Web programming

The PHP language

Our objective

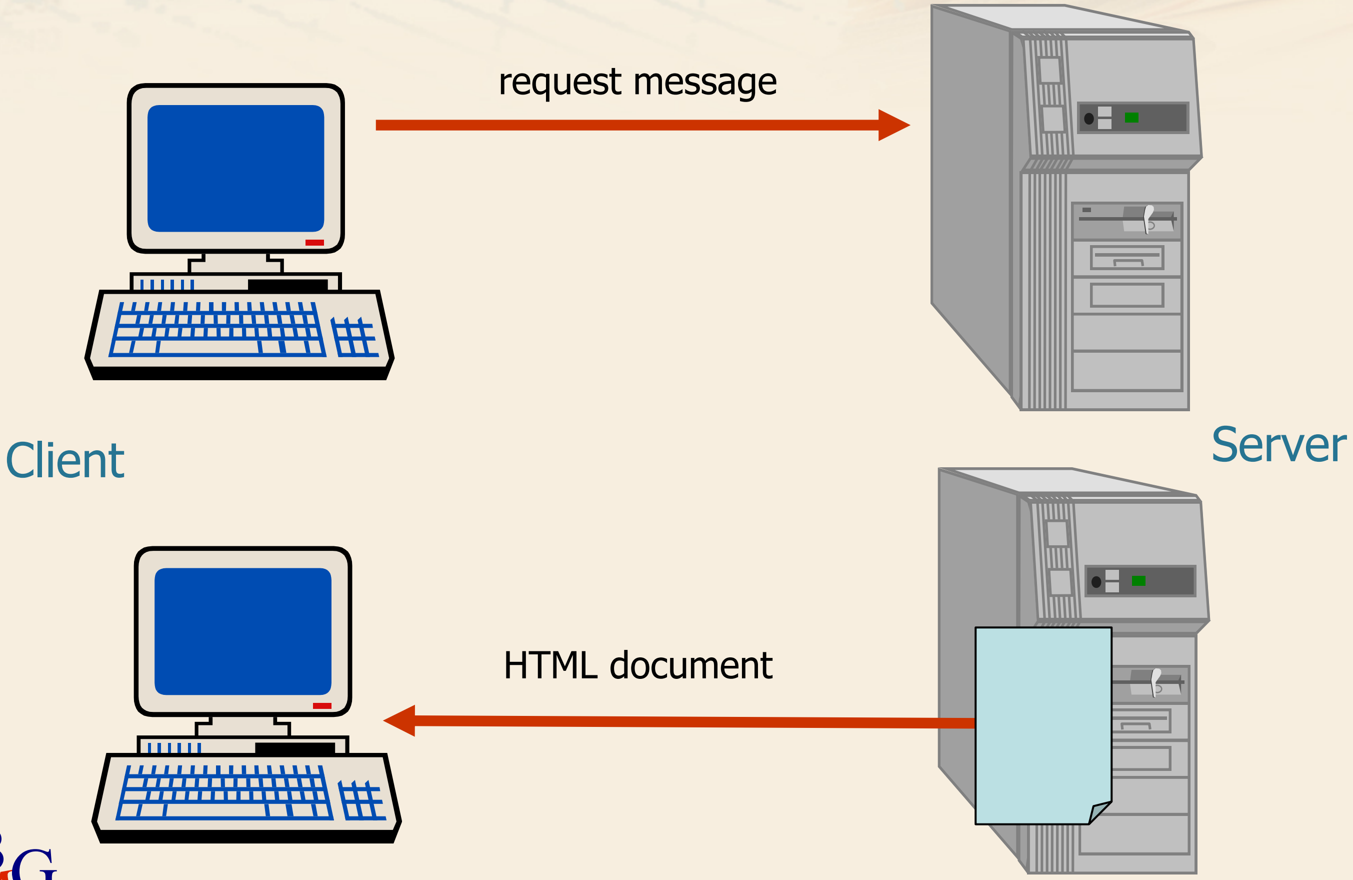
- Teaching you everything about PHP? Not exactly
- Goal: teach you how to interact with a database via web
  - Access data inserted by users into HTML forms
  - Interact with a DBMS (MySQL in particular): connect to a database, execute a query, store the result of the query…
  - Access the tables returned by the DBMS
  - Assemble the HTML page on the browser, composed by HTML instructions and data extracted from the database

Contents

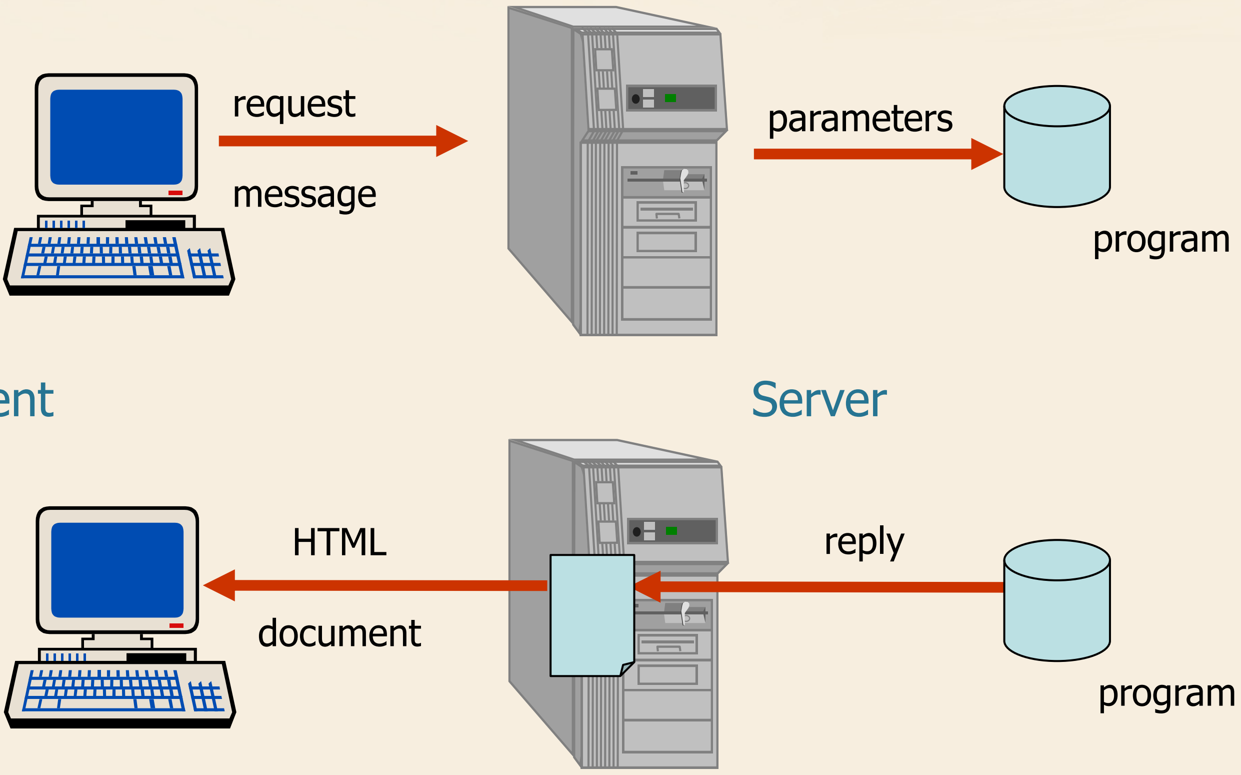
What is PHP

- Born in 1994
  - Personal Home Page, today known as PHP Hypertext Preprocessor
- Created specifically for the development of dynamic webpages
- Many useful resources, e.g.
  - https://www.w3schools.com/php7
  - https://www.php.net

Static webpages



Dynamic webpages



Primary goal

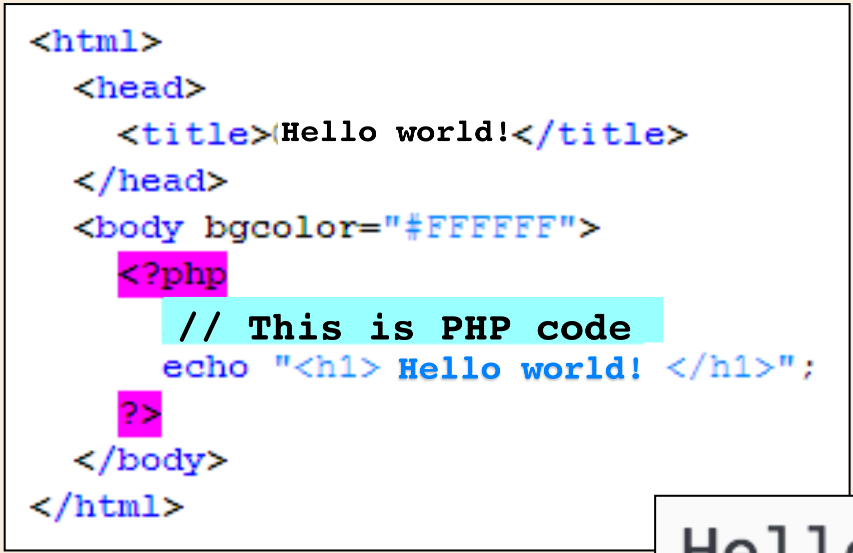
- PHP's primary goal is to generate HTML code
  - In particular, generating HTML code by the results of an elaboration, that depend on the user input on the database contents, …
- The PHP code is inserted inside the HTML code
  - The PHP code is executed on the Web server and the result (HTML and script result) is sent back to the browser

Why using PHP?

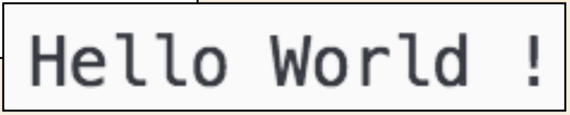
- Available for many platforms, different in
  - Hardware (Intel, Sparc, Mac, etc....)
  - Operative system (Linux, Unix, Windows, etc...)
  - Web server (Apache, IIS, IPlanet, etc...)
- PHP code is "highly portable"
- The PHP interpreter is Open Source
  - Free, wide availability of tools, support,developers, community of users
- Pretty easy to learn, very simple if you already know C
- Able to interact with various Database Management Systems (MySql, Postgres, Oracle, ...)

