# PHP Essentials 1

# Why PHP?

- PHP is a server-side scripting language, easy to learn and use.
- popular with developers who want to build web applications. (e.g. WordPress, Laravel, Symfony, drupal, magento, etc.)
- php stands for Hypertext Preprocessor, builds dynamic web pages on the server.
- server will execute the code and send the result back to the client.

# Installing php xampp

- we need to install php and mysql on the computer.
- we could do all that independently or we can use xampp.
- xampp is a free php development environment.
- localhost is acting as a server
- change port number

# First PHP pages

- create a folder 'myPage' in htdocs, create a php file 'index.php' in the folder.
- <?php echo "Hello World"; ?>
- we can use echo to print out a string.
- to print out a variable, we use echo.
- all the php code goes inside the php tags `<?php ?>`
- example of php showing and updating time

```php
<?php
$time = date("h:i:sa");
echo 'hello'$time;
?>
```

# embedding PHP in html

- we can use php in html by using the `<?php ?>` tags.
- all php codes must end with `;` (semicolon)
- you can add php inside html
- example of php in h1 title

```
<h1>
  <?php
    $time = date("h:i:sa");
    echo $time;
  ?>
</h1>
```

# variables in php

- variables are used to store data, that can be used later.
- in php we use `$` to declare variables.
- some rules for naming variables
  - you can use letters, numbers, and underscores.
  - example of variable name using snake case: `$my_variable`
  - example of variable name using camel case: `$myVariable`
- string is stored as a sequence of characters inside single or double quotes. eg: `$myString = "Hello World";`
- numbers are stored as integers or floats. eg: `$myNumber = 10;`
- booleans are stored as true or false. eg: `$myBoolean = true;`

# overwriting variables

- if we want to change the value of a variable, we can use `=` to assign a new value to the variable.
- example of overwriting variable

```
<?php $myNumber = 10; $myNumber = 20; ?>
```

- you can also declare constants, which are variables that cannot be changed eg: `define('MY_CONSTANT', 10);`

# the String DataType

- we can join strings using the `.` operator. eg: `$myString = $myName . $myAge . "some text";`
- we can directly call string variable in double eg: `echo "hello this is $myString";`
  - this wont work for single quotes.
- escaping characters in php
  - we can use `\` to escape characters. eg: `echo "some \" text \"";`
  - or use a single quote to escape characters. eg: `echo 'some "text"';`
- getting some char from a string
  - get the first character of a string: `$myString[0];`

# String functions

- we can use the following functions to manipulate strings:
  - `strlen()`: returns the length of a string.eg: `echo strlen("hello");`
  - `strtoupper()`: converts a string to uppercase. eg: `echo strtoupper("hello");`
  - `strtolower()`: converts a string to lowercase. eg: `echo strtolower("HELLO");`
  - `strpos()`: returns the position of the first occurrence of a substring. eg `echo strpos("hello", "e");`
  - `substr()`: returns a part of a string. eg: `echo substr("hello", 1, 3);`
  - `str_replace()`: replaces a string with another string. eg: `echo str_replace("hello", "world", "hello world");`
  - `str_repeat()`: repeats a string. eg: `echo str_repeat("hello", 3);`
  - `str_shuffle()`: shuffles a string. eg: `echo str_shuffle("hello");`
  - `str_split()`: splits a string into an array. eg: `$myArray = str_split("hello");`
  - `str_word_count()`: returns the number of words in a string. eg: `echo str_word_count("hello world");`

# the Number DataType

- `int`: represents an integer. eg: `$myNumber = 10;`
- `float`: represents a floating point number. eg: `$myNumber = 10.5;`
- order of operations: `()` `**` `*` `/` `+` `-` `%` BIDMAS: brackets, division, multiplication, addition, subtraction, modulus
- basic operation on numbers

```
// addition
$myNumber = 10 + 20;
// subtraction
$myNumber = 10 - 20;
// multiplication
$myNumber = 10 * 20;
// division
$myNumber = 10 / 20;
// modulus
$myNumber = 10 % 20;
// exponentiation
$myNumber = 10 ** 20;
// increment
$myNumber++;
```

