Internet Engineering
PHP
مهندسی اینترنت

Dr. Yaeghoobi

Ph.D. (Computer Science and Engineering), India

وزارت علوم تحقیقات و فناوری

دانشگاه علم و فرهنگ

Contents

Introduction

Introduction

- **PHP Hypertext Preprocessor (PHP)**
- The **PHP** is a programming language that allows web developers **to create dynamic** content that **interacts with databases**.
- PHP is basically used for developing web based software applications.
- PHP is **a server side scripting** language that is embedded in HTML.
- It is used to **manage dynamic** content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

Common uses of PHP

- PHP performs **system functions**, i.e. **from files on a system it can create, open, read, write, and close them.**
- PHP can **handle forms**, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You **add, delete, modify** elements within your database through PHP.
- **Access cookies** variables and **set cookies**.
- Using PHP, you can **restrict users** to access some pages of your website.
- It can **encrypt data**.

Characteristics of PHP

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

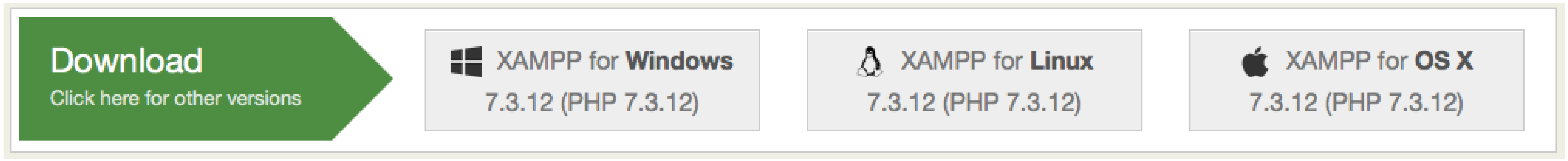
Develop and Run PHP

- **Web Server** – PHP will work with virtually **all Web Server software**
- **Database** – PHP will work with virtually **all database software**, including Oracle and Sybase but most commonly used is freely available MySQL database.
- **PHP Parser** – In order to process PHP script instructions **a parser must be installed** to generate HTML output that can be sent to the Web Browser.

Local Server

Install Local Server

- Eg. **XAMPP**
- XAMPP is a **free** and **open-source cross-platform** web server solution stack package developed by Apache Friend



Download
Click here for other versions

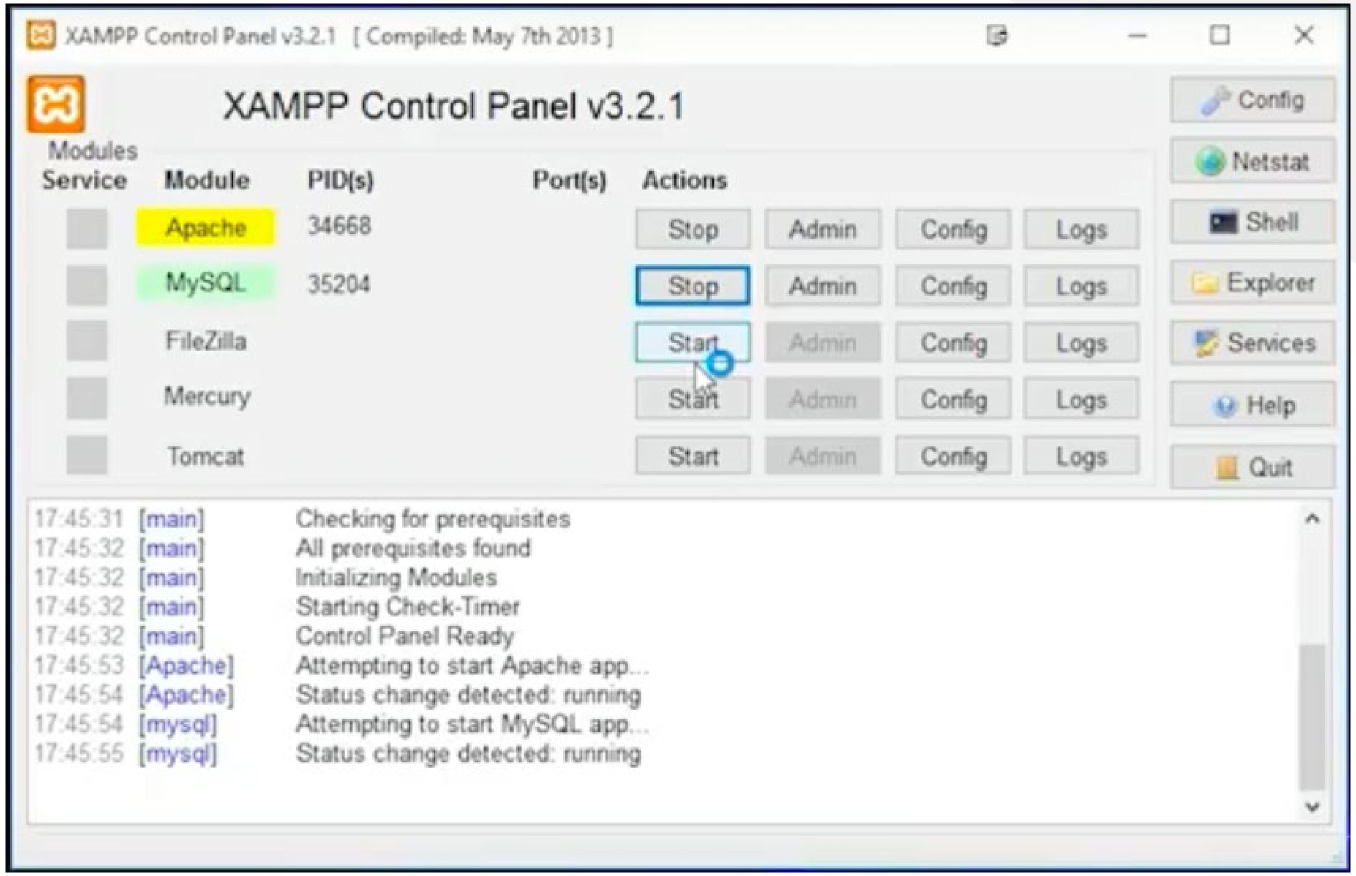
XAMPP for **Windows**
7.3.12 (PHP 7.3.12)

XAMPP for **Linux**
7.3.12 (PHP 7.3.12)

XAMPP for **OS X**
7.3.12 (PHP 7.3.12)

Activate Local Server

- Install
- go to xampp folder -> Xampp-control.exe -> run
- Active Apache and MySQL

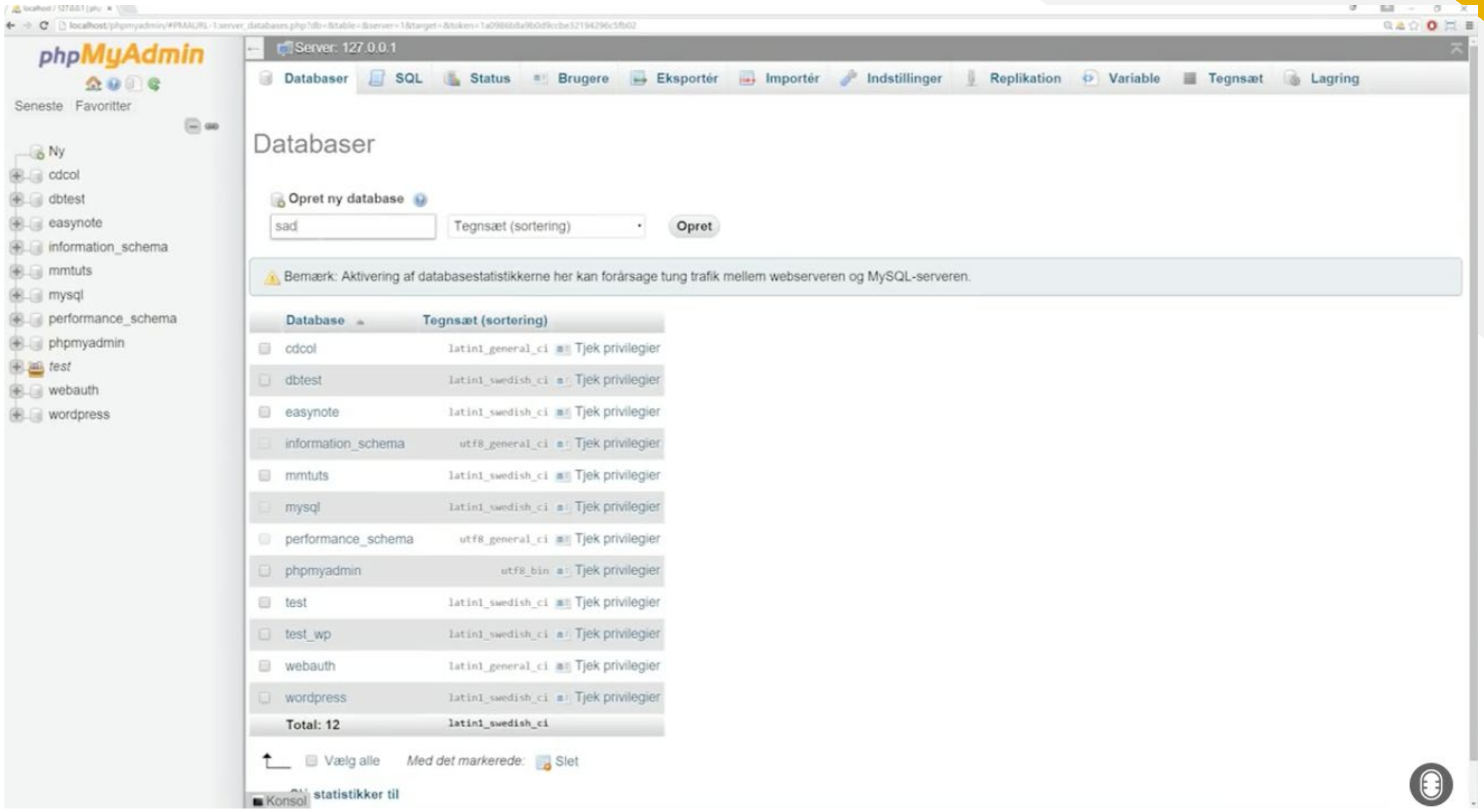

XAMPP Control Panel v3.2.1 [Compiled: May 7th 2013]
XAMPP Control Panel v3.2.1
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
	Apache	34668		Stop	Admin	Config	Logs
	MySQL	35204		Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
17:45:31  [main]     Checking for prerequisites
17:45:32  [main]     All prerequisites found
17:45:32  [main]     Initializing Modules
17:45:32  [main]     Starting Check-Timer
17:45:32  [main]     Control Panel Ready
17:45:53  [Apache]   Attempting to start Apache app...
17:45:54  [Apache]   Status change detected: running
17:45:54  [mysql]    Attempting to start MySQL app...
17:45:55  [mysql]    Status change detected: running
```

Create a DataBase

- On Browser:
- Localhost/phpmyadmin



Server: 127.0.0.1

Databaser | SQL | Status | Brugere | Eksportér | Importér | Indstillinger | Replikation | Variable | Tegnsæt | Lagring

Databaser

Opret ny database

sad | Tegnsæt (sortering) | Opret

Bemærk: Aktivering af databasestatistikkerne her kan forårsage tung trafik mellem webserveren og MySQL-serveren.

Database	Tegnsæt (sortering)	
cdcol	latin1_general_ci	Tjek privilegier
dbtest	latin1_swedish_ci	Tjek privilegier
easynote	latin1_swedish_ci	Tjek privilegier
information_schema	utf8_general_ci	Tjek privilegier
mmtuts	latin1_swedish_ci	Tjek privilegier
mysql	latin1_swedish_ci	Tjek privilegier
performance_schema	utf8_general_ci	Tjek privilegier
phpmyadmin	utf8_bin	Tjek privilegier
test	latin1_swedish_ci	Tjek privilegier
test_wp	latin1_swedish_ci	Tjek privilegier
webauth	latin1_general_ci	Tjek privilegier
wordpress	latin1_swedish_ci	Tjek privilegier
Total: 12	latin1_swedish_ci	

Vælg alle *Med det markerede:* Slet

statistikker til

Run a Code

- go to xampp folder -> htdocs folder -> delete all the random default files or folder -> create new folder (root folder- all the php files will save in this folder)

- Go to Notepad
- Write a php code
- Save as (example).php in root folder (inside htdocs)
- On browser: localhost/name of root folder

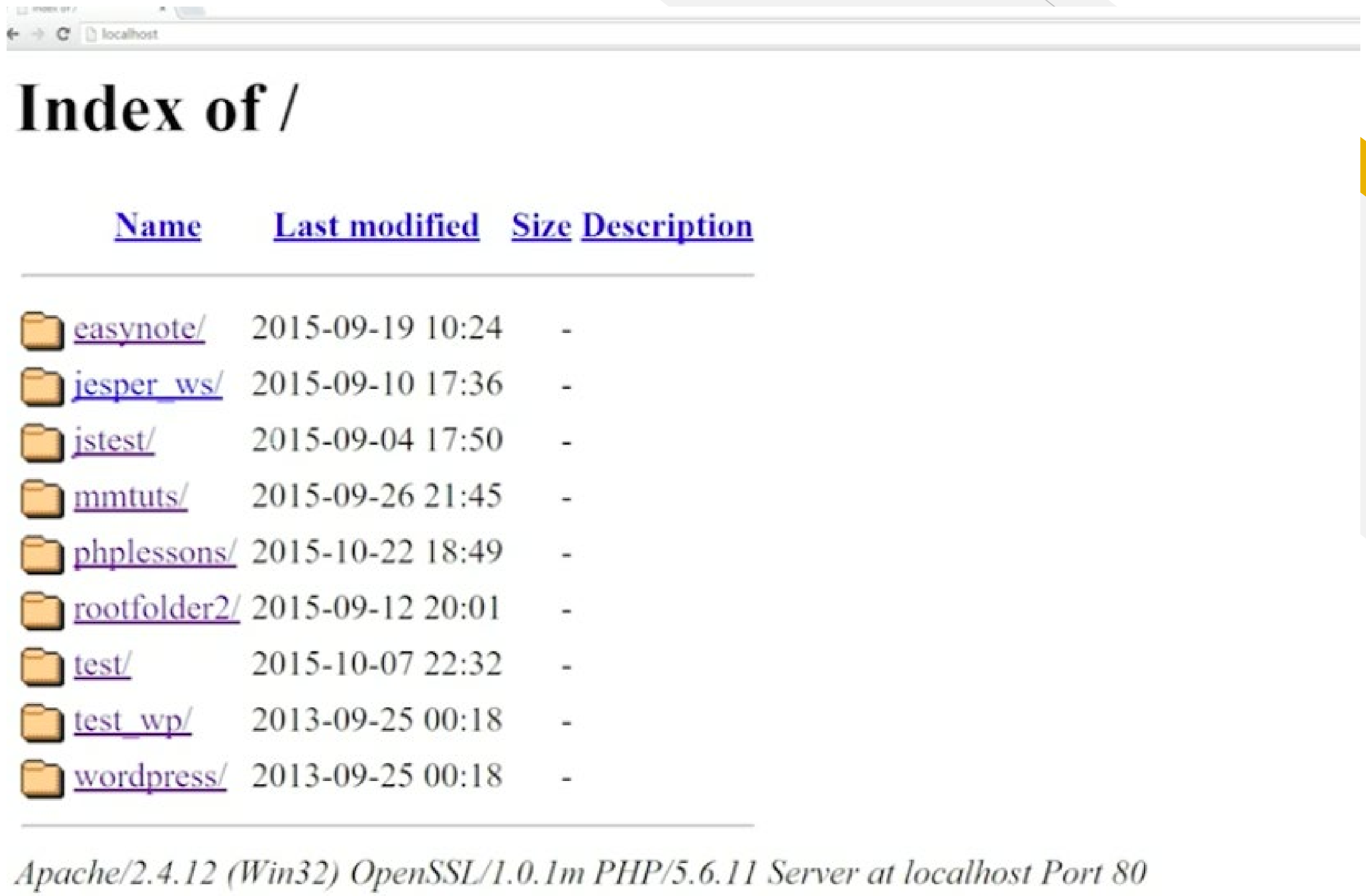
Write PHP Code

localhost

- Run local server
- On browser:
- Localhost

(shows folders created in hddocs)

- Create a file as .html or .php



localhost

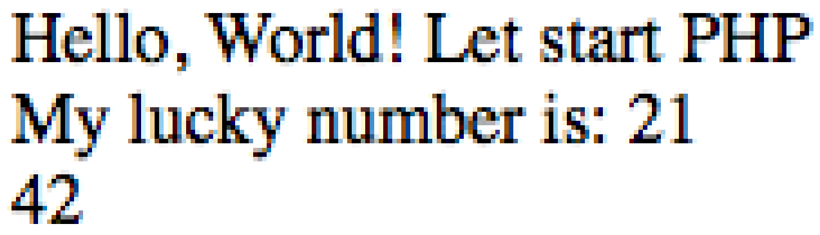
Index of /

Name	Last modified	Size	Description
easynote/	2015-09-19 10:24	-	
jesper_ws/	2015-09-10 17:36	-	
jstest/	2015-09-04 17:50	-	
mmtuts/	2015-09-26 21:45	-	
phplessons/	2015-10-22 18:49	-	
rootfolder2/	2015-09-12 20:01	-	
test/	2015-10-07 22:32	-	
test_wp/	2013-09-25 00:18	-	
wordpress/	2013-09-25 00:18	-	

Apache/2.4.12 (Win32) OpenSSL/1.0.1m PHP/5.6.11 Server at localhost Port 80

Hello World

