inputan yang bertipe data **string**. Hasil dari operator ini adalah sebuah string yang terdiri dari sambungan kedua string tersebut.

### Cara Penggunaan Operator String di dalam PHP

Berikut adalah contoh kode program cara penggunaan operator string dalam PHP:

```
1 <?php
2     $a = "Hello ";
3     $hasil = $a . "World!";
4     echo $hasil; // Hello World!
5     echo "<br />";
6
7     $a = "belajar ";
8     $b = "PHP ";
9     $c = "di Lab. Komp";
10    $hasil= "Saya sedang ".$a.$b.$c;
11    echo $hasil; // Saya sedang belajar PHP di Lab. Komp
12 ?>
```

Pada kode program di atas, kita menyambung beberapa string sederhana menggunakan operator concatenation (tanda .).

### Cara Alternatif: Penyambung string dengan kurung kurawal { }

Didalam PHP, tanda **kurung kurawal** (karakter { dan }) untuk variabel bisa berfungsi sebagai **penyambung string**. Contoh kode program diatas dapat juga ditulis menjadi:

```
1 <?php
2     $a = "Hello ";
3     $hasil = "{$a} World!";
4     echo $hasil; // Hello World!
5     echo "<br />";
6
7     $a = "belajar ";
8     $b = "PHP ";
9     $c = "di Lab. Komp";
10    $hasil= "Saya sedang {$a}{$b}{$c}";
11    echo $hasil; // Saya sedang belajar PHP di Lab. Komp
12 ?>
```

## 1.10 Operator Logika

**Operator Logika** adalah operator yang digunakan untuk membandingkan 2 kondisi logika, yaitu logika benar (**TRUE**) dan logika salah (**FALSE**). **Operator logika** sering digunakan untuk **kodisi IF**, atau untuk keluar dari proses perulangan (**looping**).

### Jenis-jenis Operator Logika dalam PHP

Jenis-jenis operator logika dalam PHP dapat dilihat dari tabel berikut:

Contoh	Nama Operator	Hasil
\$a and \$b	AND	TRUE jika \$a dan \$b sama-sama bernilai TRUE.
\$a && \$b	AND	TRUE jika \$a dan \$b sama-sama bernilai TRUE.
\$a or \$b	OR	TRUE jika salah satu dari \$a atau \$b adalah TRUE.
\$a    \$b	OR	TRUE jika salah satu dari \$a atau \$b adalah TRUE.
\$a xor \$b	XOR	TRUE jika salah satu dari \$a atau \$b adalah TRUE, tetapi bukan keduanya.
! \$a	NOT	TRUE jika \$a=FALSE.



Perbedaan dari operator **AND** dengan **&&**, dan **OR** dengan **||** terkait dengan cara penulisan dan aturan “*kekuatan*” operator. Operator **&&** dan **||** memiliki “*kekuatan*” lebih tinggi dari pada **AND** dan **OR**, sehingga baris perintah: **\$a AND \$b || \$c**, akan dieksekusi oleh PHP menjadi **\$a AND (\$b || \$c)**.

### Cara Penggunaan Operator Logika di dalam PHP

Berikut adalah contoh kode program, cara penggunaan operator logika dalam PHP:

```
1 <?php
2     $hasil1 = true and false;
3     echo 'hasil1 = ';
4     echo var_dump($hasil1)."<br/>"; // $hasil1 = bool(true)
5
6     $hasil2 = (true and false);
7     echo 'hasil2 = ';
8     echo var_dump($hasil2)."<br/>"; // $hasil2 = bool(false)
9
10    $hasil3 = (true xor false);
11    echo 'hasil3 = ';
12    echo var_dump($hasil3)."<br/>"; // $hasil3 = bool(true)
13
14    $hasil4 = (false or true && false);
15    echo 'hasil4 = ';
16    echo var_dump($hasil4)."<br/>"; // $hasil4 = bool(false)
17
18    $a=true;
19    $b=false;
20    $hasil5 = ($a and $b || $a or b);
21    echo 'hasil5 = ';
22    echo var_dump($hasil5); // $hasil5 = bool(true)
23
```

## 1.11 Operator Perbandingan

Sesuai dengan namanya, operator perbandingan membandingkan nilai dari 2 **operand**. Hasilnya selalu salah satu dari **TRUE** atau **FALSE**. Hasil perbandingan akan bernilai **TRUE** jika kondisi perbandingan tersebut benar, atau **FALSE** jika kondisinya salah.

Operand untuk operator perbandingan ini bisa berupa tipe data angka (**integer** atau **float**), maupun bertipe **string**. Operator perbandingan akan memeriksa **nilai** dan (untuk beberapa operator) **juga tipe data** dari **operand**.

### Jenis-jenis Operator Perbandingan dalam PHP

Jenis-jenis dari **operator perbandingan** dalam PHP dapat dilihat dari tabel dibawah ini:

Contoh	Nama Operator	Hasil
\$a == \$b	Sama dengan	TRUE jika \$a sama dengan \$b (tanpa melihat tipe data)
\$a === \$b	Identik dengan	TRUE jika \$a sama dengan \$b, dan memiliki tipe data yang sama
\$a != \$b	Tidak sama dengan	TRUE jika \$a tidak sama dengan \$b (tanpa melihat tipe data)
\$a <> \$b	Tidak sama dengan	TRUE jika \$a tidak sama dengan \$b (tanpa melihat tipe data)
\$a !== \$b	Tidak identik dengan	TRUE jika \$a tidak sama dengan \$b, dan memiliki tipe data yang tidak sama
\$a < \$b	Kurang dari	TRUE jika \$a kurang dari \$b
\$a > \$b	Lebih dari	TRUE jika \$a lebih dari \$b
\$a <= \$b	Kurang dari atau sama dengan	TRUE jika \$a kurang dari atau sama dengan \$b
\$a >= \$b	Lebih dari atau sama dengan	TRUE jika \$a lebih dari atau sama dengan \$b