First example

- Text file with .php extension

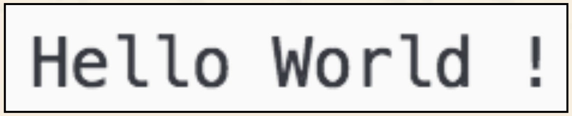


```
<html>
  <head>
    <title>Hello world!</title>
  </head>
  <body bgcolor="#FFFFFF">
    <?php
      // This is PHP code
      echo "<h1> Hello world! </h1>";
    ?>
  </body>
</html>
```

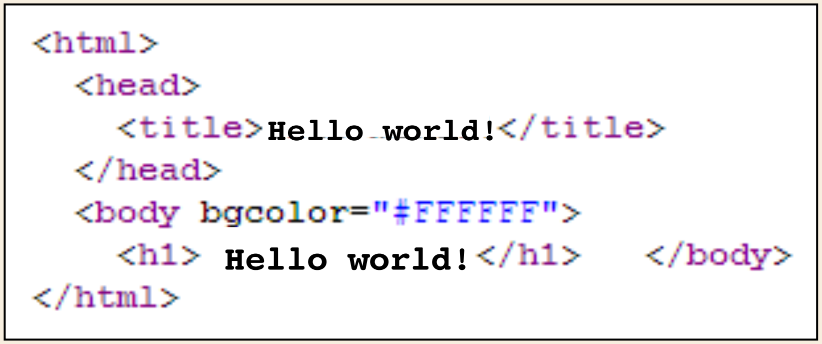


```
Hello World !
```

First example


Hello World !

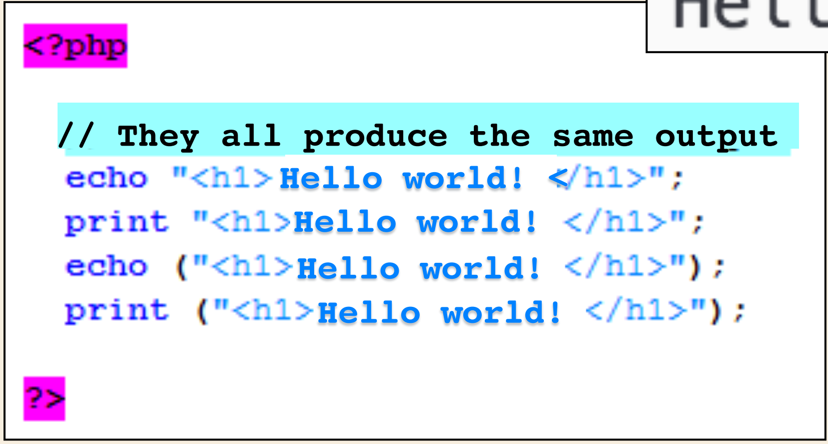
- If I look at the source code on the browser…


```
<html>
  <head>
    <title>Hello world!</title>
  </head>
  <body bgcolor="#FFFFFF">
    <h1> Hello world!</h1>   </body>
</html>
```

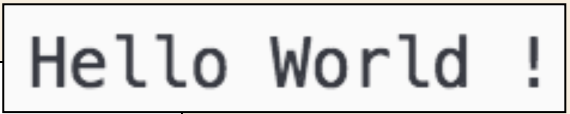
- Why?
  - The browser shows the result of the execution of the PHP file, NOT the PHP file

"Printing" strings

- One of the most important (and frequent) tasks of PHP code is to create HTML code that will be displayed on the browser
  - echo and print constructs



```php
<?php

  // They all produce the same output
  echo "<h1>Hello world! </h1>";
  print "<h1>Hello world! </h1>";
  echo ("<h1>Hello world! </h1>");
  print ("<h1>Hello world! </h1>");

?>
```



```
Hello World !
```

A quick digression: XAMPP

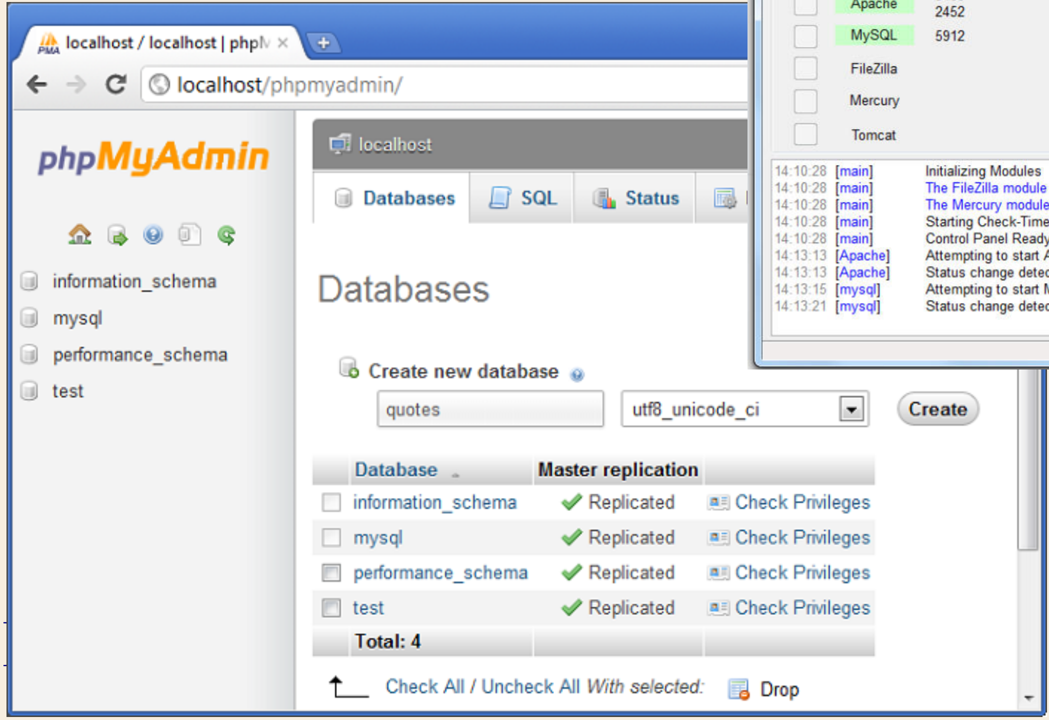
- XAMPP is a cross-platform Apache distribution that includes:
  - A web server (Apache)
  - A database management system (MySQL/MariaDB)
  - PHP and Perl script interpreters
  - A graphical administrator of MySQL database (phpMyAdmin)
- It can be used as a web-database development environment, thus making server-side scripts (e.g., PHP) and programs (e.g., DBMS, Web server) work locally

A quick digression: XAMPP

- XAMPP installs all necessary software for the development and deployment of a local web site
  - The PC becomes client and server
- The Apache web server automatically creates a virtual domain (with local validity) at the localhost address (http://127.0.0.1 or http://localhost)
  - Being connected to the Internet is not needed to use XAMPP

XAMPP : DB administration

- Allows to manage databases
  - Graphical interface



localhost / localhost | phpM

localhost/phpmyadmin/

phpMyAdmin

- information_schema
- mysql
- performance_schema
- test

localhost

Databases | SQL | Status

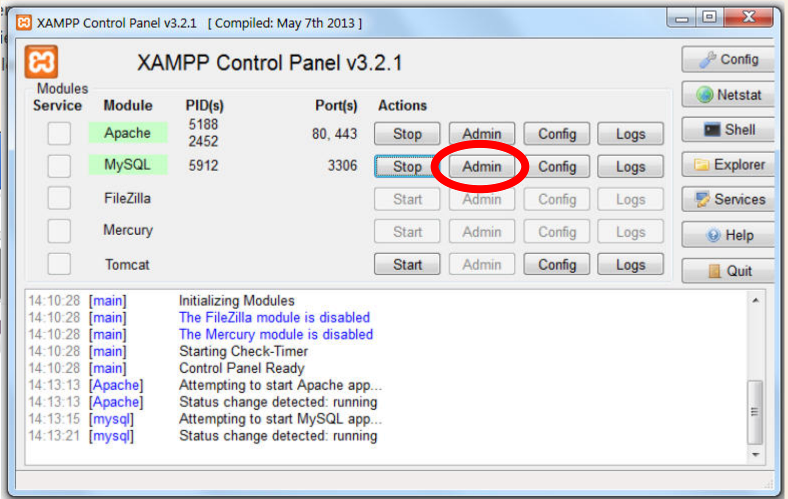
Databases

Create new database

quotes | utf8_unicode_ci | Create

	Database	Master replication	
☐	information_schema	✓ Replicated	Check Privileges
☐	mysql	✓ Replicated	Check Privileges
☐	performance_schema	✓ Replicated	Check Privileges
☐	test	✓ Replicated	Check Privileges
	Total: 4		

Check All / Uncheck All *With selected:* Drop



XAMPP Control Panel v3.2.1 [Compiled: May 7th 2013]

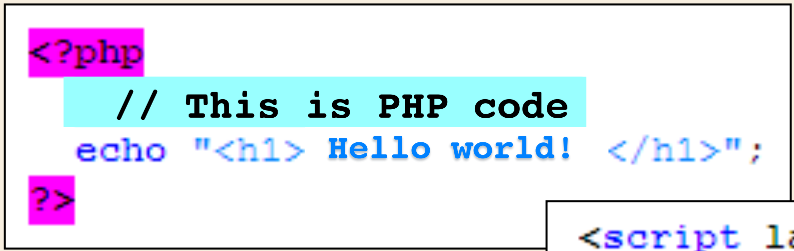
XAMPP Control Panel v3.2.1

Modules Service	Module	PID(s)	Port(s)	Actions			
☐	Apache	5188 2452	80, 443	Stop	Admin	Config	Logs
☐	MySQL	5912	3306	Stop	Admin	Config	Logs
☐	FileZilla			Start	Admin	Config	Logs
☐	Mercury			Start	Admin	Config	Logs
☐	Tomcat			Start	Admin	Config	Logs

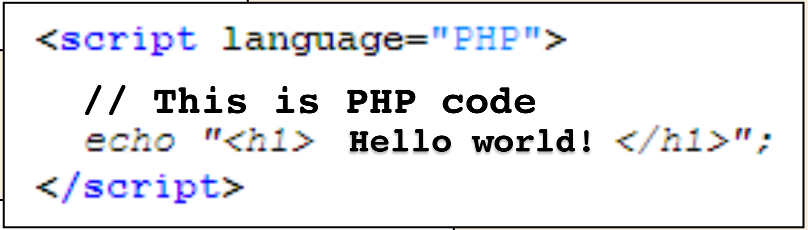
```
14:10:28  [main]      Initializing Modules
14:10:28  [main]      The FileZilla module is disabled
14:10:28  [main]      The Mercury module is disabled
14:10:28  [main]      Starting Check-Timer
14:10:28  [main]      Control Panel Ready
14:13:13  [Apache]    Attempting to start Apache app...
14:13:13  [Apache]    Status change detected: running
14:13:15  [mysql]     Attempting to start MySQL app...
14:13:21  [mysql]     Status change detected: running
```