```
<html>
<head>
        <title>Hello World</title>
</head>
<body>
  <?php
        echo "Hello, World! ";
        print ”Let start PHP ”;
        print "Let start PHP <br>";
        echo "My lucky number is: 21<p>";
        echo 12+30;
  ?>
</body>
</html>
```



```
Hello, World! Let start PHP
My lucky number is: 21
42
```

https://www.tutorialspoint.com/php_webview_online.php

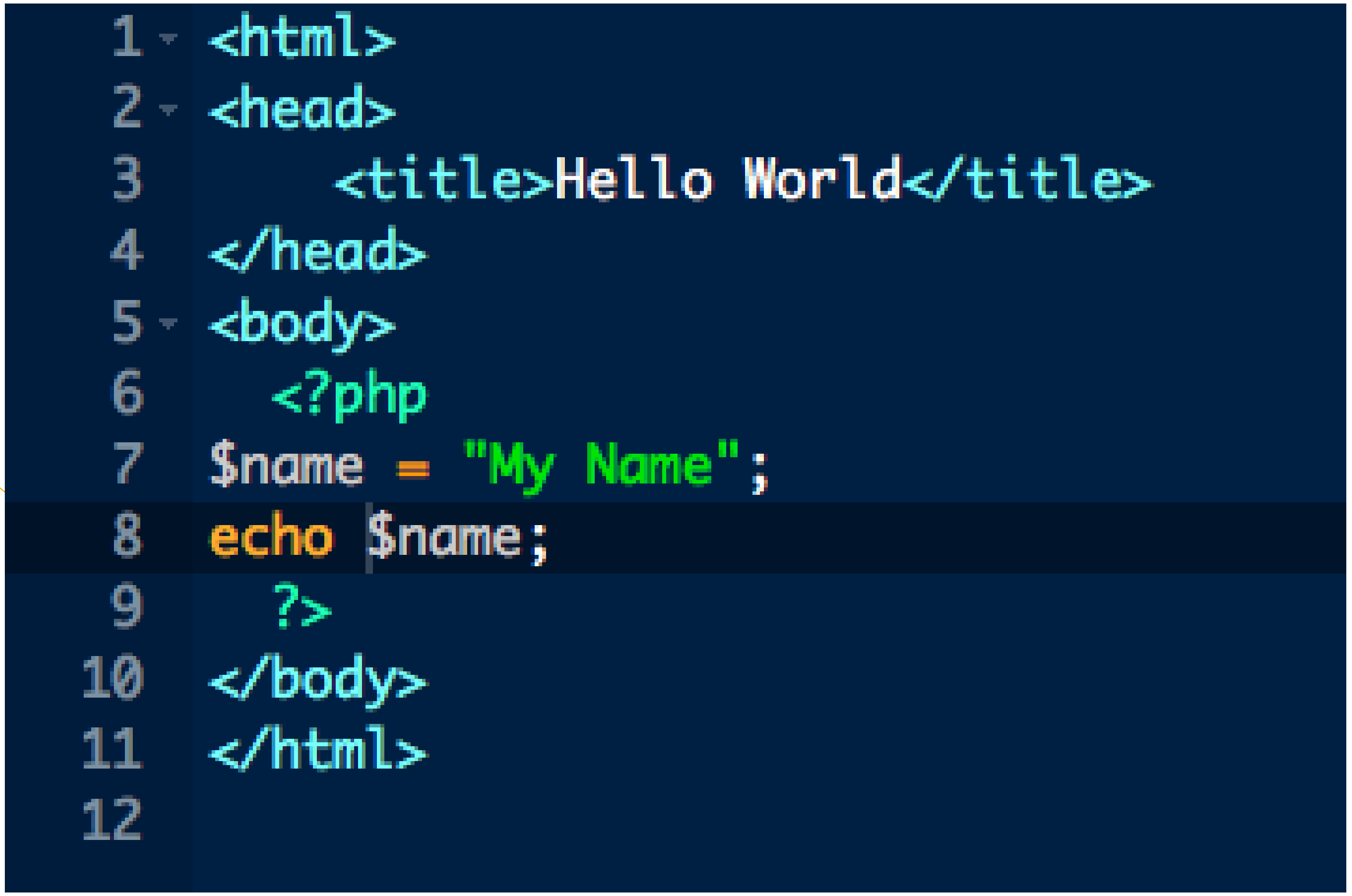
PHP Variables

PHP Variables

- All variables in PHP are denoted with a leading dollar sign ($).
- **Integers** − are whole numbers, without a decimal point, like 4195.
- **Doubles** − are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** − have only two possible values either true or false.
- **NULL** − is a special type that only has one value: NULL.
- **Strings** − are sequences of characters, like 'PHP supports string operations.'
- **Arrays** − are named and indexed collections of other values.
- **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

Variable Naming

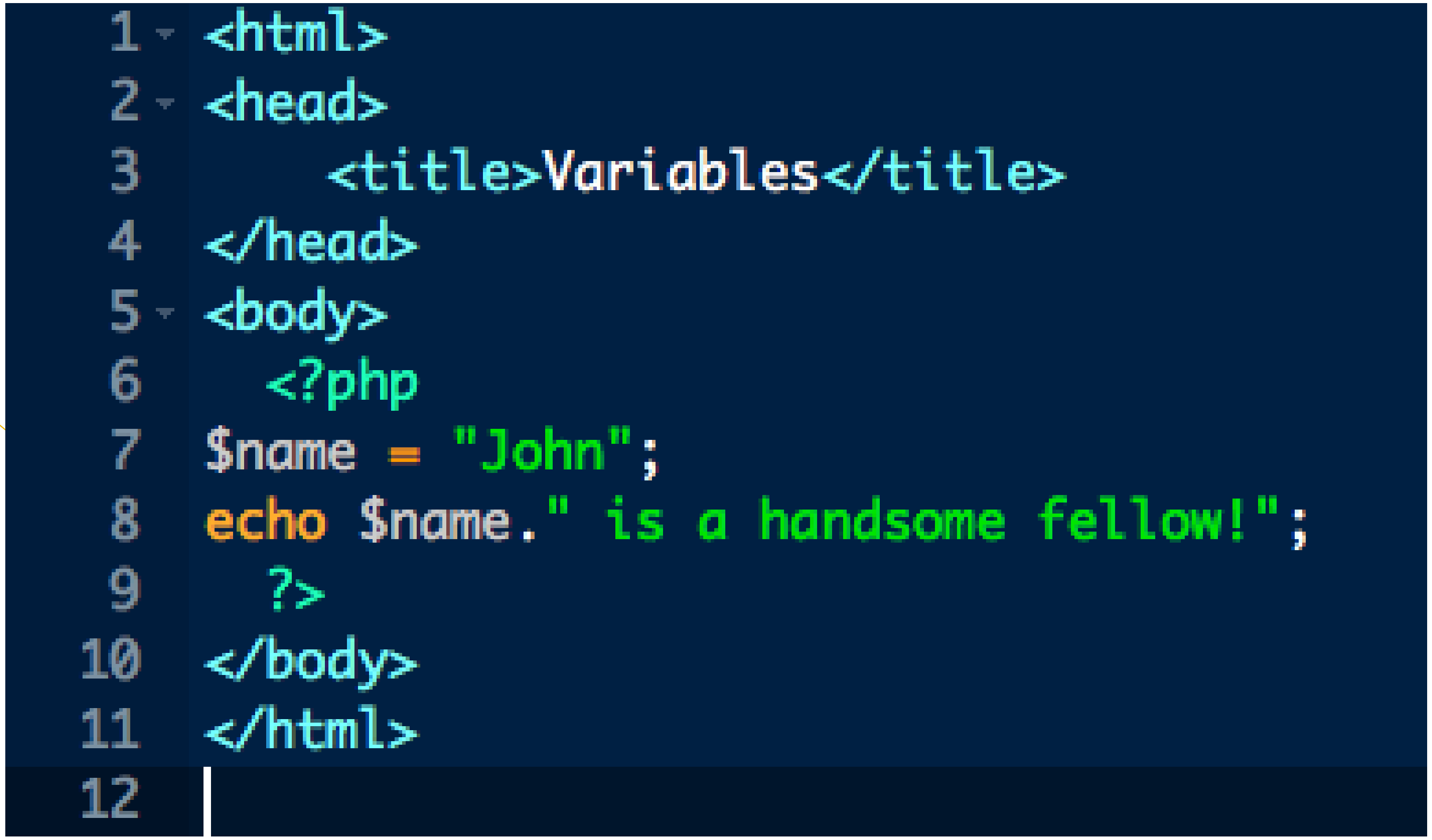
- Variable names must begin with a **letter or underscore** character.
- A variable name can consist of numbers, letters, underscores but you **cannot** use characters like + , - , % , (,) . & , etc
- There is **no size limit** for variables.



```
<html>
<head>
     <title>Hello World</title>
</head>
<body>
  <?php
$name = "My Name";
echo $name;
  ?>
</body>
</html>

```

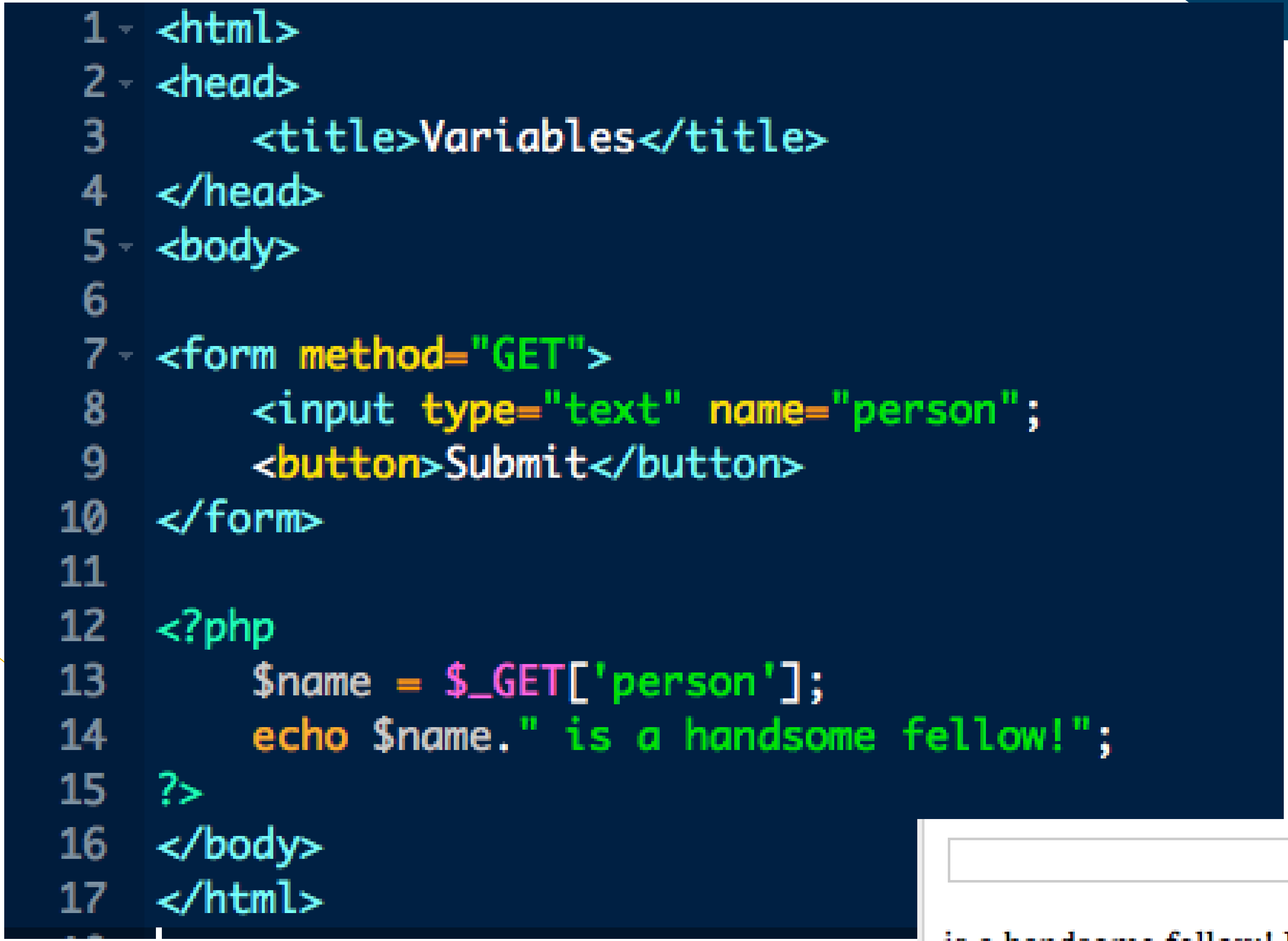
My Name

Combine PHP code and Strings

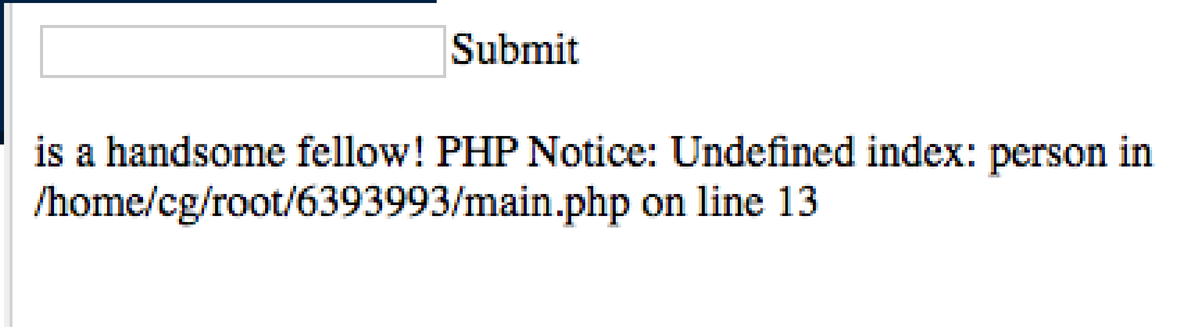


```
<html>
<head>
     <title>Variables</title>
</head>
<body>
  <?php
$name = "John";
echo $name." is a handsome fellow!";
  ?>
</body>
</html>
```