## Cara Penggunaan Operator Perbandingan di dalam PHP

Berikut adalah beberapa contoh penggunaan **operator perbandingan** dalam PHP:

```

1  <?php
2      echo "1. 12 < 14 = "; var_dump(12<14); // bool(true)
3      echo "<br />";
4      echo "2. 14 < 14 = "; var_dump(14<14); // bool(false)
5      echo "<br />";
6      echo "3. 14 <= 14 = "; var_dump(14<=14); // bool(true)
7      echo "<br />";
8      echo "4. 10 <> '10' = "; var_dump(10<>'10'); // bool(false)
9      echo "<br />";
10     echo "5. 10 == '10' = "; var_dump(10=='10'); // bool(true)
11     echo "<br />";
12     echo "6. 10 === '10' = "; var_dump(10==='10'); // bool(false)
13     echo "<br />";
14     echo "7. '150' == '1.5e2' = "; var_dump('150'=='1.5e2'); // bool(true)
15     echo "<br />";
16     echo "8. 'duniaikom' == 0 = "; var_dump('duniaikom'==0); // bool(true)
17     echo "<br />";
18     ?>

```

### 1.12 Operator Increment dan Decrement

**Operator Increment** dan **Decrement** adalah penyebutan untuk operasi seperti **\$a++**, dan **\$a--**. Jika anda telah mempelajari bahasa pemrograman lain, operasi **increment** dan **decrement** ini sering digunakan dalam *perulangan (looping)*.

**Increment** digunakan untuk *menambah variabel sebanyak 1 angka*, sedangkan **decrement** digunakan untuk *mengurangi variabel sebanyak 1 angka*. Penulisannya menggunakan tanda tambah 2 kali untuk **increment**, dan tanda kurang 2 kali untuk **decrement**. Penempatan tanda tambah atau kurang ini boleh diawal, atau diakhir variabel, namun keduanya memiliki perbedaan, sehingga terdapat **4 jenis increment dan decrement** dalam PHP.

## Jenis Operator Increment dan Decrement dalam PHP

Berikut adalah tabel 4 jenis operator **increment** dan **decrement** dalam PHP:

Contoh	Nama	Hasil
<code>++\$a</code>	Pre-increment	Tambah nilai \$a sebanyak 1, lalu kirim nilai \$a
<code>\$a++</code>	Post-increment	Kirim nilai \$a, lalu tambah nilai \$a sebanyak 1
<code>--\$a</code>	Pre-decrement	Kurangi nilai \$a sebanyak 1, lalu kirim nilai \$a
<code>\$a--</code>	Post-decrement	Kirim nilai \$a, lalu kurangi nilai \$a sebanyak 1

Dari tabel diatas terlihat bahwa terdapat 2 jenis **increment**, yaitu **Pre-increment**, dan **Post-Increment**, dan 2 jenis **decrement**, yaitu **Pre-decrement** dan **Post-decrement**. Perbedaan keduanya terletak pada posisi mana tanda tambah atau kurang diletakkan.

## Cara Penggunaan Operator Increment dan Decrement

Untuk memahami cara penggunaan operator **increment** dan **decrement**, berikut contoh kode program PHP:

```
1  <?php
2      echo "<h3>Postincrement</h3>";
3      $a = 5;
4      echo "\$a = $a <br />";
5      echo "\$a akan bernilai 5: " . $a++ . " (\$a++)<br />";
6      echo "\$a akan bernilai 6: " . $a . "<br />";
7      .....
8      echo "<h3>Preincrement</h3>";
9      $a = 5;
10     echo "\$a = $a <br />";
11     echo "\$a akan bernilai 6: " . ++$a . " (++\$a)<br />";
12     echo "\$a akan bernilai 6: " . $a . "<br />";
13     .....
14     echo "<h3>Postdecrement</h3>";
15     $a = 5;
16     echo "\$a = $a <br />";
17     echo "\$a akan bernilai 5: " . $a-- . " (\$a--)<br />";
18     echo "\$a akan bernilai 4: " . $a . "<br />";
19     .....
20     echo "<h3>Predecrement</h3>";
21     $a = 5;
22     echo "\$a = $a <br />";
23     echo "\$a akan bernilai 4: " . --$a . " (--\$a)<br />";
24     echo "\$a akan bernilai 4: " . $a . "<br />";
25  ?>
```

### Postincrement

```
$a = 5  
$a akan bernilai 5: 5 ($a++)  
$a akan bernilai 6: 6
```

### Preincrement

```
$a = 5  
$a akan bernilai 6: 6 (++$a)  
$a akan bernilai 6: 6
```

### Postdecrement

```
$a = 5  
$a akan bernilai 5: 5 ($a--)  
$a akan bernilai 4: 4
```

### Predecrement

```
$a = 5  
$a akan bernilai 4: 4 (--$a)  
$a akan bernilai 4: 4
```

Terlihat bahwa **Post-increment** (\$a++), akan memberikan hasilnya dulu, baru menambahkan nilai variabel \$a sebanyak 1 angka, namun dengan **Pre-increment**, \$a akan ditambahkan 1 angka, baru nilainya ditampilkan. Begitu juga hal nya dengan operasi **Post-decrement** dan **Pre-decrement**.