Tags to include PHP code

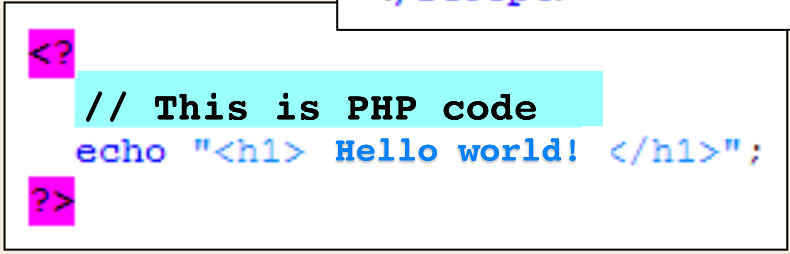
- PHP code can be insterted in any point of a HTML page
- Needs to be enclosed by tags



```
<?php
    // This is PHP code
  echo "<h1> Hello world! </h1>";
?>
```



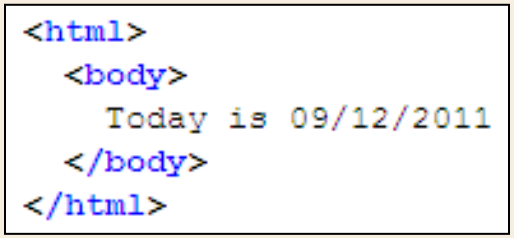
```
<script language="PHP">
  // This is PHP code
  echo "<h1> Hello world! </h1>";
</script>
```



```
<?
  // This is PHP code
  echo "<h1> Hello world! </h1>";
?>
```

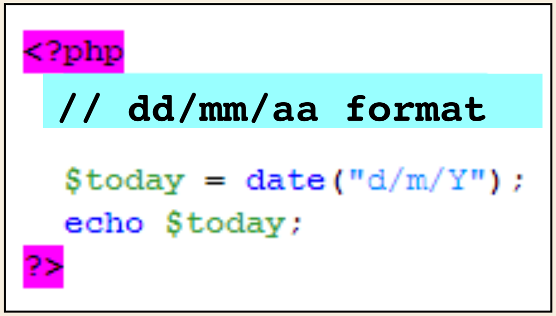
Another example

- Display the current date
- In a static way
  - And tomorrow?



```
<html>
  <body>
    Today is 09/12/2011
  </body>
</html>
```

- In a dynamic way
  - Updates in real time

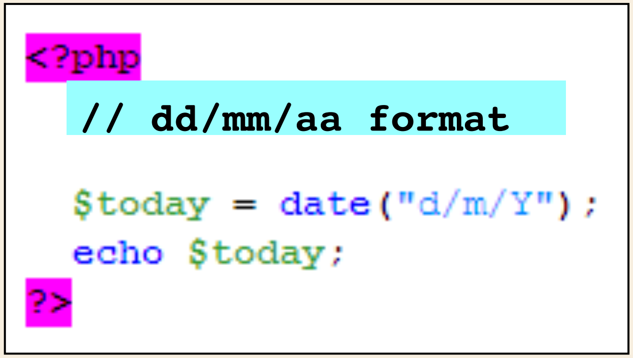


```
<?php
 // dd/mm/aa format

  $today = date("d/m/Y");
  echo $today;
?>
```

Code analysis

- In a script we use

Comments:	//.........
Variables:	$today
Operators and language constructs:	echo
Functions:	date()



```
<?php
   // dd/mm/aa format

   $today = date("d/m/Y");
   echo $today;
?>
```

Variables

- A variable is a symbol or a name that represents a value
- A variable can represent different types of value
  - Integer number, real number, character, …
  - The data type can change during the execution of the program
- When a program is executed, variables are replaced by real data
  - The same program can elaborate different types of data sets in this way

Variables

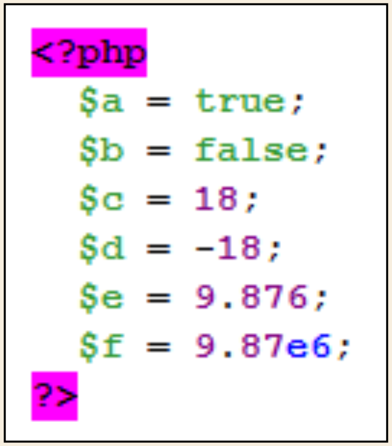
- In PHP the name of variables is preceded by the dollar symbol (‘$’)
- PHP does not require that variables are declared before their usage
  - Higher flexibility with respect to other programming languages

```php
<?php
  $a = 5;
?>
```

```php
<?php
  $a = 9;
  $b = 4;
  $c = $a * $b;
  echo "The result of the operation (9 * 4) is: ";
  echo $c;
?>
```

Data types

- PHP supports different data types
  - Boolean: true or false
  - Integer: decimal numbers
  - Float: floating point numbers
  - String
  - Array
  - Object
  - Resource
- Data types don't need to be set by the programmer but they are automatically detected by the PHP intepreter



```
<?php
  $a = true;
  $b = false;
  $c = 18;
  $d = -18;
  $e = 9.876;
  $f = 9.87e6;
?>
```

Data types

- Data types are not specified by the programmer, but they are automatically inferred by the PHP interpreter
  - It is possible to check the type of a variable using: **is_int()**, **is_float()**, **is_bool()**, **is_string()**
- PHP supports both implicit and explicit casting (C like syntax)

```
<?php
   $a = 56;
   $b = 12;
   $c = $a/$b;  // $c è 4.66666 (float)
   $d = (int)($a/$b);  // $d è 4 (int)
?>
```

Strings

- A string is a sequence of characters, with no length limitation
- Included between a couple of single or double quotes
  - If double quotes are used (""), the string content gets expanded (or, technically, "interpolated")

```php
<?php
  $num = 10;
  $stringa = "<strong>The number is$num</strong>";
  echo $stringa;
?>
```

```
The number is 10
```

Arrays

- An array is a complex variable that contains a series of values, each of them characterized by a key (or index) that unambiguously identifies it
- PHP supports both scalar and associative arrays
  - Scalar arrays identify each element with its position number in a sequence
  - Associative arrays identify each element with a key in an univocal way

Arrays

- Example of a scalar array

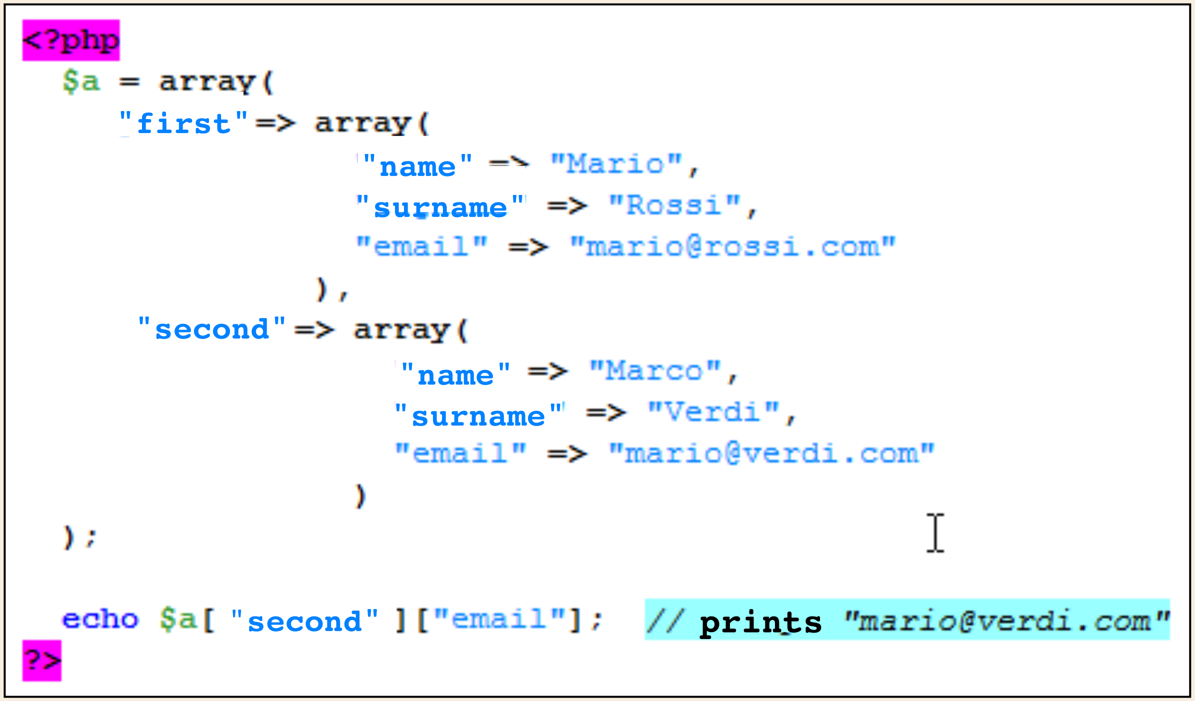
```
<?php
  $colors = array('white', 'black', 'yellow', 'green', 'red');

   echo $colors[1]; // prints 'black'
   echo $colors[4]; // prints 'red'
?>
```

- Example of an associative array
  - The key can be a string or an integer

```
<?php
  $a = array(
     "name" => "Mario",
    "surname" => "Rossi",
    "email" => "mario@rossi.com"
  );

  echo $a["name"];   // prints "Mario"
  echo $a["email"];  // prints "mario@rossi.com"
?>
```

Arrays

- Multidimensional arrays are possible



```
<?php
  $a = array(
     "first"=> array(
                    "name" => "Mario",
                    "surname" => "Rossi",
                    "email" => "mario@rossi.com"
                  ),
      "second"=> array(
                     "name" => "Marco",
                     "surname" => "Verdi",
                     "email" => "mario@verdi.com"
                   )
  );

  echo $a[ "second" ]["email"];   // prints "mario@verdi.com"
?>
```

Arrays

- Array elements can also be heterogeneous

```
<?php
  $mix = array( 1,"hello", 3.14, array( 1, 2, 3 ) );
?>
```

- In PHP it's very easy to add or remove elements of an array

```
<?php
  $x = array(25, 50);

  $x[ ] = 75;      // adds an element to the first available position
  $x[4] = 125;     // adds an element on the specified position
  echo $x[3];      // Error!!! (The element does not exist)

  unset($x[1]);    // removes the specified element
  unset($x);       // removes the entire array
?>
```