```
John is a handsome fellow!
```

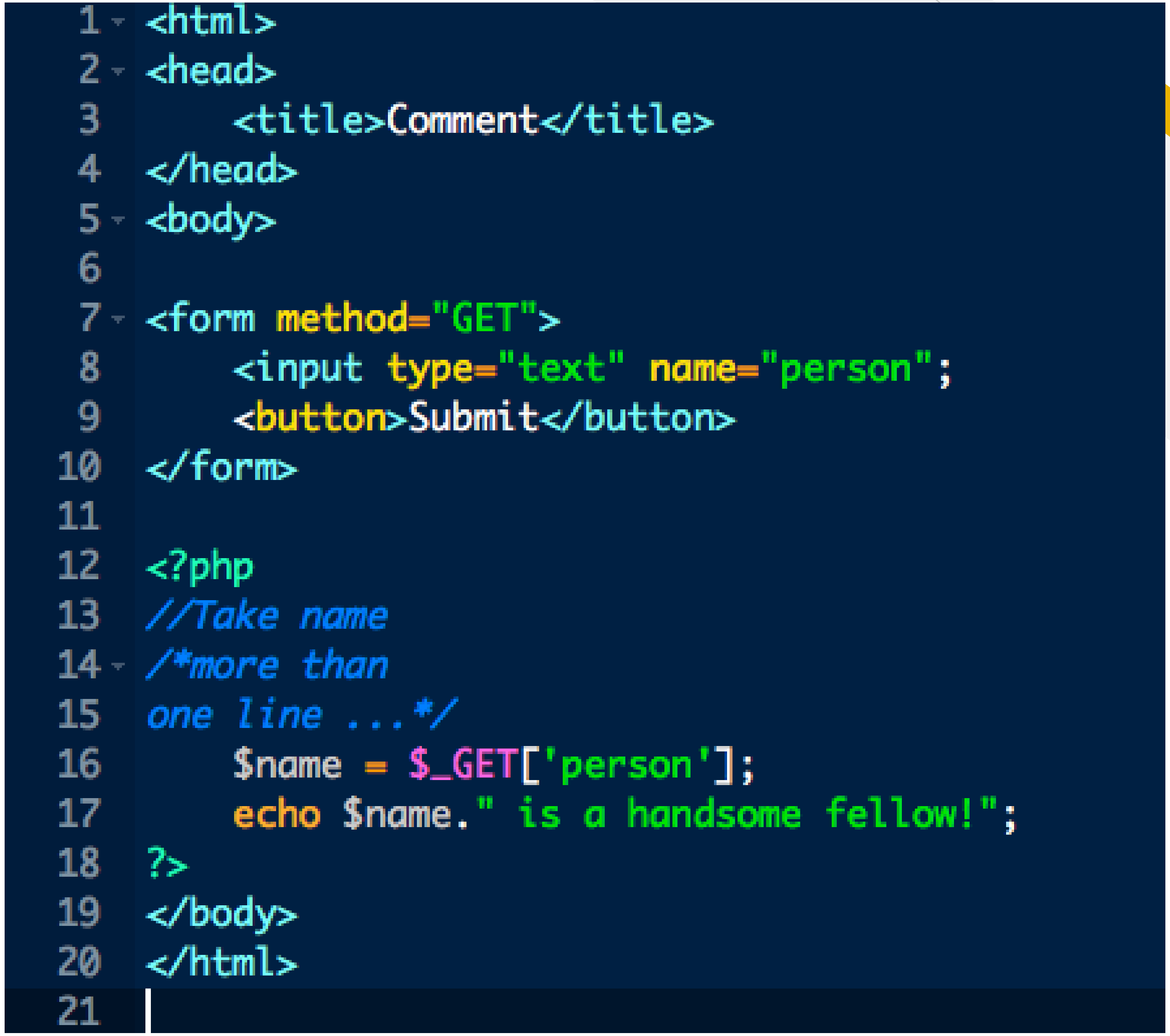


```
<html>
<head>
    <title>Variables</title>
</head>
<body>

<form method="GET">
    <input type="text" name="person";
    <button>Submit</button>
</form>

<?php
    $name = $_GET['person'];
    echo $name." is a handsome fellow!";
?>
</body>
</html>
```



```
is a handsome fellow! PHP Notice: Undefined index: person in
/home/cg/root/6393993/main.php on line 13
```

Comments in PHP

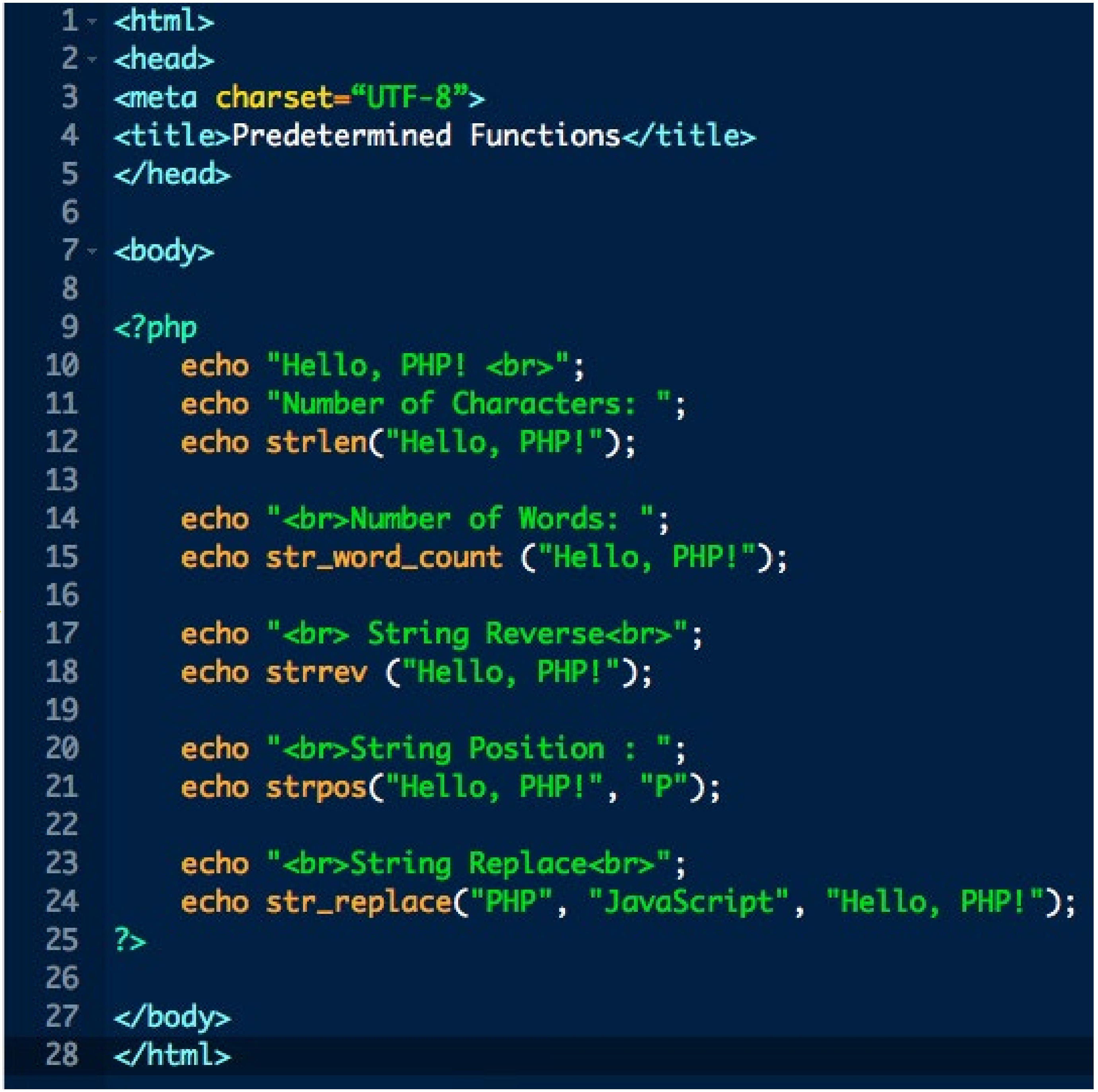


```
<html>
<head>
    <title>Comment</title>
</head>
<body>

<form method="GET">
    <input type="text" name="person";
    <button>Submit</button>
</form>

<?php
//Take name
/*more than
one line ...*/
    $name = $_GET['person'];
    echo $name." is a handsome fellow!";
?>
</body>
</html>

```

Functions

Predetermined Functions

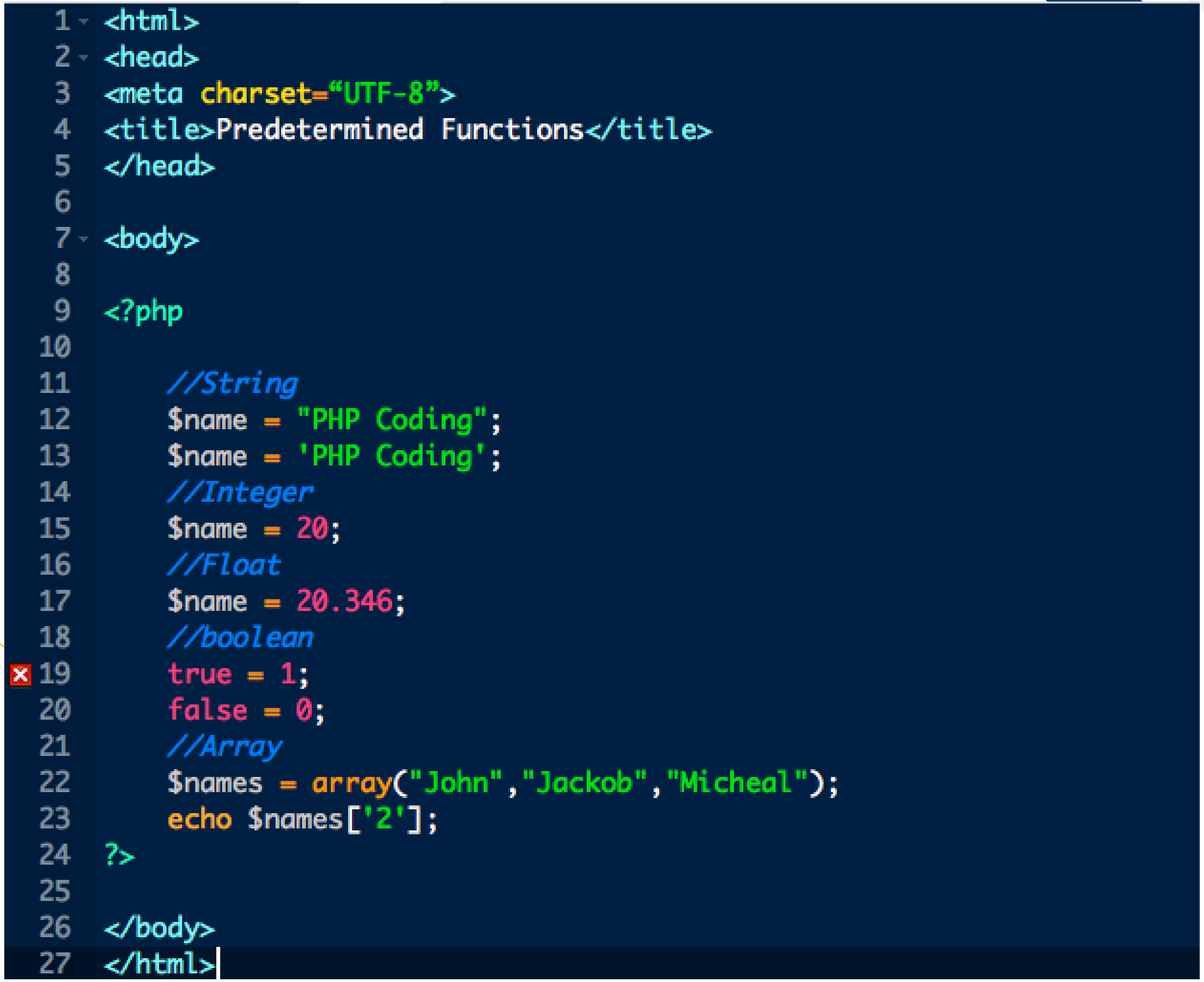
- strlen(”String");
- str_word_count (”Sentences");
- strrev (”String");
- strpos(”String", ”letter/word to find");
- str_replace(”word to be replace", ”replace by", ”Sentences");



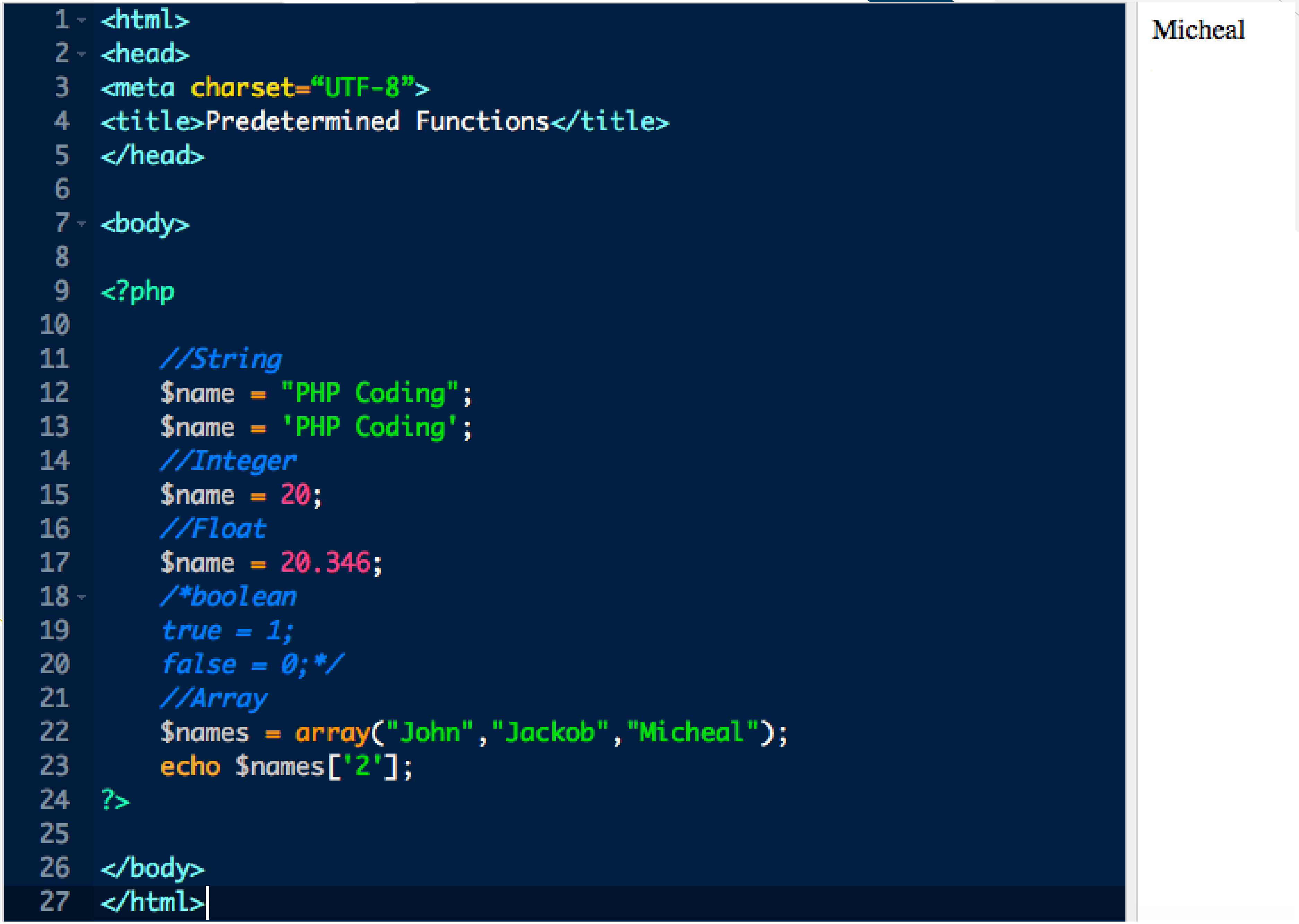
```
<html>
<head>
<meta charset="UTF-8">
<title>Predetermined Functions</title>
</head>

<body>

<?php
    echo "Hello, PHP! <br>";
    echo "Number of Characters: ";
    echo strlen("Hello, PHP!");

    echo "<br>Number of Words: ";
    echo str_word_count ("Hello, PHP!");

    echo "<br> String Reverse<br>";
    echo strrev ("Hello, PHP!");

    echo "<br>String Position : ";
    echo strpos("Hello, PHP!", "P");

    echo "<br>String Replace<br>";
    echo str_replace("PHP", "JavaScript", "Hello, PHP!");
?>

</body>
</html>
```

```
Hello, PHP!
Number of Characters: 11
Number of Words: 2
String Reverse
!PHP ,olleH
String Position : 7
String Replace
Hello, JavaScript!
```

Data Types



```
<html>
<head>
<meta charset="UTF-8">
<title>Predetermined Functions</title>
</head>

<body>

<?php

    //String
    $name = "PHP Coding";
    $name = 'PHP Coding';
    //Integer
    $name = 20;
    //Float
    $name = 20.346;
    //boolean
    true = 1;
    false = 0;
    //Array
    $names = array("John","Jackob","Micheal");
    echo $names['2'];
?>

</body>
</html>
```



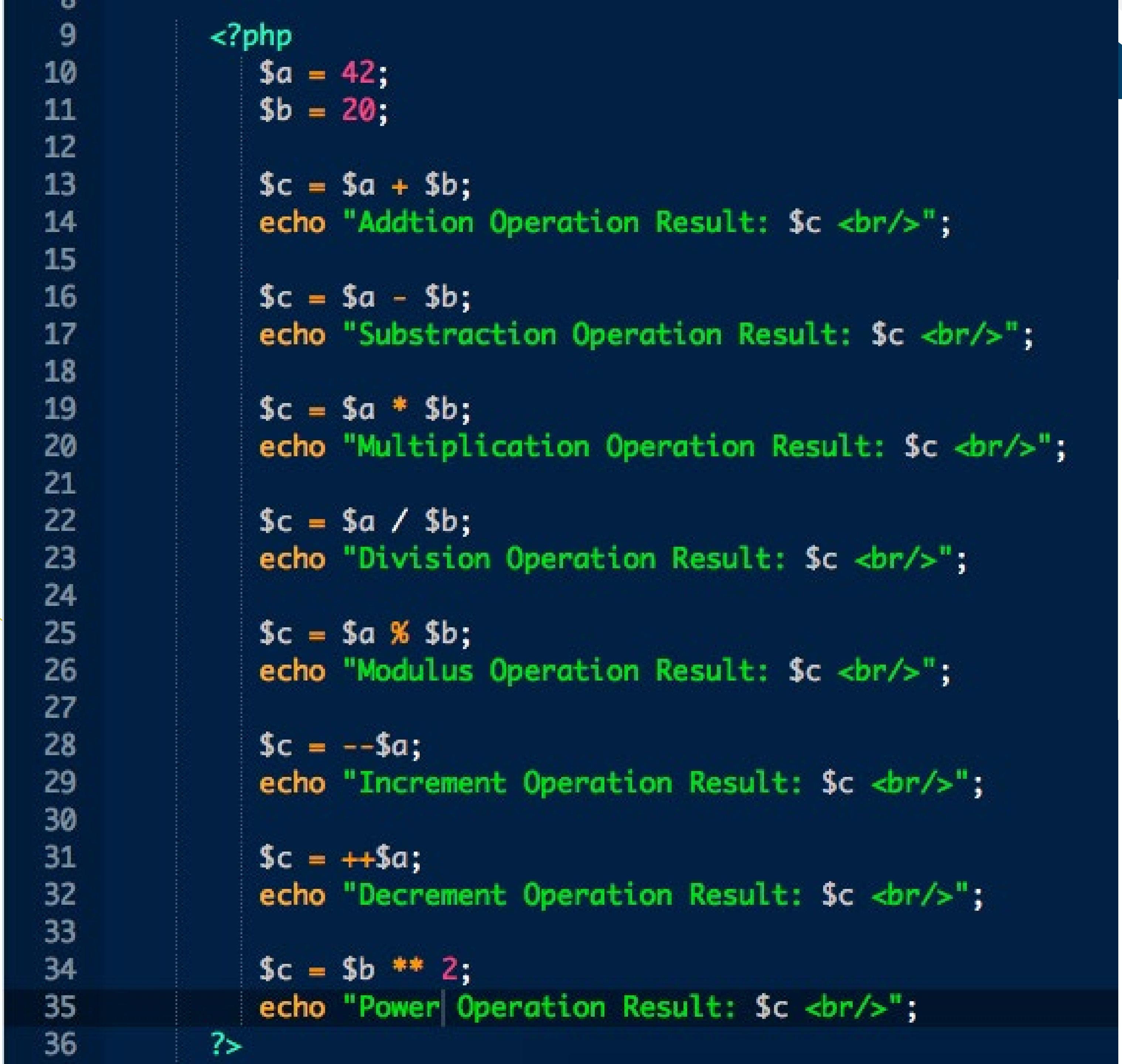
```
<html>
<head>
<meta charset="UTF-8">
<title>Predetermined Functions</title>
</head>

<body>

<?php

    //String
    $name = "PHP Coding";
    $name = 'PHP Coding';
    //Integer
    $name = 20;
    //Float
    $name = 20.346;
    /*boolean
    true = 1;
    false = 0;*/
    //Array
    $names = array("John","Jackob","Micheal");
    echo $names['2'];
?>

</body>
</html>
```

```
Micheal
```

Operators

Arithmetic Operators

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

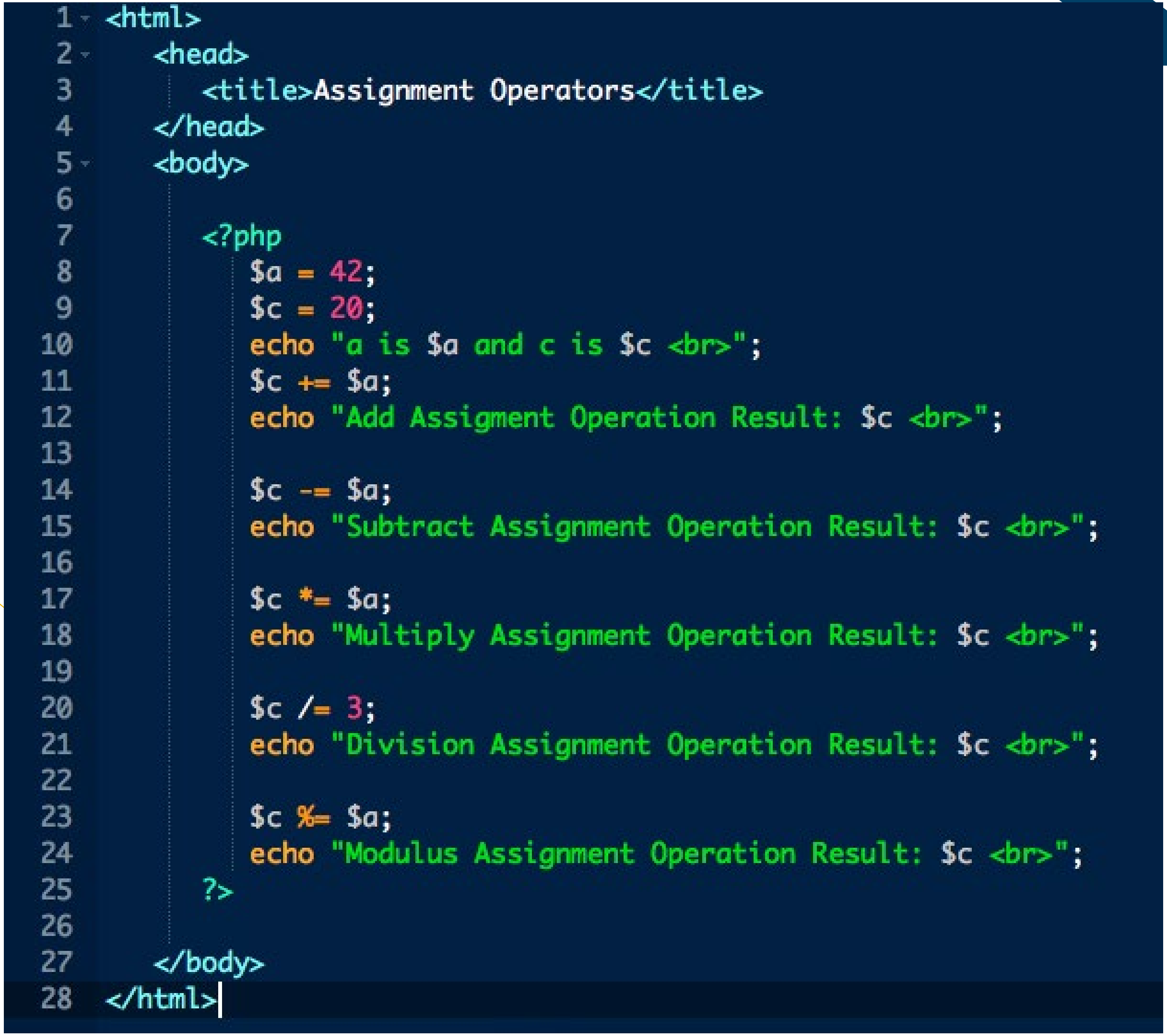


```php
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b;
    echo "Addtion Operation Result: $c <br/>";

    $c = $a - $b;
    echo "Substraction Operation Result: $c <br/>";

    $c = $a * $b;
    echo "Multiplication Operation Result: $c <br/>";

    $c = $a / $b;
    echo "Division Operation Result: $c <br/>";

    $c = $a % $b;
    echo "Modulus Operation Result: $c <br/>";

    $c = --$a;
    echo "Increment Operation Result: $c <br/>";

    $c = ++$a;
    echo "Decrement Operation Result: $c <br/>";

    $c = $b ** 2;
    echo "Power Operation Result: $c <br/>";
?>
```

```
Addtion Operation Result: 62
Substraction Operation Result: 22
Multiplication Operation Result: 840
Division Operation Result: 2.1
Modulus Operation Result: 2
Increment Operation Result: 41
Decrement Operation Result: 42
Power Operation Result: 400
```

Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

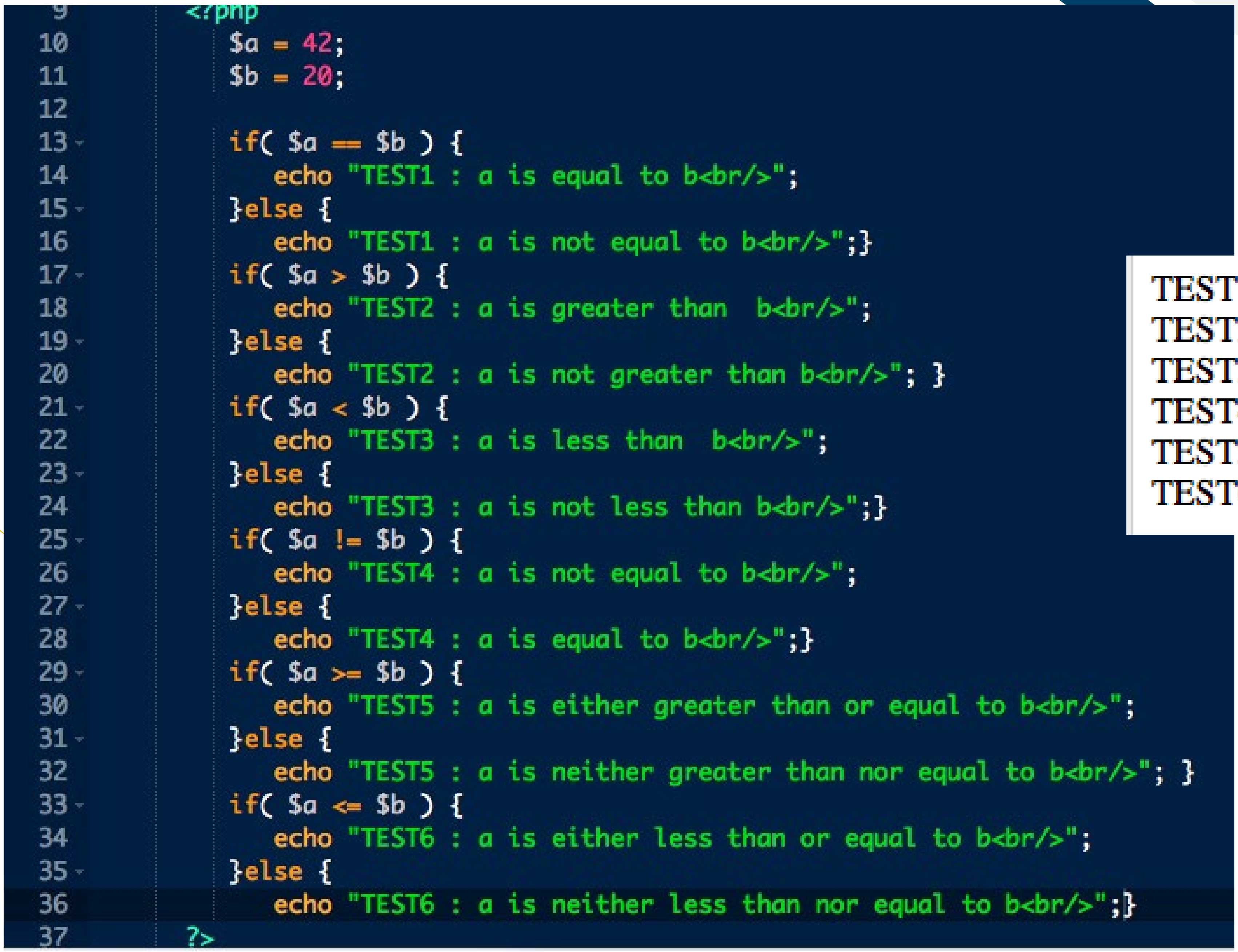


```
<html>
    <head>
        <title>Assignment Operators</title>
    </head>
    <body>

        <?php
            $a = 42;
            $c = 20;
            echo "a is $a and c is $c <br>";
            $c += $a;
            echo "Add Assigment Operation Result: $c <br>";

            $c -= $a;
            echo "Subtract Assignment Operation Result: $c <br>";

            $c *= $a;
            echo "Multiply Assignment Operation Result: $c <br>";

            $c /= 3;
            echo "Division Assignment Operation Result: $c <br>";

            $c %= $a;
            echo "Modulus Assignment Operation Result: $c <br>";
        ?>

    </body>
</html>
```

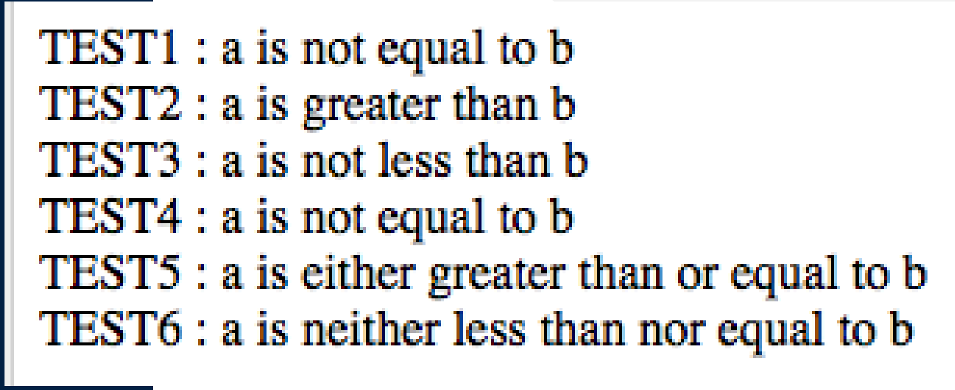
```
a is 42 and c is 20
Add Assigment Operation Result: 62
Subtract Assignment Operation Result: 20
Multiply Assignment Operation Result: 840
Division Assignment Operation Result: 280
Modulus Assignment Operation Result: 28
```

Comparison Operators

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

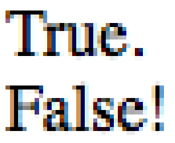