DBMG

Useful functions for arrays

- is_array(array): return True if the parameter is an array
- count(array): return the number of elements in the array
- sort(array): it sort the array. It is possible to specify the order
- array_key_exists(key,array): It verify that a specific ***key*** exists in the ***array***

Expressions and operators

- Arithmetic operators

```
<?php
   $a = $x + 7;  // addition
   $b = $x - 2;  // subtraction
   $c = $x * 6;  // multiplication
   $d = $x / 2;  // division
   $e = $x % 4;  // module (division remainder)

   $x += 4;// increment $x by 4 (equivalent to $x = $x + 4)
   $x -= 3;// decrement $x by 3 (equivalent to $x = $x + 3)
   $x /= 5;// equivalent to $x = $x / 5
   $x *= 4;// equivalent to $x = $x * 4
   $x %= 2;// equivalent to $x = $x % 3

   $a++; // increment by 1
   ++$a; // increment by 1
   $a--; // decrement by 1
   --$a; // decrement by 1
?>
```

Expressions and operators

- Logical operators

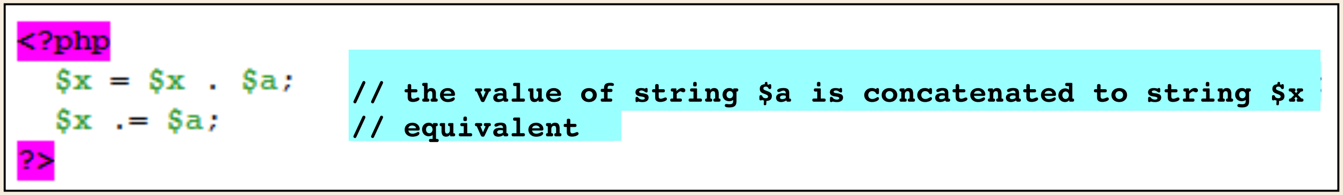
```
<?php
  $x = $a && $b;  // logical and
  $x = $a || $b;  // logical or
  $x = $a xor $b; // logical xor
  $x = !$a;       // logical not
?>
```

- Comparison operators

```
<?php
  if ($a == $b) ...   // equal
  if ($a != $b) ...   // not equal
  if ($a >  $b) ...   // greater
  if ($a >= $b) ...   // greater or equal
  if ($a <  $b) ...   // less
  if ($a <= $b) ...   // less or equal
?>
```

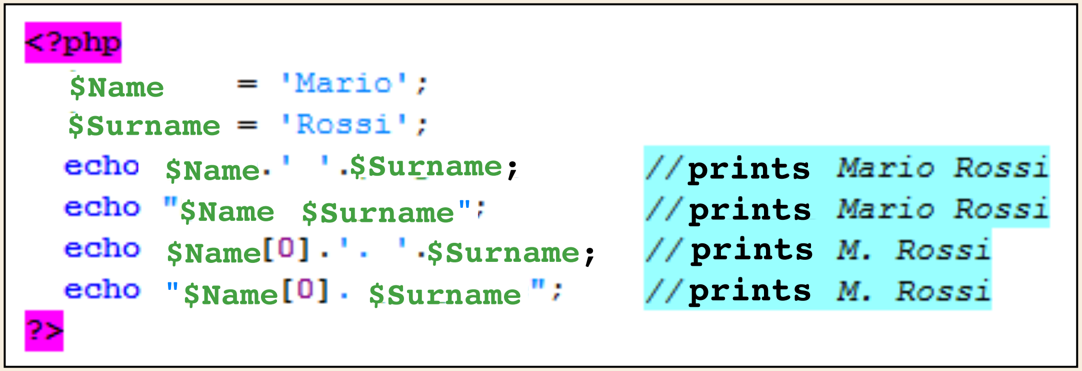
Expressions and operators

- String operations
  - Concatenation



```
<?php
  $x = $x . $a;       // the value of string $a is concatenated to string $x
  $x .= $a;           // equivalent
?>
```

- Example



```
<?php
   $Name     = 'Mario';
   $Surname = 'Rossi';
   echo $Name.' '.$Surname;          //prints  Mario Rossi
   echo "$Name $Surname";            //prints  Mario Rossi
   echo $Name[0].'. '.$Surname;      //prints  M. Rossi
   echo "$Name[0]. $Surname";        //prints  M. Rossi
?>
```

Control structures

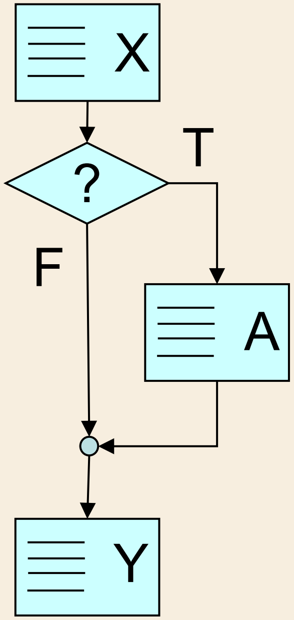
- Allow the conditional execution of parts of the program
- Allow the iterative execution of parts of the program
- Evaluate certain conditions
- PHP control structures
  - if, if..else, if..elseif
  - switch
  - while, do..while
  - for
  - foreach

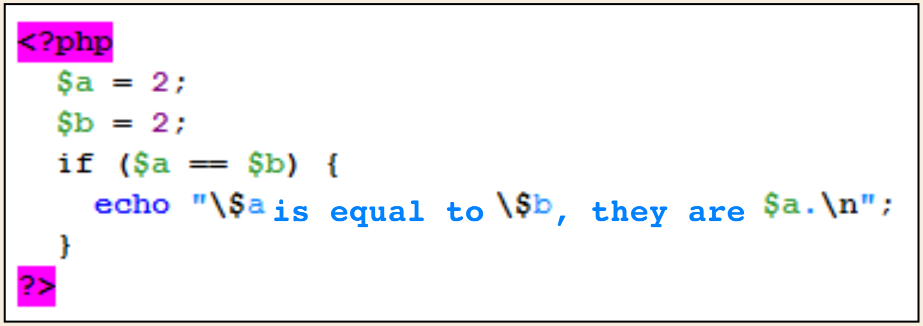
Conditions

- A condition is an expression that generates a boolean value (true or false)
  - They use comparison operators and Boolean operators
- The following values are equivalent to “false”
  - The Boolean value false
  - The integer number 0 and the real number 0.0
  - The empty string (“”) and the “0” string
  - An empty array
- Each other value is considered true

The IF construct

- If the condition expressed by the IF block is true, the piece of code is executed

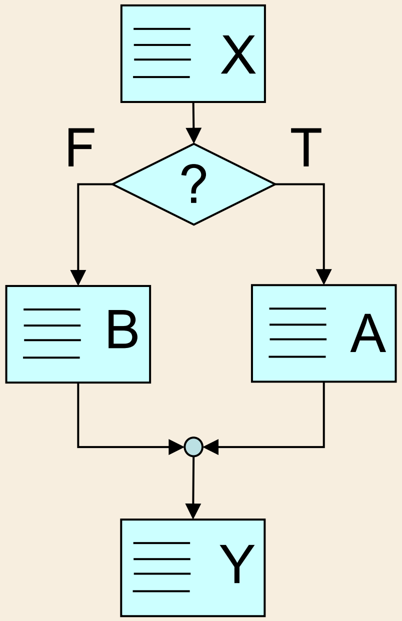




```
<?php
  $a = 2;
  $b = 2;
  if ($a == $b) {
    echo "\$a is equal to \$b, they are $a.\n";
  }
?>
```

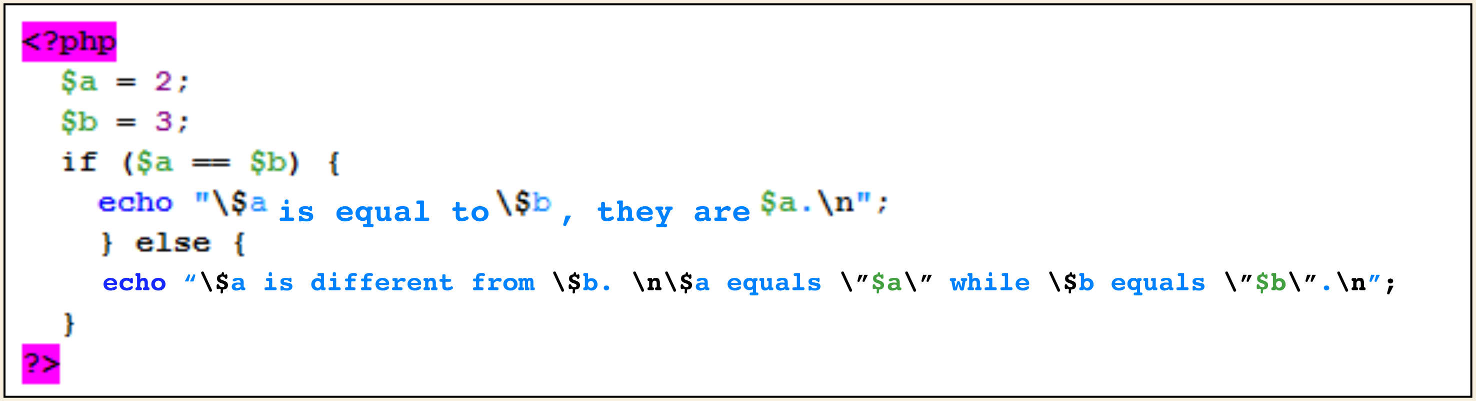
The IF .. ELSE construct

- If the condition expressed by the IF block is true, the sequence of operations follows the THEN branch, otherwise the ELSE branch



The IF .. ELSE construct

- If the condition expressed by the IF block is true, the sequence of operations follows the THEN branch, otherwise the ELSE branch