```php
<?php
    $a = 42;
    $b = 20;

    if( $a == $b ) {
       echo "TEST1 : a is equal to b<br/>";
    }else {
       echo "TEST1 : a is not equal to b<br/>";}
    if( $a > $b ) {
       echo "TEST2 : a is greater than  b<br/>";
    }else {
       echo "TEST2 : a is not greater than b<br/>"; }
    if( $a < $b ) {
       echo "TEST3 : a is less than  b<br/>";
    }else {
       echo "TEST3 : a is not less than b<br/>";}
    if( $a != $b ) {
       echo "TEST4 : a is not equal to b<br/>";
    }else {
       echo "TEST4 : a is equal to b<br/>";}
    if( $a >= $b ) {
       echo "TEST5 : a is either greater than or equal to b<br/>";
    }else {
       echo "TEST5 : a is neither greater than nor equal to b<br/>"; }
    if( $a <= $b ) {
       echo "TEST6 : a is either less than or equal to b<br/>";
    }else {
       echo "TEST6 : a is neither less than nor equal to b<br/>";}
?>
```



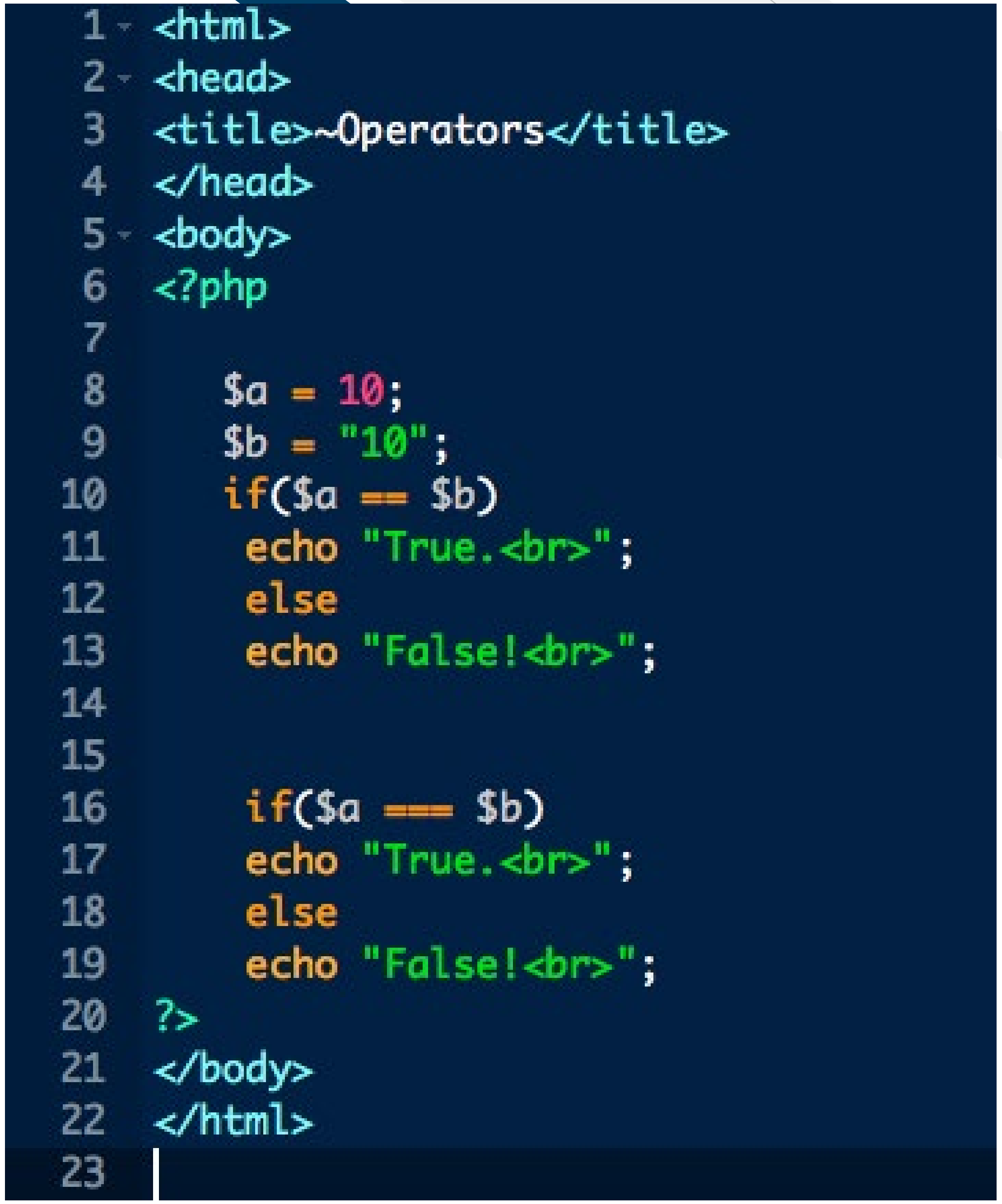
```
TEST1 : a is not equal to b
TEST2 : a is greater than b
TEST3 : a is not less than b
TEST4 : a is not equal to b
TEST5 : a is either greater than or equal to b
TEST6 : a is neither less than nor equal to b
```

What is the Answer?



```
True.
False!
```

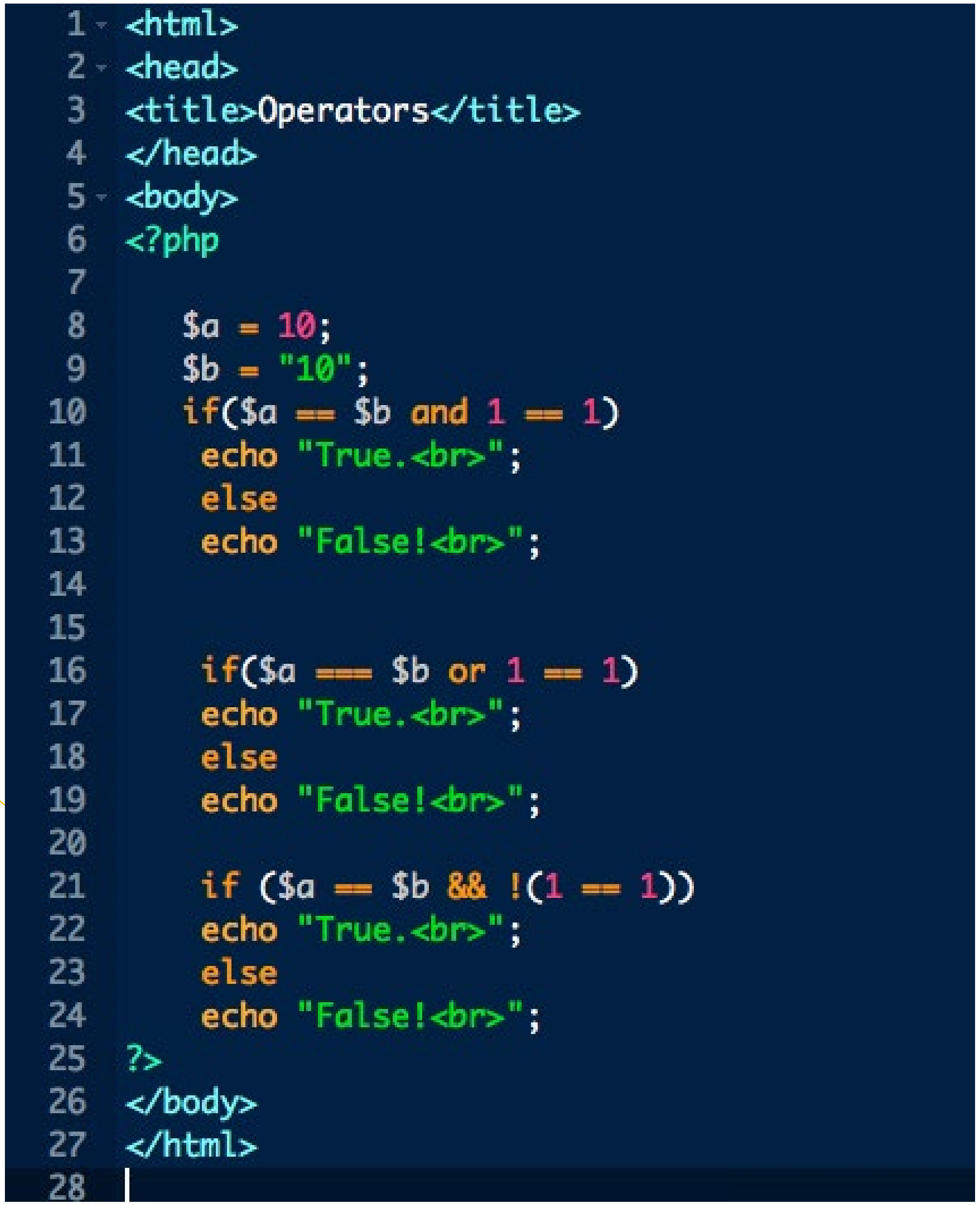
= = = means a is equal to b and in same datatype.



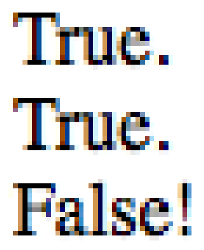
```
<html>
<head>
<title>~Operators</title>
</head>
<body>
<?php

    $a = 10;
    $b = "10";
    if($a == $b)
     echo "True.<br>";
     else
     echo "False!<br>";


     if($a === $b)
     echo "True.<br>";
     else
     echo "False!<br>";
?>
</body>
</html>

```

Logical Operators

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

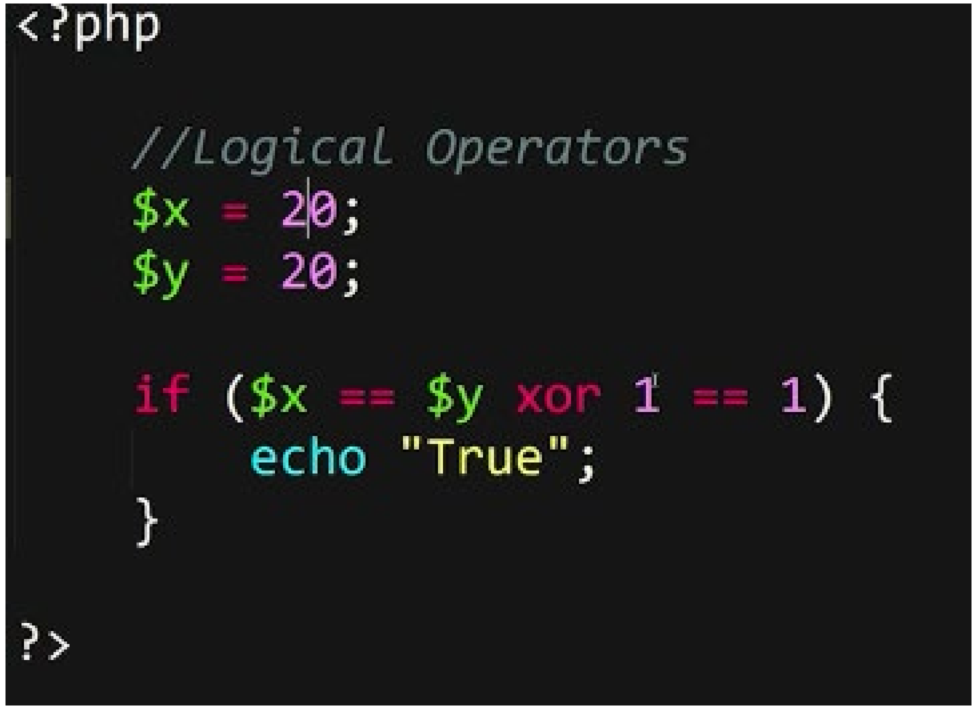


```php
<html>
<head>
<title>Operators</title>
</head>
<body>
<?php

   $a = 10;
   $b = "10";
   if($a == $b and 1 == 1)
    echo "True.<br>";
    else
    echo "False!<br>";


    if($a === $b or 1 == 1)
    echo "True.<br>";
    else
    echo "False!<br>";

    if ($a == $b && !(1 == 1))
    echo "True.<br>";
    else
    echo "False!<br>";
?>
</body>
</html>

```



```
True.
True.
False!
```

What is the Answer?

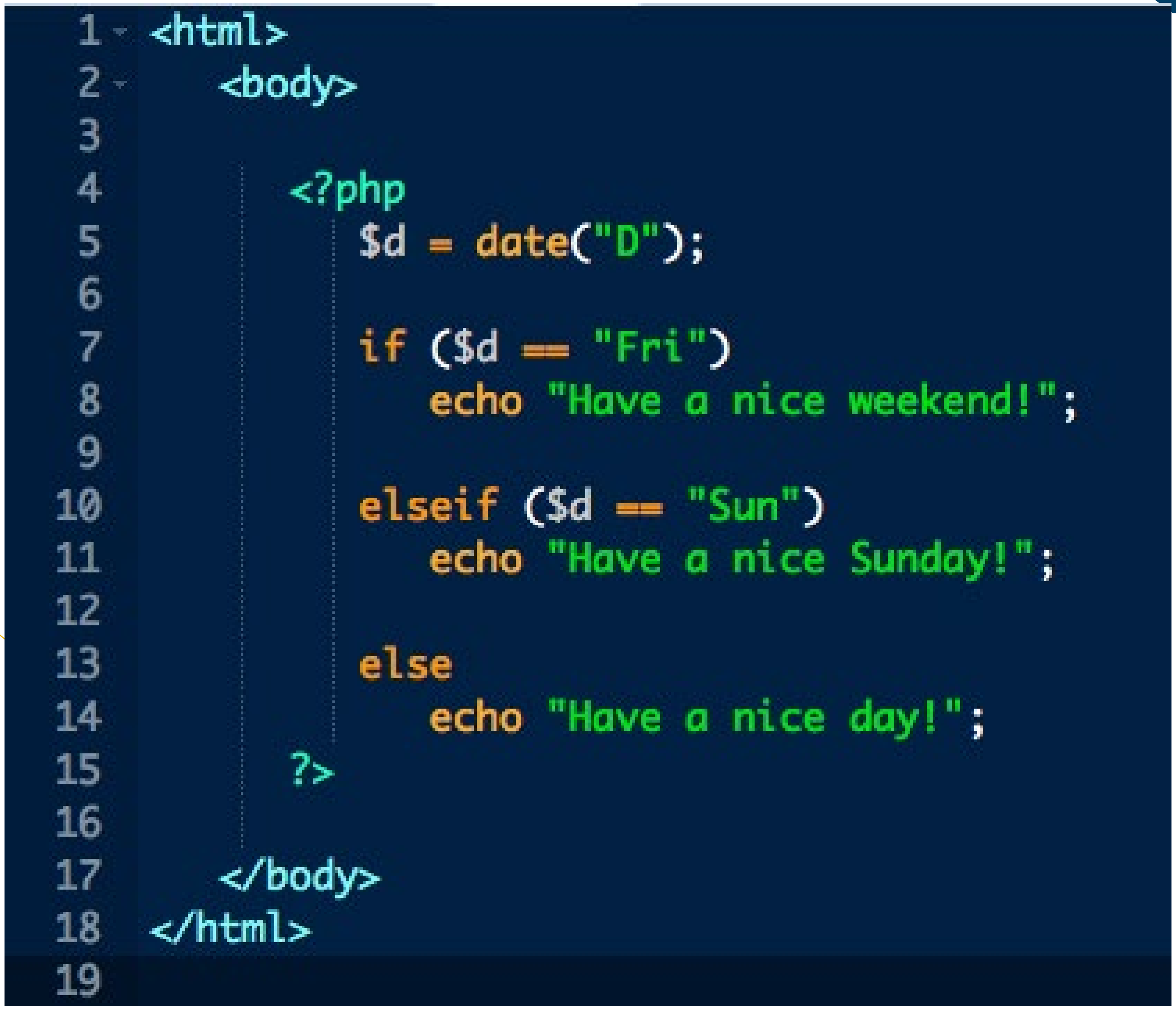


```
<?php

    //Logical Operators
    $x = 20;
    $y = 20;

    if ($x == $y xor 1 == 1) {
        echo "True";
    }

?>
```

Conditional Statement

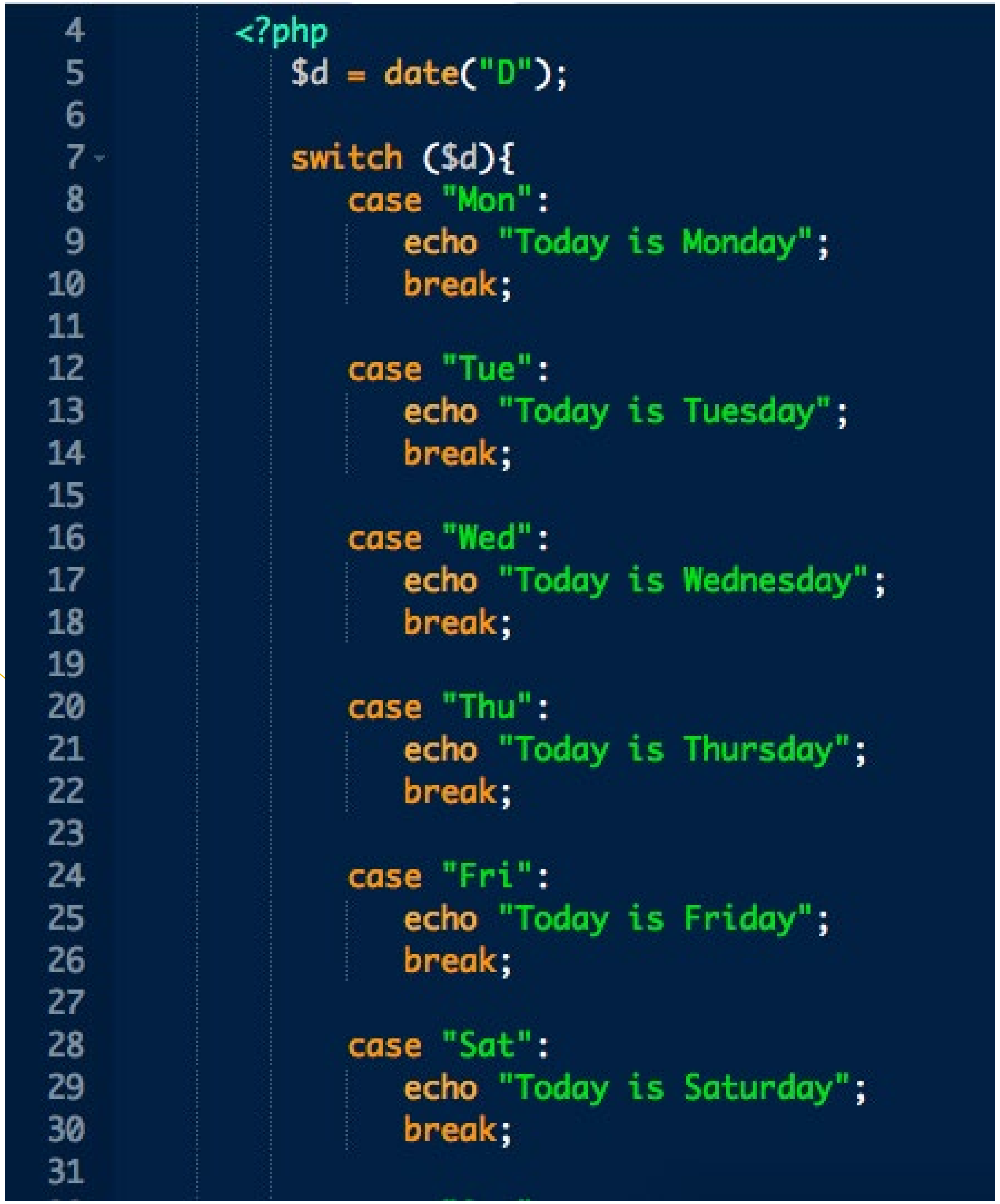
Decision Making

- **if...else statement** – use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** – is used with the if...else statement to execute a set of code if **one** of the several condition is true
- **switch statement** – is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

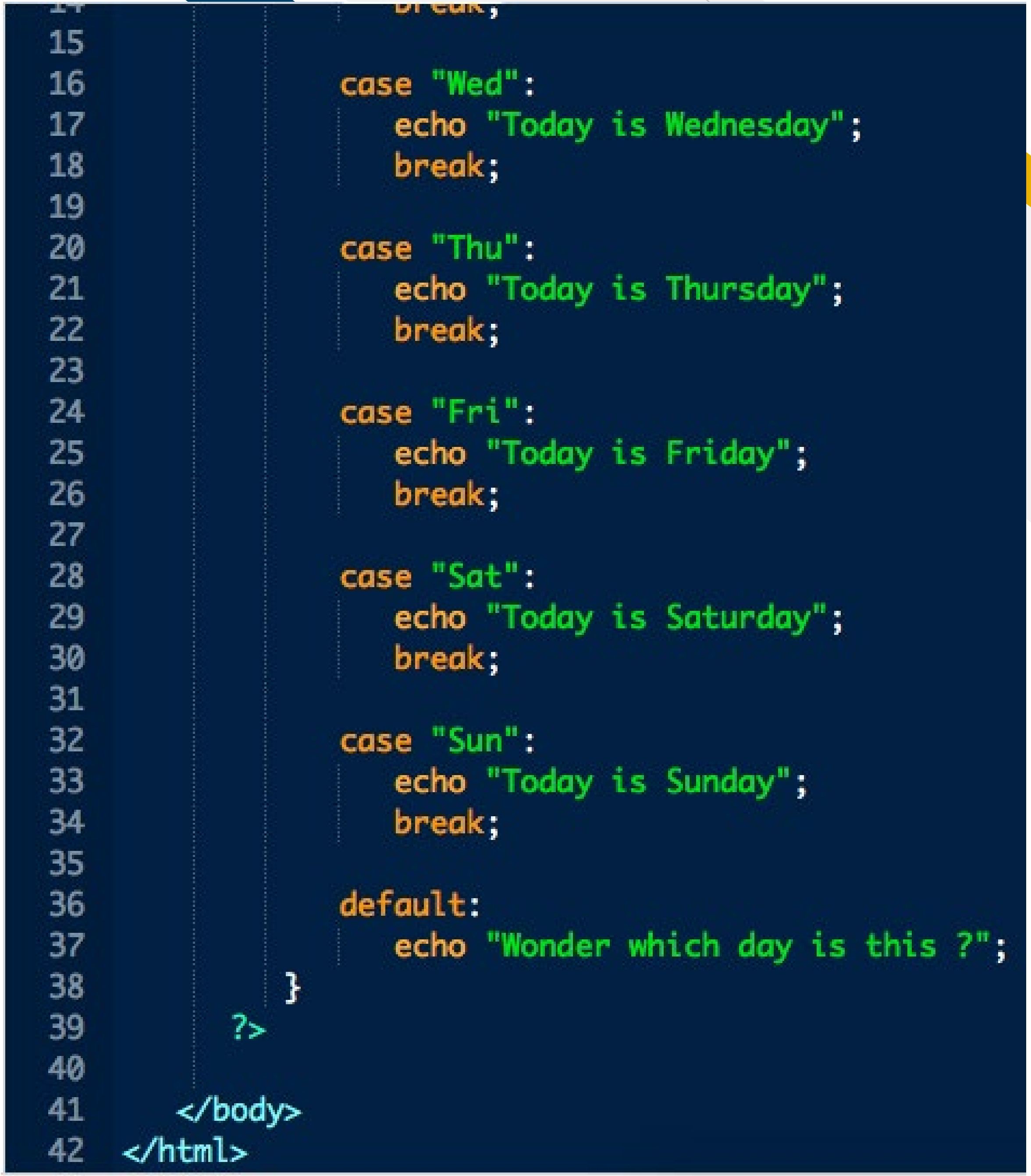


```
<html>
    <body>

        <?php
            $d = date("D");

            if ($d == "Fri")
                echo "Have a nice weekend!";

            elseif ($d == "Sun")
                echo "Have a nice Sunday!";

            else
                echo "Have a nice day!";
        ?>

    </body>
</html>

```

Have a nice weekend!



```php
<?php
    $d = date("D");

    switch ($d){
        case "Mon":
            echo "Today is Monday";
            break;

        case "Tue":
            echo "Today is Tuesday";
            break;

        case "Wed":
            echo "Today is Wednesday";
            break;

        case "Thu":
            echo "Today is Thursday";
            break;

        case "Fri":
            echo "Today is Friday";
            break;

        case "Sat":
            echo "Today is Saturday";
            break;

```



```php

        case "Wed":
            echo "Today is Wednesday";
            break;

        case "Thu":
            echo "Today is Thursday";
            break;

        case "Fri":
            echo "Today is Friday";
            break;

        case "Sat":
            echo "Today is Saturday";
            break;

        case "Sun":
            echo "Today is Sunday";
            break;

        default:
            echo "Wonder which day is this ?";
    }
?>

    </body>
</html>
```

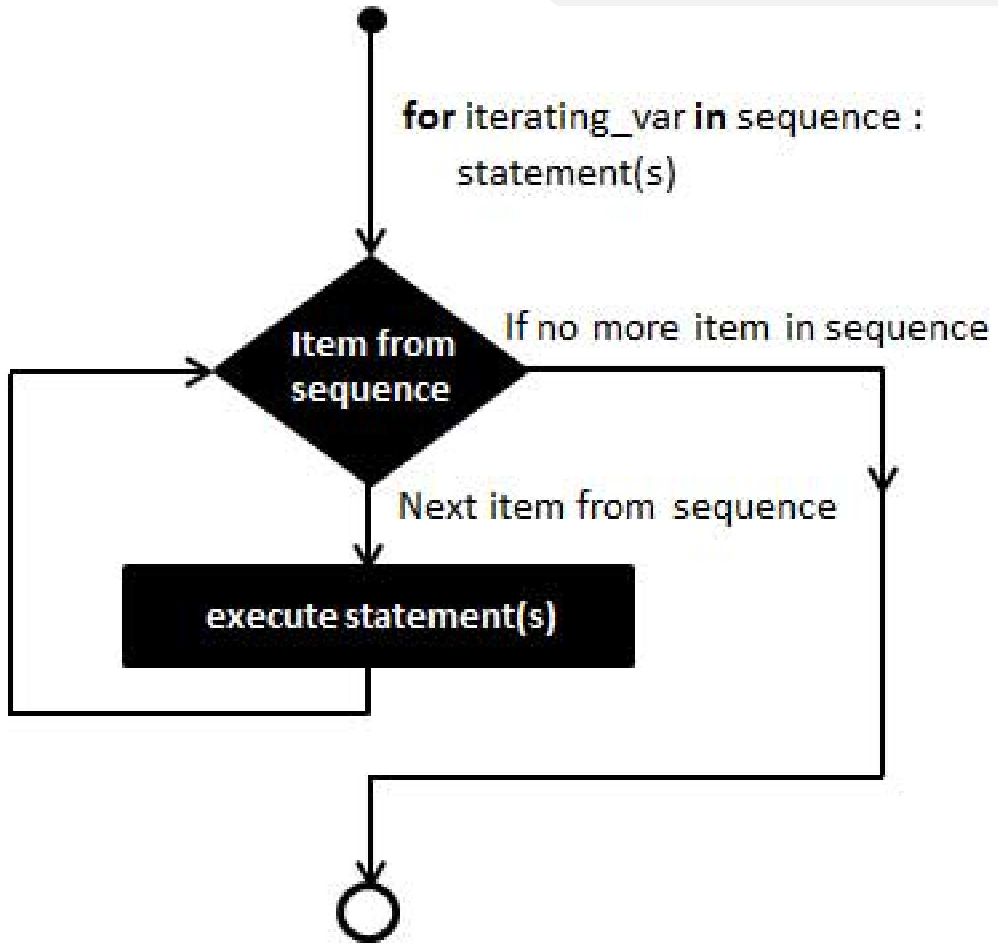
Loop

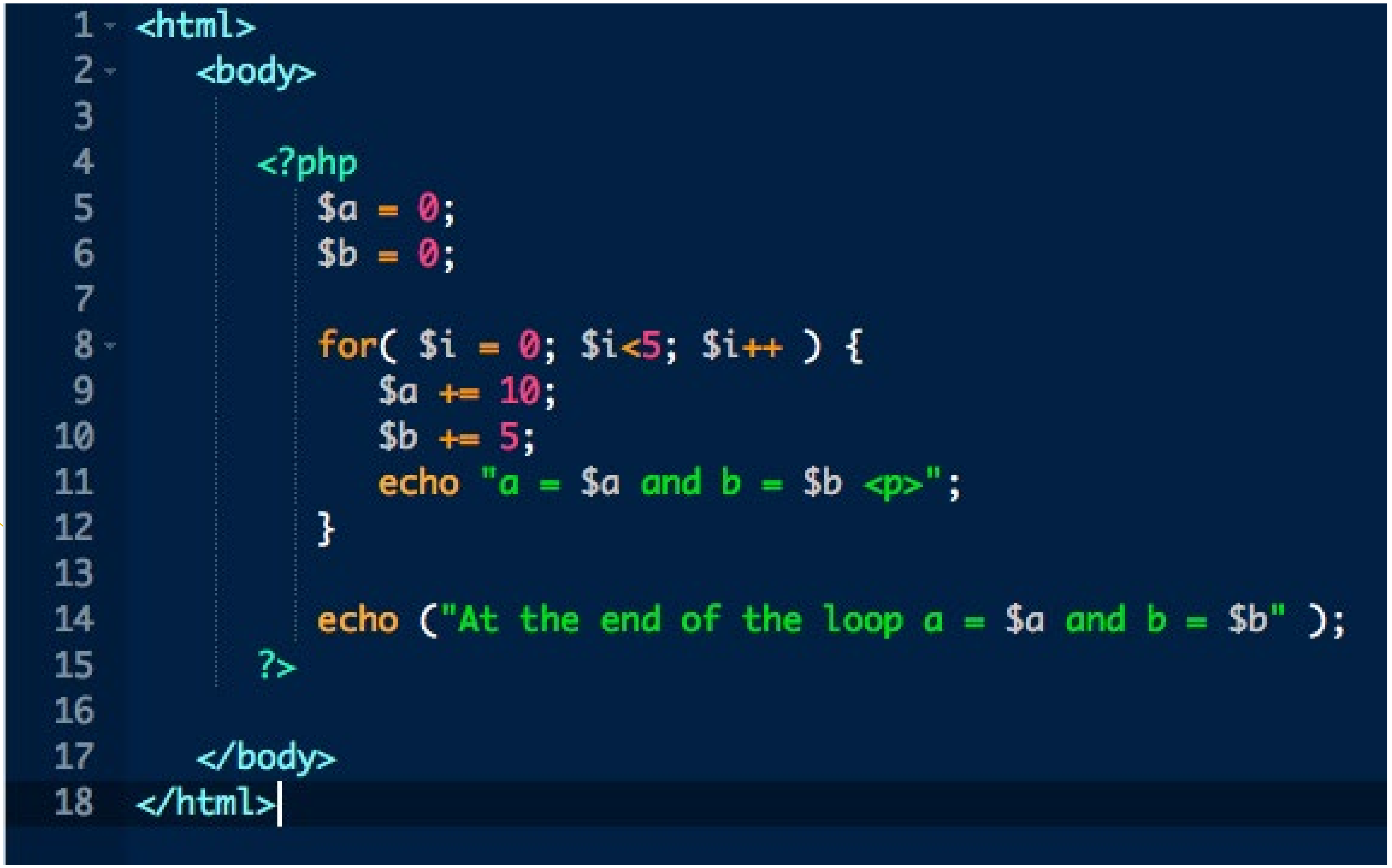
For Loop

- **for** – loops through a block of code a specified number of times.

for (*initialization*; *condition*; *increment*)

{ *code to be executed;* }