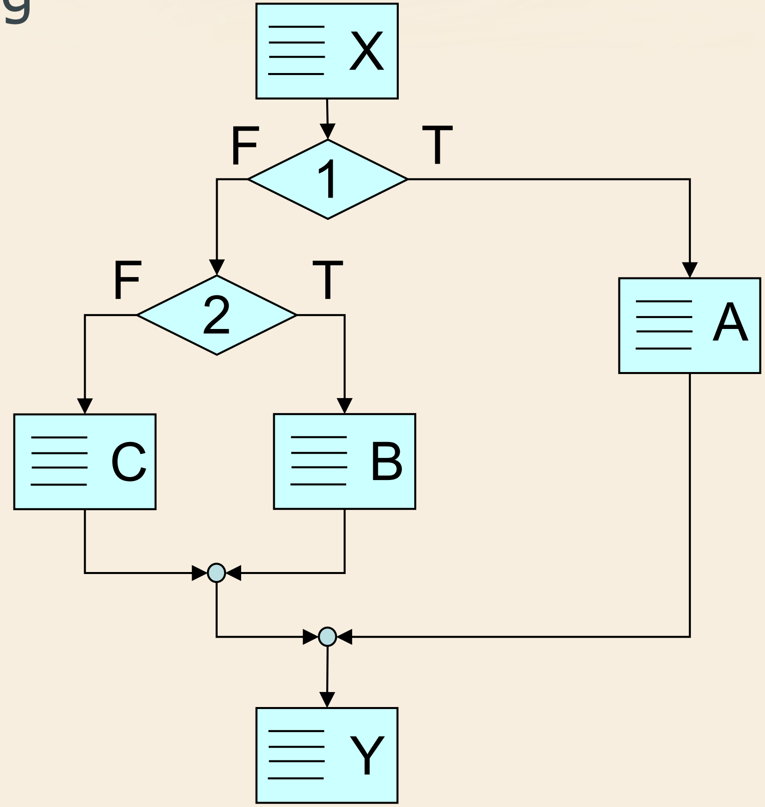


```
<?php
  $a = 2;
  $b = 3;
  if ($a == $b) {
    echo "\$a is equal to \$b , they are $a.\n";
    } else {
    echo “\$a is different from \$b. \n\$a equals \”$a\” while \$b equals \”$b\”.\n”;
  }
?>
```

The IF .. ELSEIF construct

- Allows to choose among many options

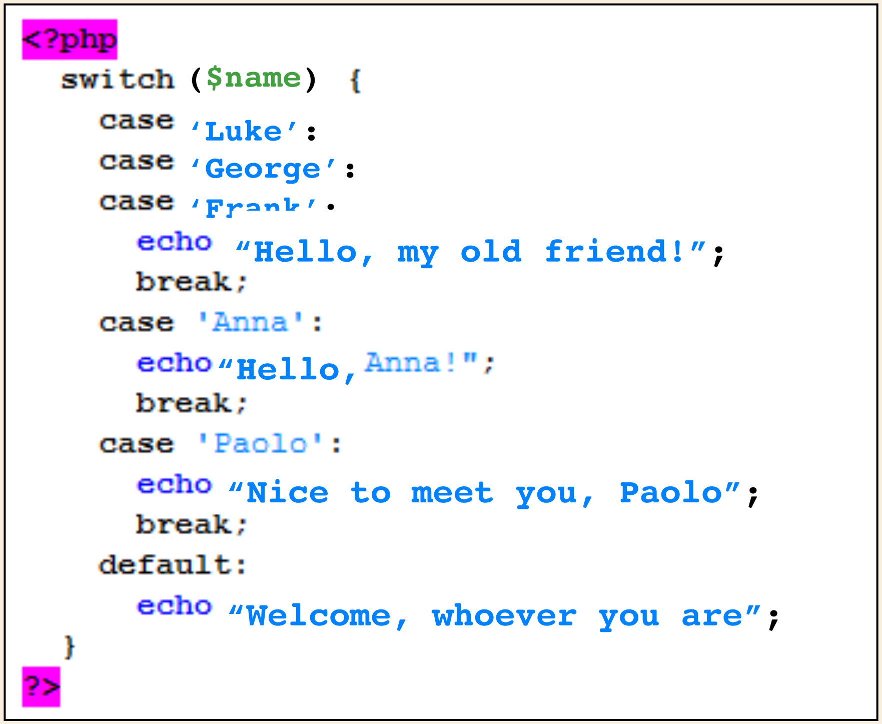
```
<?php
  $a = 2;
  $b = 3;
  if ($a == $b) {
    echo "\$a is equal to\$b.\n";
  } elseif ($a > $b) {
    echo "\$a is greater than\$b.\n";
  } else {
    echo "\$a is less than\$b.\n";
  }
?>
```



DBMG

The switch construct

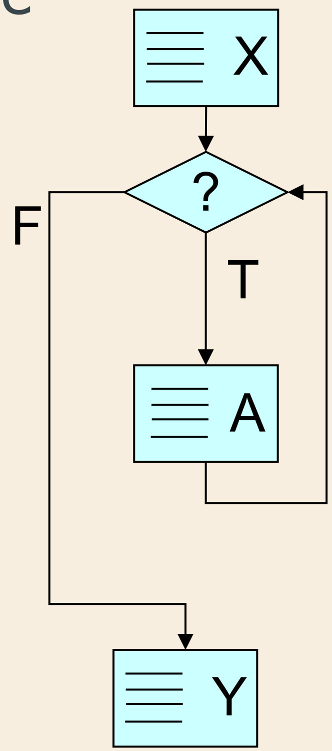
- Allows to predict different possible values for an expression and to execute specific code according to the value
  - Replaces a series of IF
  - break: forces the exit from the switch block
  - default: is optional



```
<?php
  switch ($name)  {
    case 'Luke':
    case 'George':
    case 'Frank':
      echo "Hello, my old friend!";
      break;
    case 'Anna':
      echo "Hello, Anna!";
      break;
    case 'Paolo':
      echo "Nice to meet you, Paolo";
      break;
    default:
      echo "Welcome, whoever you are";
  }
?>
```

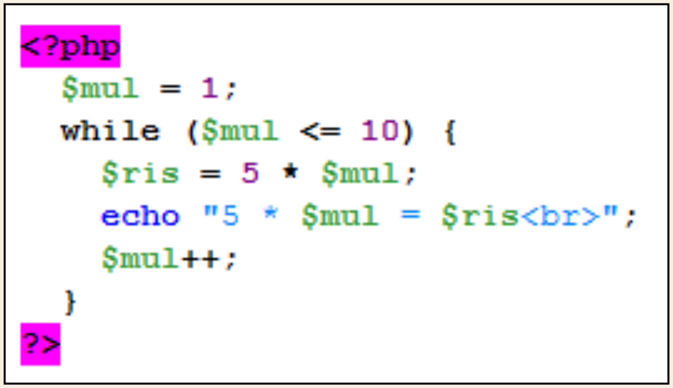
The while loop

- The block of instructions inside the while gets executed until the condition stays true
  - It's possible that the cycle is never executed, in case the condition is false from the beginning
  - In general the block of instructions modifies the parameters used in the condition

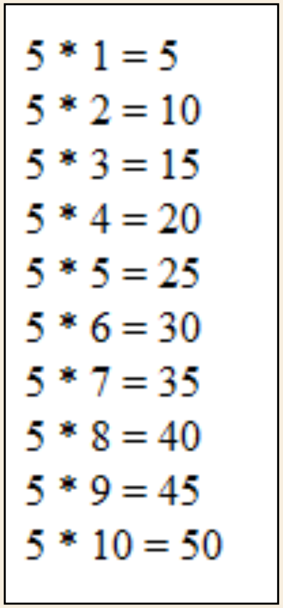


The while loop

- The block of instructions inside the while gets executed until the condition stays true



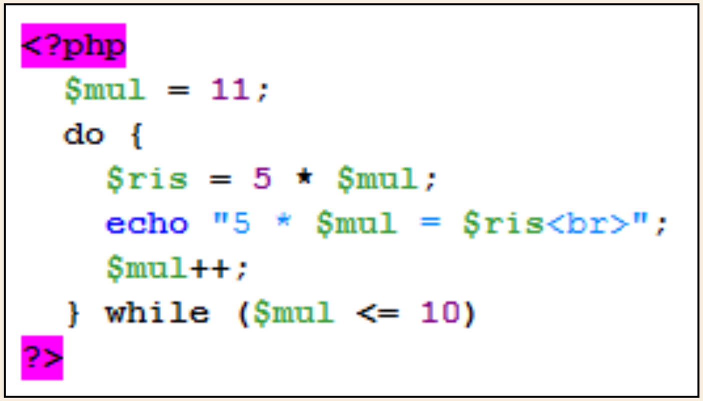
```
<?php
  $mul = 1;
  while ($mul <= 10) {
    $ris = 5 * $mul;
    echo "5 * $mul = $ris<br>";
    $mul++;
  }
?>
```



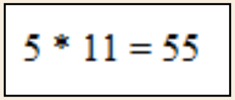
```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

The do .. while cycle

- Similar to the while, but it guarantees that the block of instructions is executed at least once
  - The condition is checked after the execution of the block of instructions



```
<?php
  $mul = 11;
  do {
    $ris = 5 * $mul;
    echo "5 * $mul = $ris<br>";
    $mul++;
  } while ($mul <= 10)
?>
```

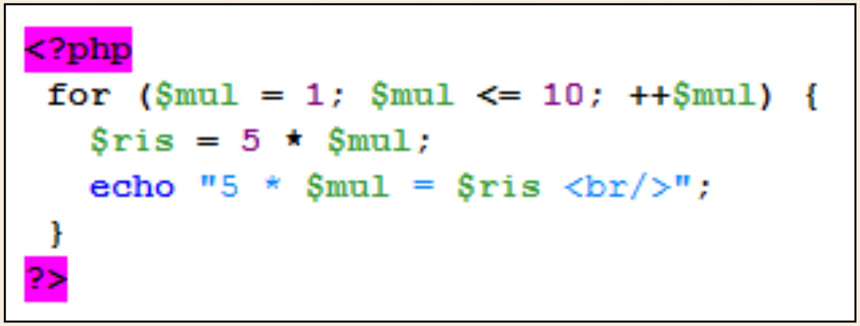


```
5 * 11 = 55
```

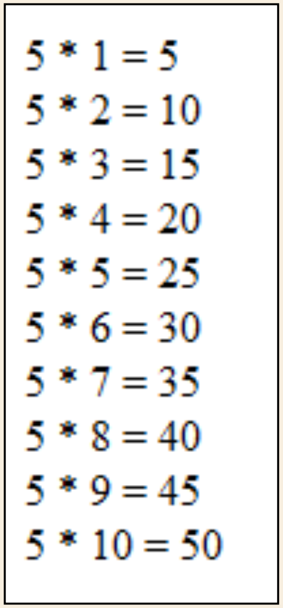
The for cycle

- Allows to repeat a block of instructions directly defining
  - The instructions of inizialization, executed only once upon entering the cycle
  - The condition, that, must be true to execute the block of instructions
  - The update, executed at the end of iteration
- Can always be rewritten as a while loop

The for cycle



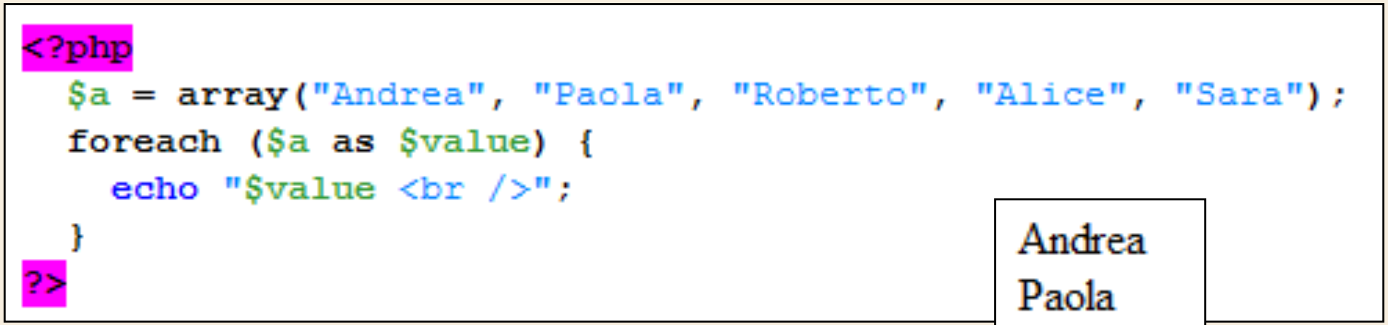
```php
<?php
 for ($mul = 1; $mul <= 10; ++$mul) {
   $ris = 5 * $mul;
   echo "5 * $mul = $ris <br/>";
 }
?>
```



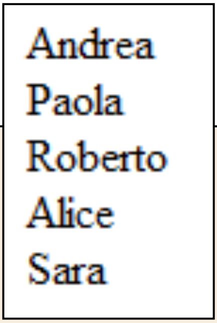
```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

The foreach cycle

- Cycle created to simplify access to arrays
  - Equivalent to a for cycle on the elements of an array



```php
<?php
  $a = array("Andrea", "Paola", "Roberto", "Alice", "Sara");
  foreach ($a as $value) {
    echo "$value <br />";
  }
?>
```



```
Andrea
Paola
Roberto
Alice
Sara
```

The foreach cycle on associative arrays

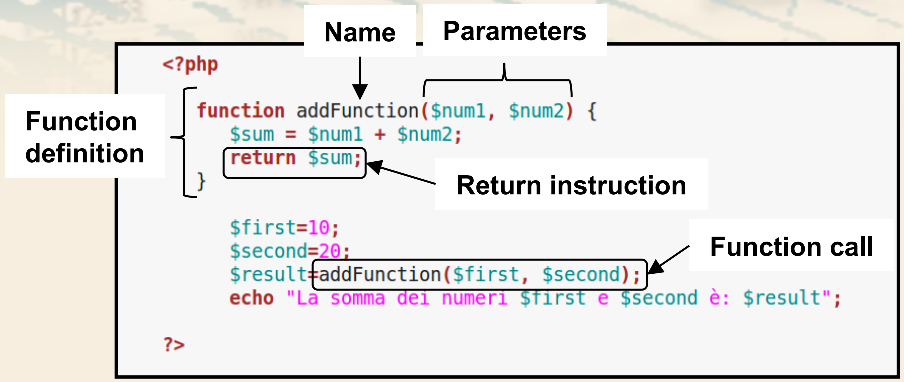
```
<?php
  $anno = array( “January” => 31,
                 “February” => 28,
                 "March” => 31,
                 “April” => 30,
                 “May” => 31,
                 “June” => 30,
                 “July” => 31,
                 “August” => 31,
                 “September” => 30,
                 “October” => 31,
                 “November” => 30,
                 “December” => 31);

  foreach ($year as $month => $days) {
    echo  “$month has $days days. <br />";
  }
?>
```

```
January has 31 days.
February has 28 days.
March has 31 days.
April has 30 days.
May has 31 days.
June has 30 days.
July has 31 days.
August has 31 days.
September has 30 days.
October has 31 days.
November has 30 days.
December has 31 days.
```

User defined functions

- C like syntax
- The name of the function is case insensitive
- The parameter list is optional and they are separated by comma
  - The parameters can be variables or values
- The return value is optional and it is specified with the keyword *return*
- The function can be used only after being defined and processed by the PHP interpreter

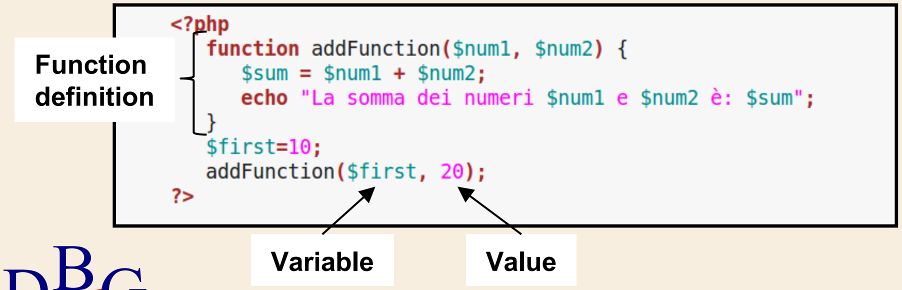
User defined functions



```
<?php

function addFunction($num1, $num2) {
    $sum = $num1 + $num2;
    return $sum;
}

    $first=10;
    $second=20;
    $result=addFunction($first, $second);
    echo "La somma dei numeri $first e $second è: $result";

?>
```

```
La somma dei numeri 10 e 20 è: 30
```

Example without the return instruction



```
<?php
function addFunction($num1, $num2) {
    $sum = $num1 + $num2;
    echo "La somma dei numeri $num1 e $num2 è: $sum";
}
$first=10;
addFunction($first, 20);
?>
```

```
La somma dei numeri 10 e 20 è: 30
```

Scoping

- The **scope** defines the region where a variable is visible and valid
  - Variables defined inside a function
    - Local scope: they are valid only in the function in which they have been declared
    - The arguments of the function have local scope too
  - Variables defined outside a function
    - Global scope: they are valid and visible by the whole script **but not inside the functions**
  - Global variables inside the functions
    - To access a global variable inside a function use the **global** qualificator

Example

```
<html>
   <body>
      <?php
         $acc = 0;
         function sum($x) {
            global $acc;
            $acc += $x;
         }
         sum(10);
         echo "La somma è: $acc";
         sum(10);
         echo "La somma è: $acc";
      ?>
   </body>
</html>
```

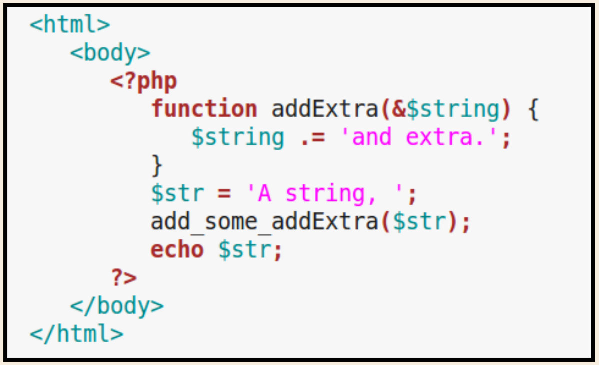
```
La somma è: 10
La somma è: 20
```

«Superglobal» variables

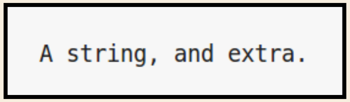
- Predefined global variables
  - They are visible and accessible everywhere
  - They are associative arrays
  - Typically used for environment information
- Examples:
  - **$GLOBALS** references all variables available in global scope
  - **$_GET** contains all the variables submitted to the script via HTTP GET
  - **$_POST** contains all the variables submitted to the script via HTTP POST

Passing parameters

- Parameters can be passed by **value** or by **reference**
- By default parameters are passed by **value**
- To pass parameters by **reference** they should be preceded by **&**



```
<html>
   <body>
      <?php
         function addExtra(&$string) {
            $string .= 'and extra.';
         }
         $str = 'A string, ';
         add_some_addExtra($str);
         echo $str;
      ?>
   </body>
</html>
```



```
A string, and extra.
```

Return «by reference»

- A function can return a result by value or by reference
- By default the return is by value
  - If the name of the of the function is preceded by & the return is by reference

```
<html>
   <body>
      <?php
         function &get_x_ref() {
            static $x = 30;
            return $x;
         }
         $y = &get_x_ref();
         echo $y;
      ?>
   </body>
</html>
```

$y is an alias of $x

```
30
```

Arguments with default value

- Each argument of a function can have a default value
- The default value should be defined with an assignment during the function declaration
  - If, during the function call, an argument with a default value is not specified then the default value is used.

```
<html>
   <body>
      <?php
         function conc($a, $b, $sep=', '){
            return ($a.$sep.$b."<br>");
         }
         echo conc("First", "Second");
         echo conc("First", "Second", '; ');
      ?>
   </body>
</html>
```

```
First, Second
First; Second
```

Script termination

- exit() and die() functions output a message and terminate the current script
  - They accept a string or an integer as parameters
  - The string is printed before the script ends
- Example:

```
exit("connection failed");
```

PHP and HTML forms

```
<form name= “userData”  action=“reponsePage.php”   method="GET">
    Elementi di input
</form>
```

- “form” tag with some attributes
  - Name: form name
  - Action: name of the program that will elaborate form data
  - Method: how parameters will be passed from the form to the program (can be "GET" or "POST")
- Inside the form there are many input elements

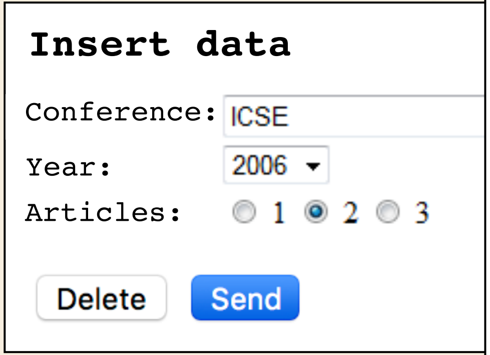
Access to form data

- The destination PHP script accesses to user inserted values through some special variables called "superglobal": the associative arrays $_GET, $_POST and $_REQUEST
  - "Superglobal" variables are accessible even inside some potential functions
- GET method
  - Values inside the query string are stored in the associative array $_GET
  - Each parameter of the form becomes a field of the associative array $_GET