```php
<html>
    <body>

        <?php
            $a = 0;
            $b = 0;

            for( $i = 0; $i<5; $i++ ) {
                $a += 10;
                $b += 5;
                echo "a = $a and b = $b <p>";
            }

            echo ("At the end of the loop a = $a and b = $b" );
        ?>

    </body>
</html>
```

```
a = 10 and b = 5

a = 20 and b = 10

a = 30 and b = 15

a = 40 and b = 20

a = 50 and b = 25

At the end of the loop a = 50 and b = 25
```

While and Do … While Loop

- **while** – loops through a block of code if and as long as a specified condition is true.

while (*condition*) {

code to be executed;

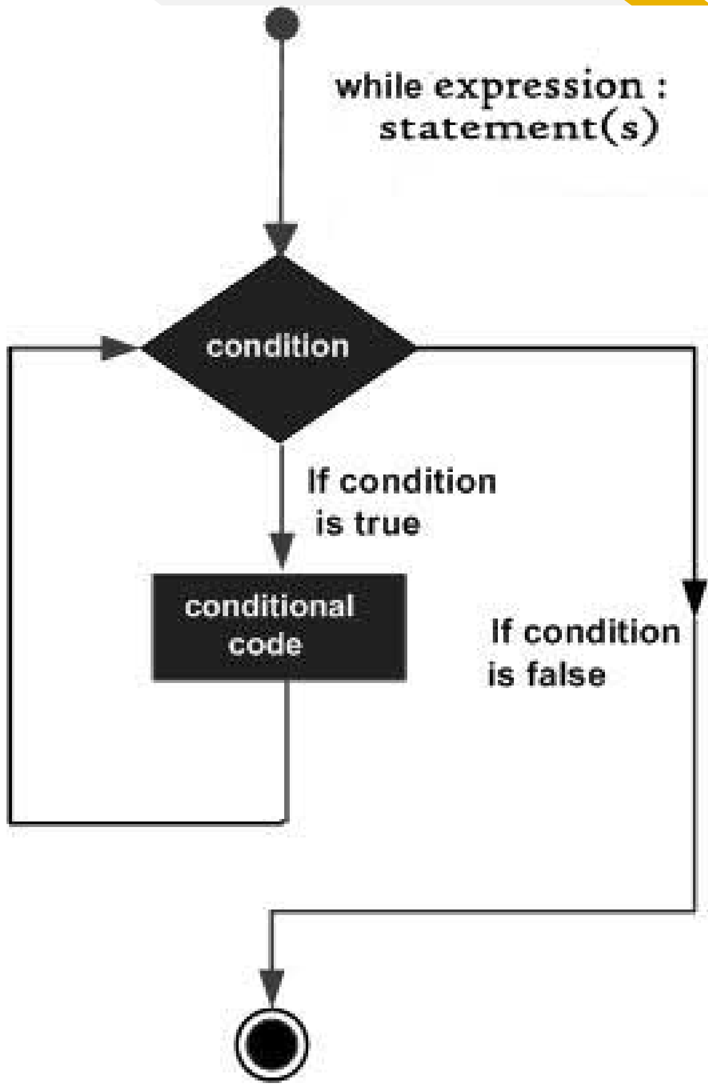
}

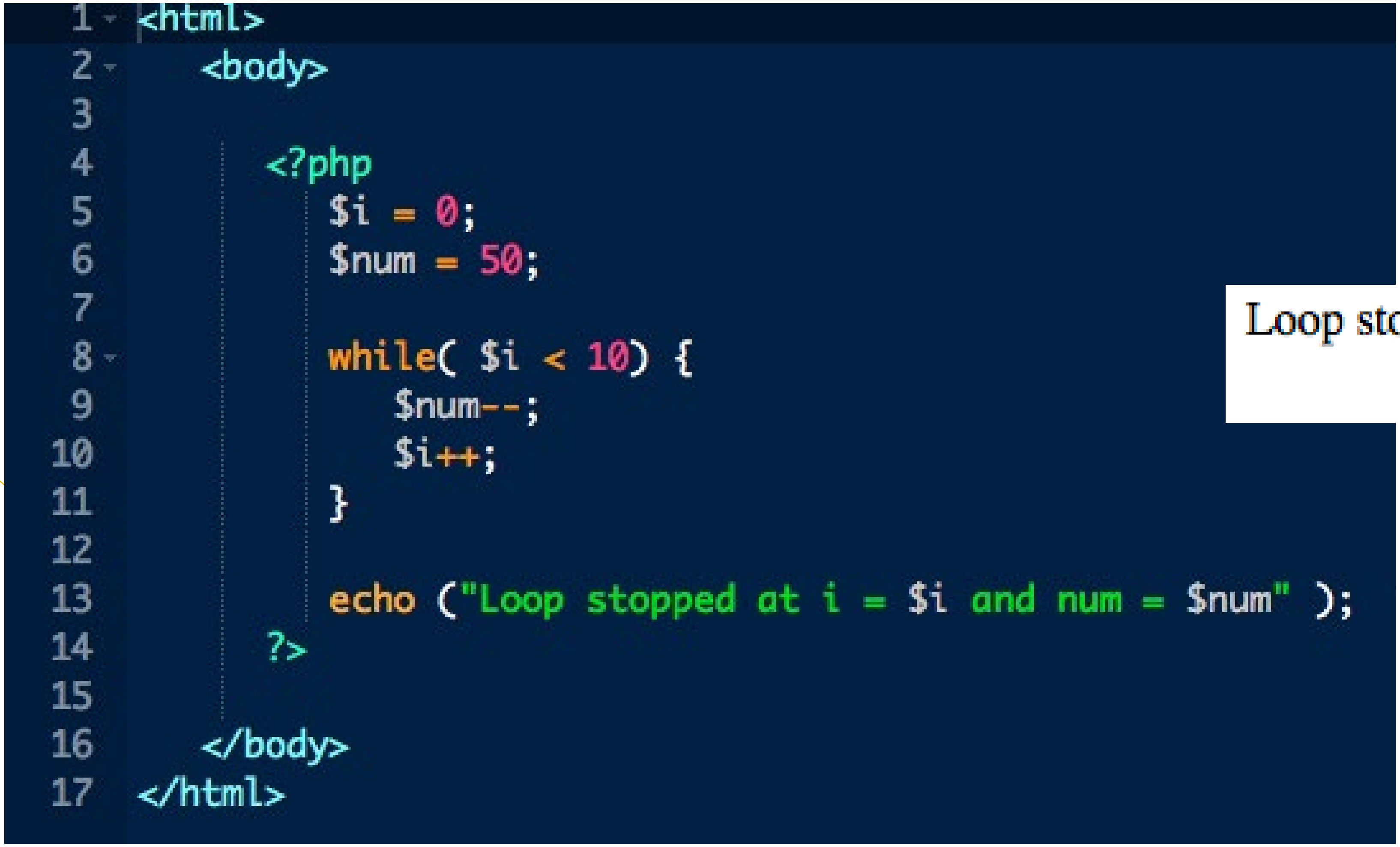
- **do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true.

do {

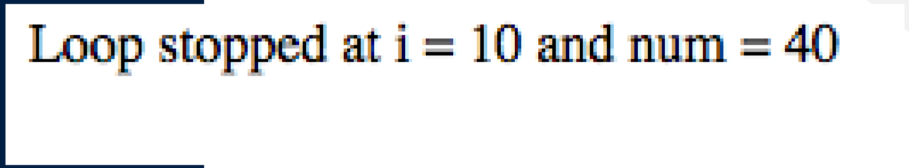
code to be executed;

} while (*condition*);

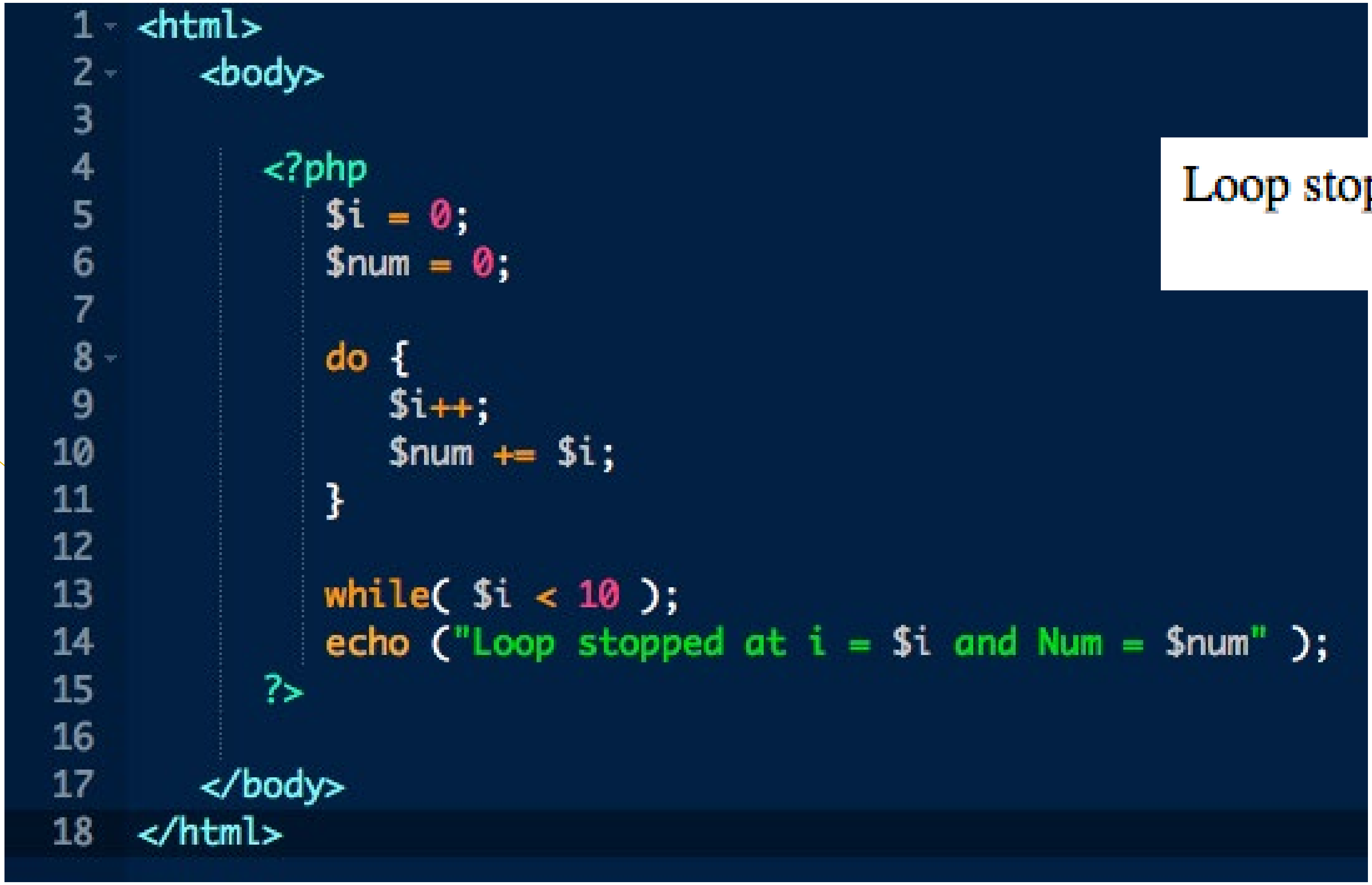




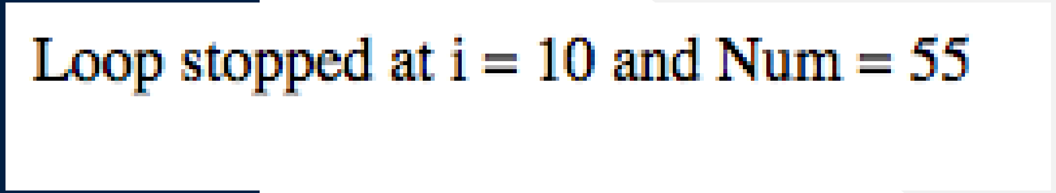
```
<html>
    <body>

        <?php
            $i = 0;
            $num = 50;

            while( $i < 10) {
                $num--;
                $i++;
            }

            echo ("Loop stopped at i = $i and num = $num" );
        ?>

    </body>
</html>
```



```
Loop stopped at i = 10 and num = 40
```



```
<html>
    <body>

        <?php
            $i = 0;
            $num = 0;

            do {
                $i++;
                $num += $i;
            }

            while( $i < 10 );
            echo ("Loop stopped at i = $i and Num = $num" );
        ?>

    </body>
</html>
```

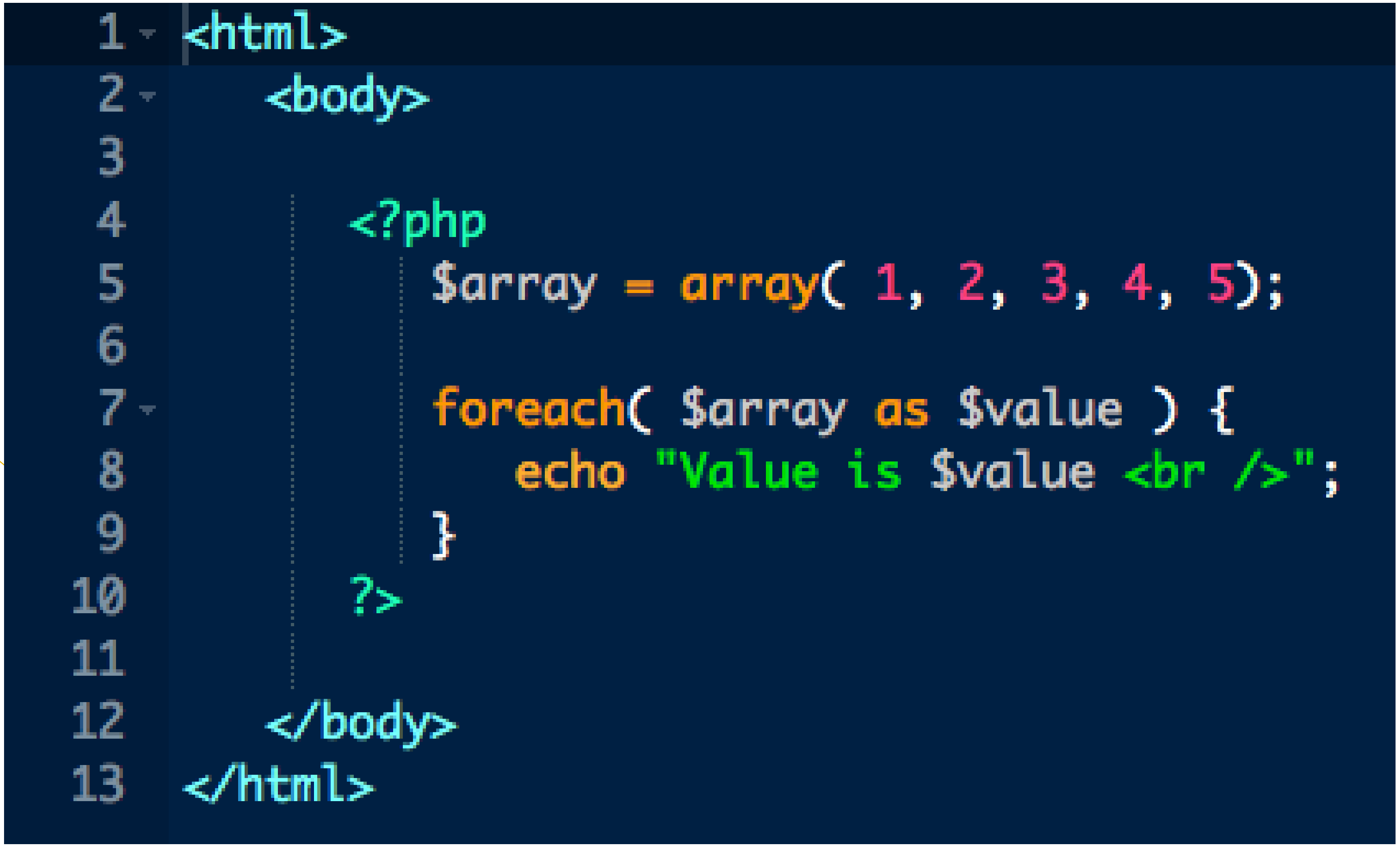


```
Loop stopped at i = 10 and Num = 55
```

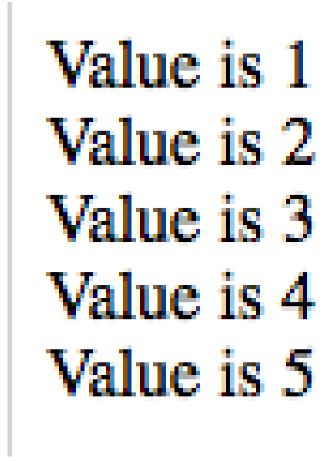
foreach

- **foreach** – loops through a block of code for each element in an array.