Access to form data

- POST method
  - Each parameter of the form becomes a field of the associative array $_POST
- The associative array $_REQUEST contains $_GET, $_POST and $_COOKIE
  - Even if it’s not the same thing, in practice it can be used with any method, in alternative to $_GET or $_POST

Example: GET method



Insert data

Conference: ICSE

Year: 2006

Articles: 1 [✓ 2] 3

```
<form method="get" action="test.php">
  <table>
    <tr>
      <td> Conference: </td>
      <td> <input type="text" name="conf" size="20"> </td>
    </tr>
    <tr>
      <td> Year: </td>
      <td>
        <select name="year">
          <option value="2005">2005</option>
          <option value="2006">2006</option>
        </select>
      </td>
    </tr>
    <tr>
      <td>    Articles:    </td>
      <td>
        <input type="radio" name="number" value="1"> 1
        <input type="radio" name="number" value="2" checked> 2
        <input type="radio" name="number" value="3"> 3
      </td>
    </tr>
  </table>
  <br />
  <input type="reset" value= "Delete" >
  <input type="submit" value="Send" >
</form>
```

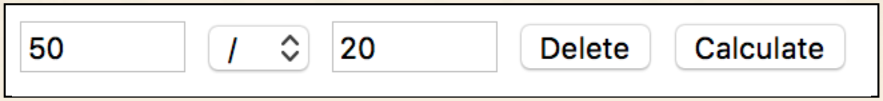
Example: GET method

- File test.php

```
<html>
<head>
<title> Result </title>
</head>

<body>

  <?php
    $conf = $_GET["conf"];
    $anno = $_GET["year"];
    $num = $_GET["number" ];

     echo "In the year $year you presented $num articles to the";
     echo "$conf conference.";
  ?>

</body>
</html>
```

```
In the year 2006 you presented 2 articles to the ICSE conference.
```

Example: calculator

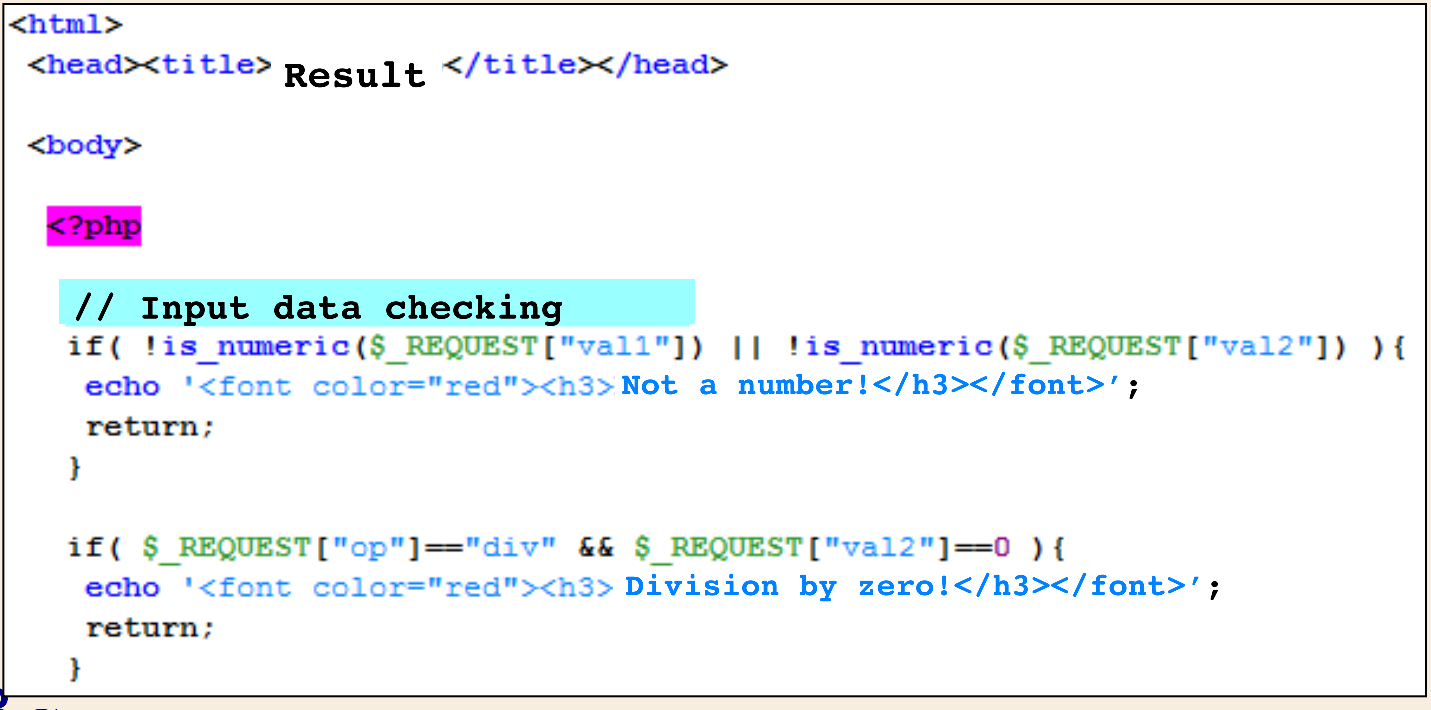


Result: 2.5

```
<html>
 <head>
  <title> Calculator</title>
 </head>
 <body>
  <form method="get" action="calculator.php">
   <input type="text" size="8" maxlength="8" name="val1" value="1">
   <select name="op">
    <option value="sum">+</option>
    <option value="sub">-</option>
    <option value="mul">*</option>
    <option value="div">/</option>
   </select>
   <input type="text" size="8" maxlength="8" name="val2" value="1">
   <input type="reset" value="Delete">
   <input type="submit" value="Calculate">
  </form>
 </body>
</html>
```

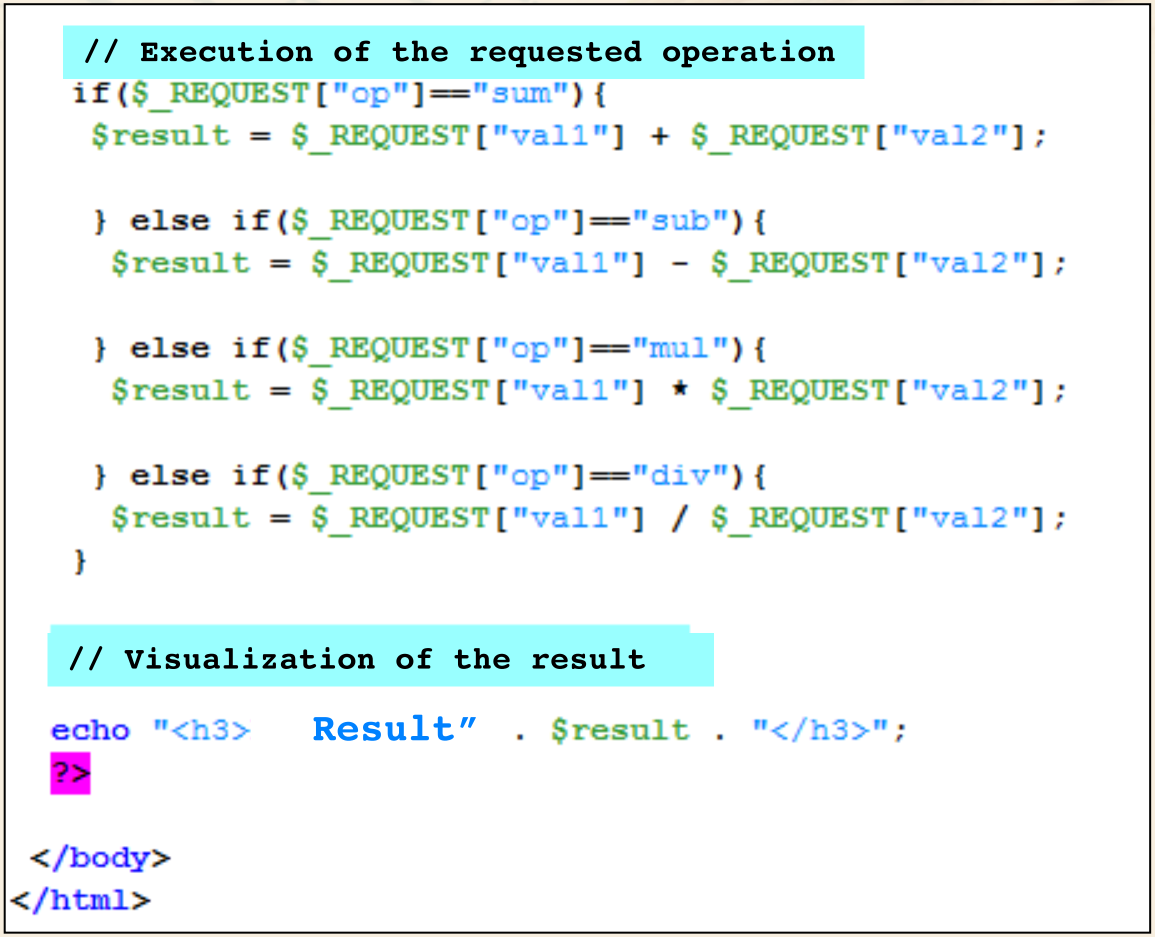
Example: calculator

- File calculator.php



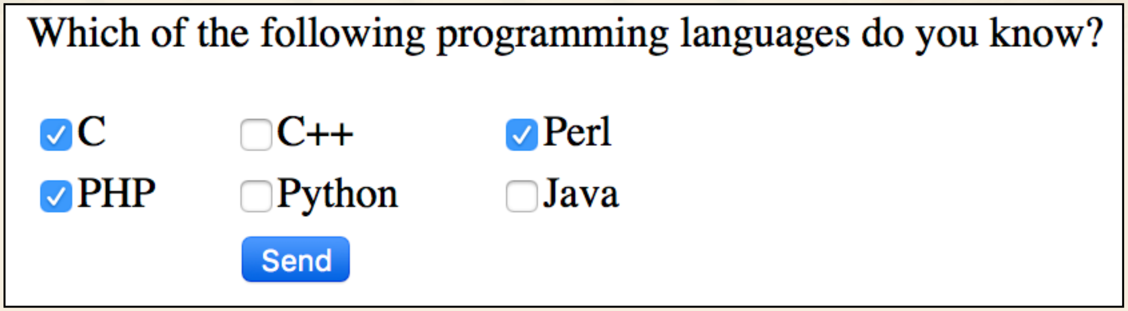
```
<html>
 <head><title> Result </title></head>

 <body>

  <?php

   // Input data checking
   if( !is_numeric($_REQUEST["val1"]) || !is_numeric($_REQUEST["val2"]) ){
    echo '<font color="red"><h3>Not a number!</h3></font>';
    return;
   }

   if( $_REQUEST["op"]=="div" && $_REQUEST["val2"]==0 ){
    echo '<font color="red"><h3> Division by zero!</h3></font>';
    return;
   }
```