```
foreach (array as value) {
code to be executed;
}
```



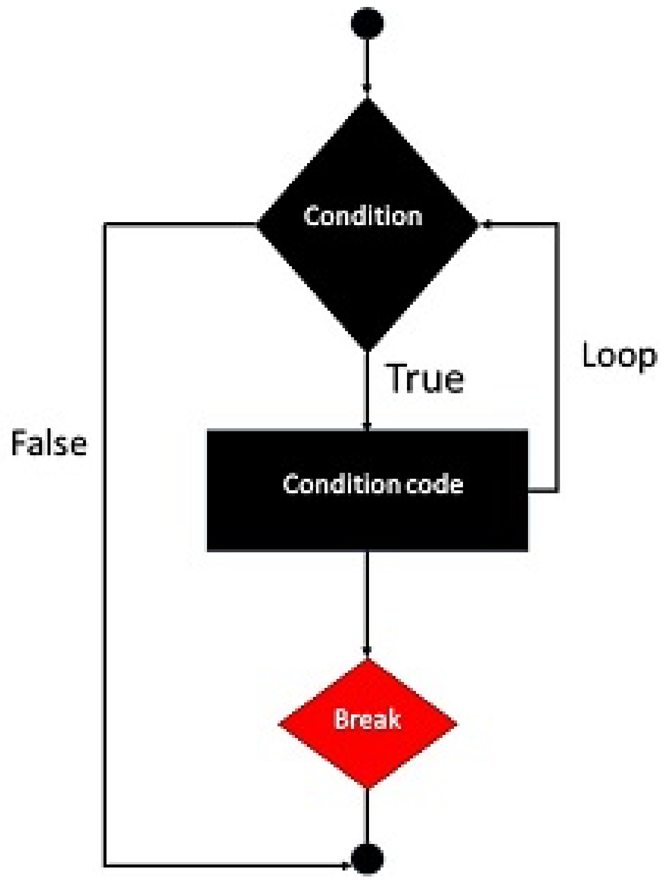
```php
<html>
    <body>

        <?php
            $array = array( 1, 2, 3, 4, 5);

            foreach( $array as $value ) {
                echo "Value is $value <br />";
            }
        ?>

    </body>
</html>
```



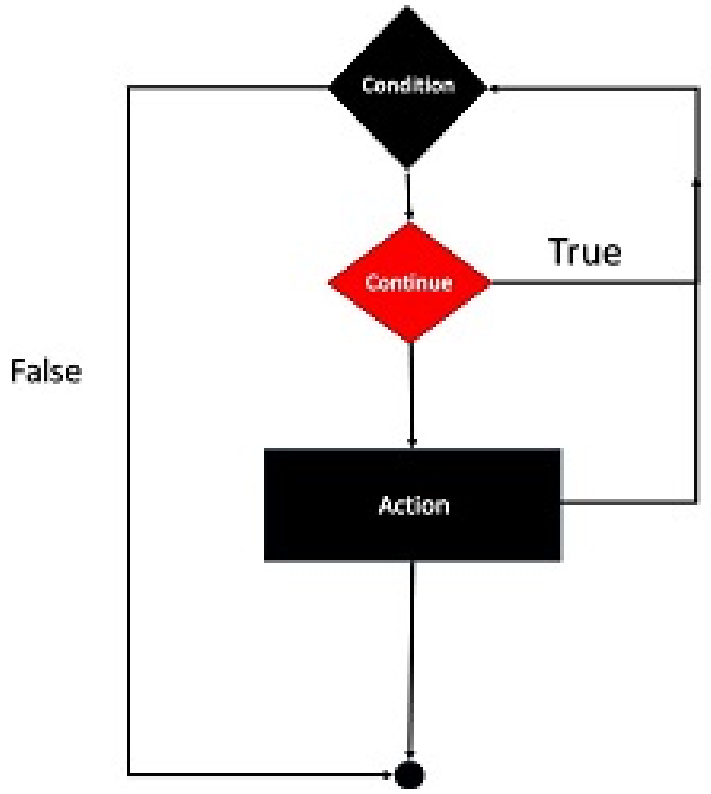
```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

break and continue

- The PHP **break** keyword is used to terminate the execution of a loop prematurely.



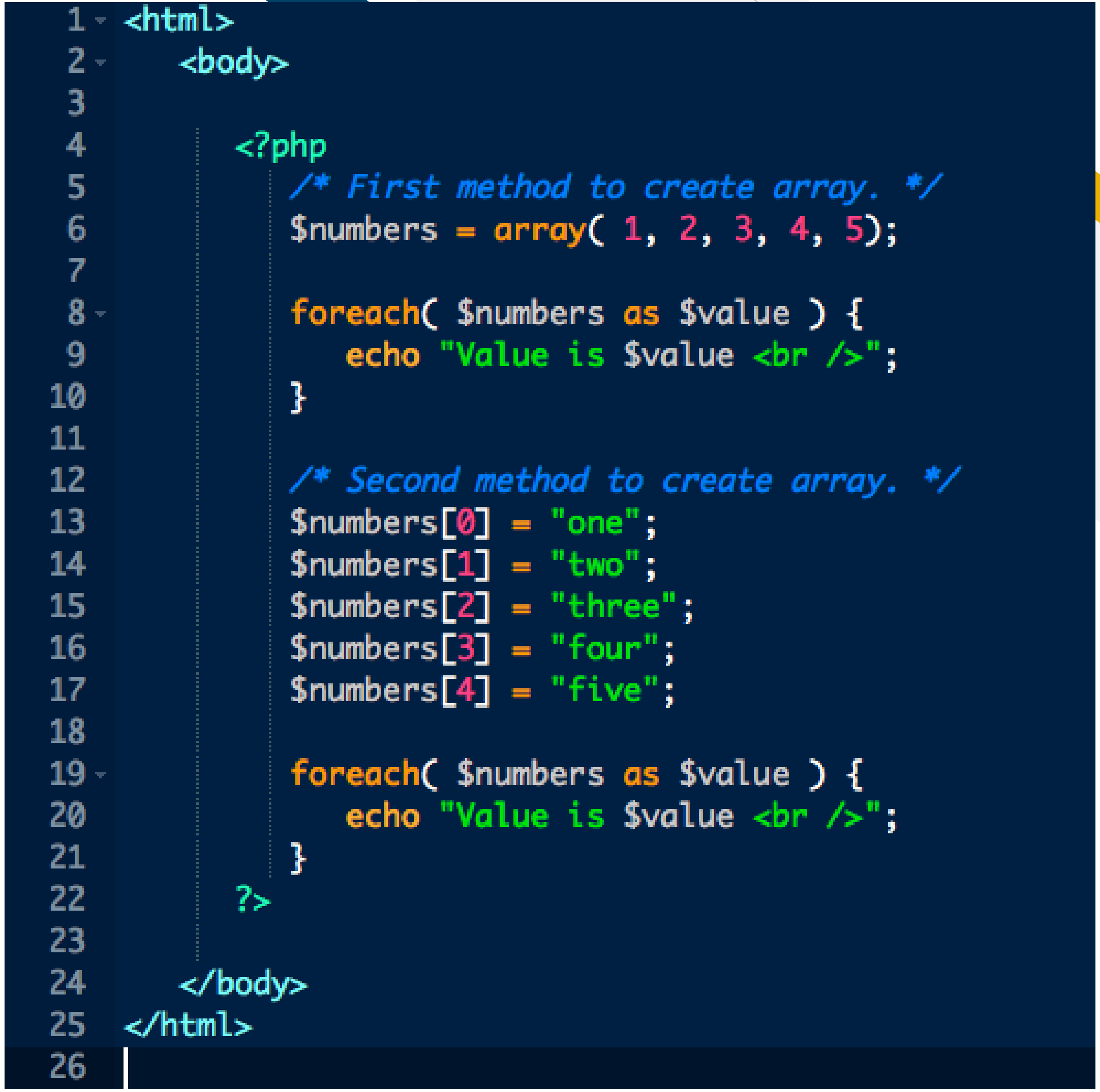
- The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.



Array

Numeric Array

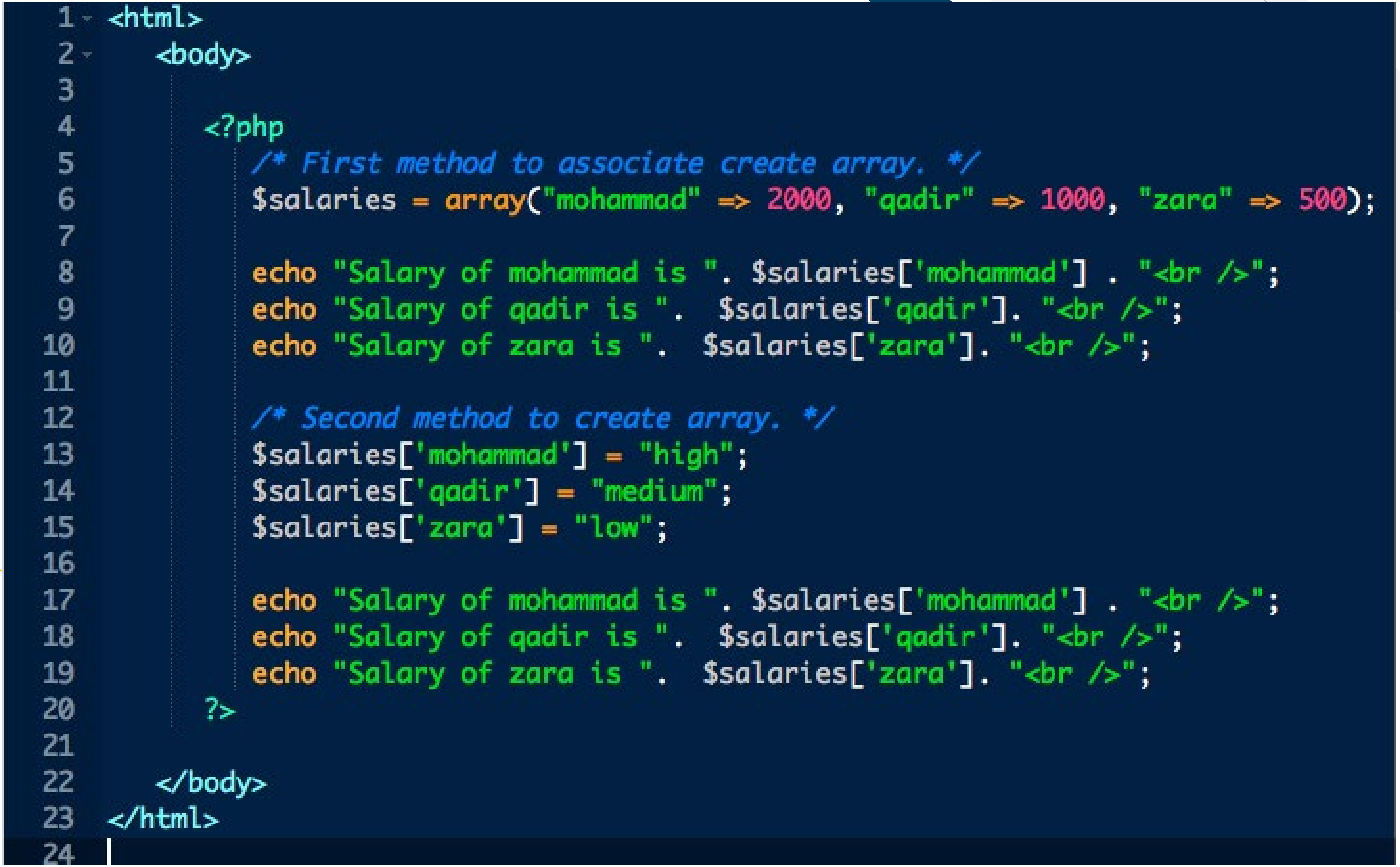
- **Numeric array** – An array with a numeric index. Values are stored and accessed in linear fashion.



```
<html>
   <body>

      <?php
         /* First method to create array. */
         $numbers = array( 1, 2, 3, 4, 5);

         foreach( $numbers as $value ) {
            echo "Value is $value <br />";
         }

         /* Second method to create array. */
         $numbers[0] = "one";
         $numbers[1] = "two";
         $numbers[2] = "three";
         $numbers[3] = "four";
         $numbers[4] = "five";

         foreach( $numbers as $value ) {
            echo "Value is $value <br />";
         }
      ?>

   </body>
</html>

```

Associative Arrays

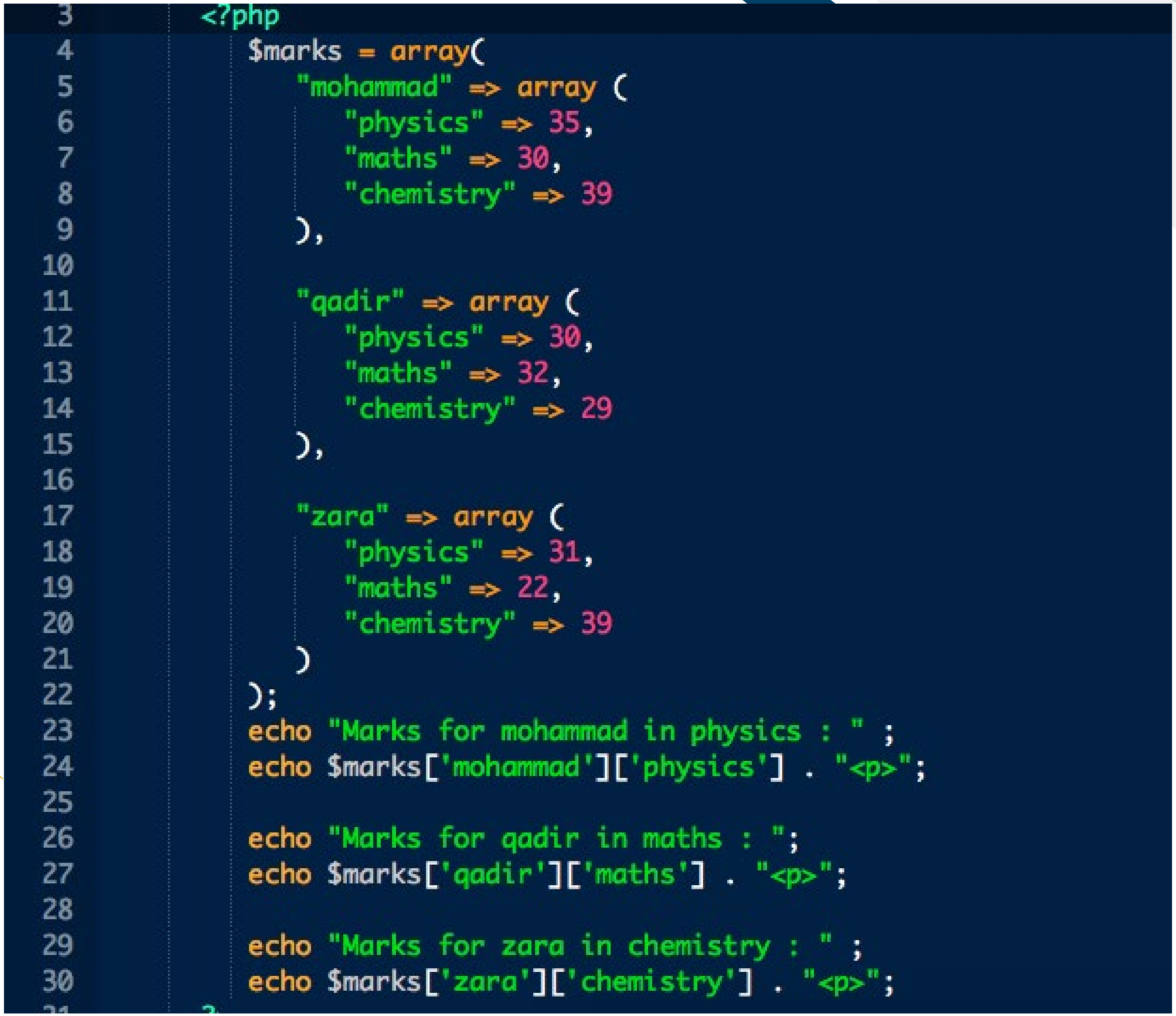
- **Associative array** – An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- Don't keep associative array inside double quote while printing otherwise it would not return any value.



```
<html>
   <body>

      <?php
         /* First method to associate create array. */
         $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);

         echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
         echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";

         /* Second method to create array. */
         $salaries['mohammad'] = "high";
         $salaries['qadir'] = "medium";
         $salaries['zara'] = "low";

         echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
         echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";
      ?>

   </body>
</html>

```

Multidimensional Arrays

- **Multidimensional array** – An array containing one or more arrays and values are accessed using multiple indices
- Each element in the main array can also be an array.
- And each element in the sub-array can be an array, and so on.
- Values in the multi-dimensional array are accessed using multiple index.



```php
<?php
    $marks = array(
        "mohammad" => array (
            "physics" => 35,
            "maths" => 30,
            "chemistry" => 39
        ),

        "qadir" => array (
            "physics" => 30,
            "maths" => 32,
            "chemistry" => 29
        ),

        "zara" => array (
            "physics" => 31,
            "maths" => 22,
            "chemistry" => 39
        )
    );
    echo "Marks for mohammad in physics : " ;
    echo $marks['mohammad']['physics'] . "<p>";

    echo "Marks for qadir in maths : ";
    echo $marks['qadir']['maths'] . "<p>";

    echo "Marks for zara in chemistry : " ;
    echo $marks['zara']['chemistry'] . "<p>";
```

Thank you for Your Attention.