Example: calculator

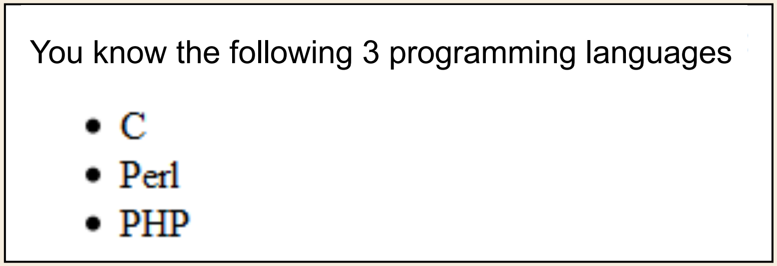


```
  // Execution of the requested operation
  if($_REQUEST["op"]=="sum"){
   $result = $_REQUEST["val1"] + $_REQUEST["val2"];

   } else if($_REQUEST["op"]=="sub"){
    $result = $_REQUEST["val1"] - $_REQUEST["val2"];

   } else if($_REQUEST["op"]=="mul"){
    $result = $_REQUEST["val1"] * $_REQUEST["val2"];

   } else if($_REQUEST["op"]=="div"){
    $result = $_REQUEST["val1"] / $_REQUEST["val2"];
  }

 // Visualization of the result

 echo "<h3>   Result" . $result . "</h3>";
 ?>

</body>
</html>
```

Example: multiple choice


Which of the following programming languages do you know?

[✓ C]	C++	[✓ Perl]
[✓ PHP]	Python	Java


You know the following 3 programming languages

- C
- Perl
- PHP

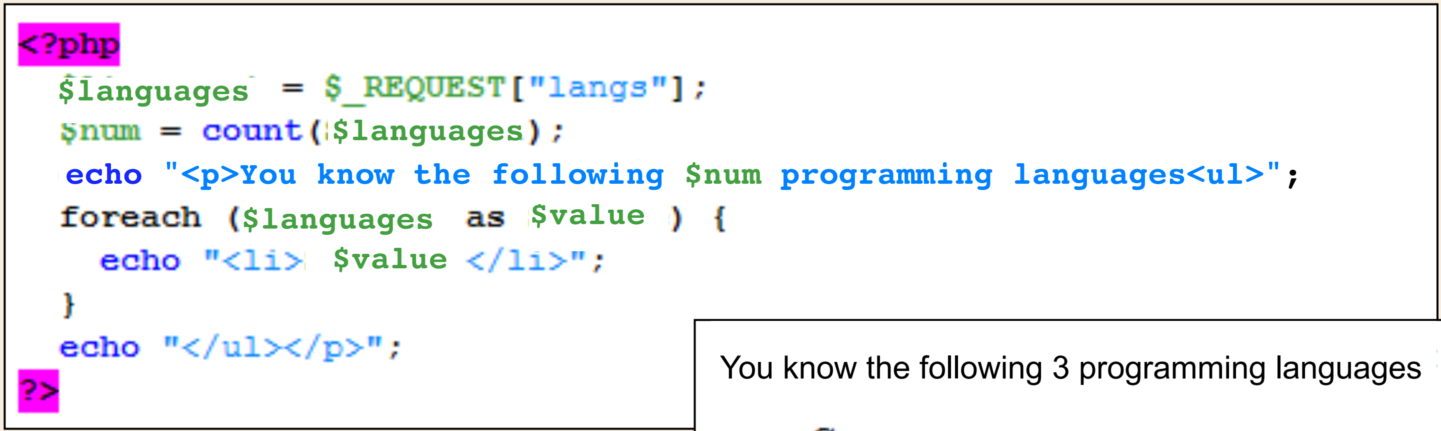
Example: multiple choice

- HTML form
  - Uses the langs[] array instead of 6 variables

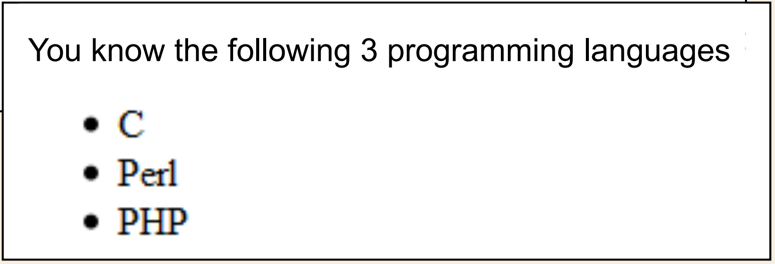
```html
<p>Which of the following programming languages do you know?</p>
<form action="select.php" method="post">
  <table width="250">
    <tr>
      <td><input type="checkbox" name="langs[]" value="C">C</td>
      <td><input type="checkbox" name="langs[]" value="C++">C++</td>
      <td><input type="checkbox" name="langs[]" value="Perl">Perl</td>
    </tr>
    <tr>
      <td><input type="checkbox" name="langs[]" value="PHP">PHP</td>
      <td><input type="checkbox" name="langs[]" value="Python">Python</td>
      <td><input type="checkbox" name="langs[]" value="Java">Java</td>
    </tr>
    <tr>
      <td></td>
      <td><input type ="submit" value="Send" ></td>
      <td></td>
    </tr>
  </table>
</form>
```

Example: multiple choice

- PHP script
  - The $_REQUEST ["langs"] array contains all selected values (in this case C, Perl and PHP)



```
<?php
  $languages  = $_REQUEST["langs"];
  $num = count($languages);
  echo "<p>You know the following $num programming languages<ul>";
  foreach ($languages as $value ) {
    echo "<li> $value </li>";
  }
  echo "</ul></p>";
?>
```



You know the following 3 programming languages

- C
- Perl
- PHP

Check submitted values

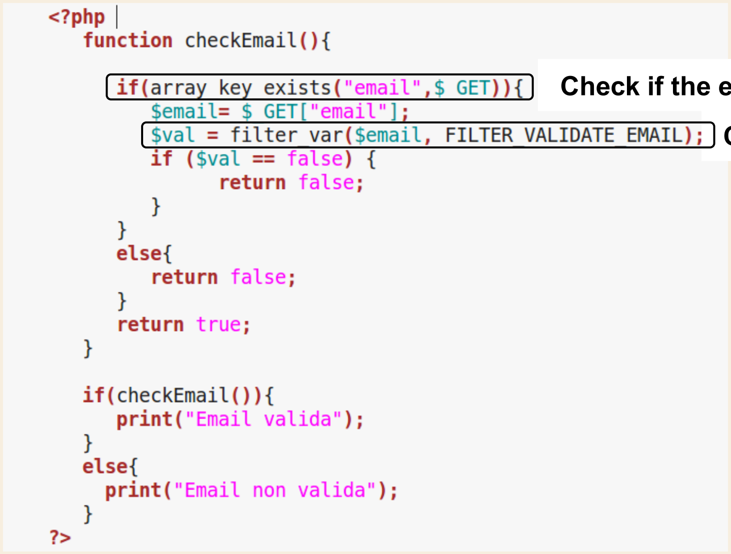
- It is **very important to validate** data received by users
  - To avoid the processing of erroneous data
    - E.g., email address erroneously formatted, insert of an unexpected value
  - To avoid hackers attacks
    - E.g., SQL injection

Data validation

- The **filter_var()** function can be used to validate different kind of data.
  - FILTER_VALIDATE_INT
  - FILTER_VALIDATE_FLOAT
  - FILTER_VALIDATE_BOOLEAN
  - FILTER_VALIDATE_EMAIL
- If the provided value is correct it returns true, otherwise false
- Moreover it is possible to check if a value assert specifics constraints (E.g., minimum age)

Data validation

- Check the correctness of the email address inserted by the user



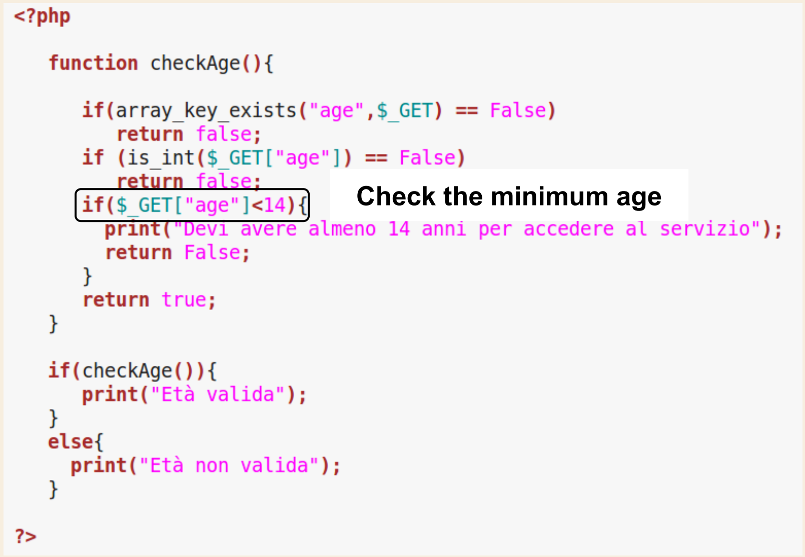
```
<?php
   function checkEmail(){

      if(array_key_exists("email",$_GET)){
         $email= $_GET["email"];
         $val = filter_var($email, FILTER_VALIDATE_EMAIL);
         if ($val == false) {
               return false;
         }
      }
      else{
         return false;
      }
      return true;
   }

   if(checkEmail()){
      print("Email valida");
   }
   else{
     print("Email non valida");
   }
?>
```

Check if the email field has been provided

Check the correctness of the email

Data validation

- Check that the user is at least 14 years old



```php
<?php

   function checkAge(){

      if(array_key_exists("age",$_GET) == False)
         return false;
      if (is_int($_GET["age"]) == False)
         return false;
      if($_GET["age"]<14){
        print("Devi avere almeno 14 anni per accedere al servizio");
        return False;
      }
      return true;
   }

   if(checkAge()){
      print("Età valida");
   }
   else{
     print("Età non valida");
   }

?>
```

Check the minimum age