# Number functions

- `abs()`: returns the absolute value of a number. eg: `echo abs(-10);`
- `ceil()`: rounds up a number. eg: `echo ceil(10.5);`
- `floor()`: rounds down a number. eg: `echo floor(10.5);`
- `round()`: rounds a number. eg: `echo round(10.5);`
- `rand()`: returns a random number. eg: `echo rand();`
- `sqrt()`: returns the square root of a number. eg: `echo sqrt(10);`
- `pi()`: returns the value of pi. eg: `echo pi();`
- `pow()`: returns the value of a number to the power of another number. eg: `echo pow(2, 3);`
- `min()`: returns the lowest value in an array. eg: `echo min([1, 2, 3, 4, 5]);`
- `max()`: returns the highest value in an array. eg: `echo max([1, 2, 3, 4, 5]);`

# The Array dataType

- arrays are a collection of values. eg indexed array:
  `$myArray = [1, 2, 3, 4, 5];`
- `array` : represents an array. eg: `$myArray = [1, 2, 3, 4, 5];`
- in php array can are indexed. eg: `$myArray[0];`
- in industry php arrays are used to store data in a list. eg:
  `$myArray = [1, 2, 3, 4, 5];`
- to print the array we use `print_r()` function. eg:
  `print_r($myArray);`
- we can change the value of an array at index 2 to a new value. eg:
  `$myArray[2] = "hello";`

# associative arrays

- associative arrays are arrays that have keys and values. eg:
  `$myArray = ["key" => "value"];`
- `array_key_exists()`: checks if an array has a key. eg:
  `array_key_exists("key", $myArray);`
- adding a new key to an array. eg: `$myArray["key"] = "value";`
- overwriting a key in an array. eg: `$myArray["key"] = "value";`

# multi dimension array

- multi dimension arrays are arrays that have arrays inside them. eg: `$myArray = [["key" => "value"], ["key" => "value"]];`
- indexed multi dimension arrays are arrays that have indexed arrays inside them. eg: `$myArray = [[0, 1, 2], [0, 1, 2]];`

# Array functions

- `count()`: returns the number of elements in an array. eg: `echo count($myArray);`
- `array_key_exists()`: returns true if the given key exists in the array. eg: `echo array_key_exists(0, $myArray);`
- `array_keys()`: returns an array containing the keys of the array. eg: `echo array_keys($myArray);`
- `array_values()`: returns an array containing the values of the array. eg: `echo array_values($myArray);`
- `array_merge()`: merges two or more arrays. eg: `$myArray = array_merge($myArray, $myOtherArray);`
- `array_pop()`: removes the last element of an array. eg: `array_pop($myArray);`
- `array_shift()`: removes the first element of an array. eg: `array_shift($myArray);`
- `array_unshift()`: adds an element to the beginning of an array. eg: `array_unshift($myArray, 10);`
- `array_push()`: adds an element to the end of an array. eg: `array_push($myArray, 10)

# Include and Require

- code once and use it multiple times
- `include` : includes a file. eg: `include "myFile.php";`
- include will stop the script if the file is not found, it will not throw an error. it will carry on with rest of the script.
- `require` : includes a file. eg: `require "myFile.php";`
- require will stop the script if the file is not found, it will throw an error. it will not carry on with rest of the script.
- `include_once` : includes a file only once if it is not already included. eg: `include_once "myFile.php";`

# Control Structures and Loops

# Loops in php

- loops are used to execute a block of code a number of times. eg:
- when we want to execute a block of code 5 times `for ($i = 0; $i < 5; $i++) {}`
- Imagine if we have a list of names and we want to print them all. we can use a loop to do this. a for loop in php `for ($i = 0; $i < count($myArray); $i++) {echo $myArray[$i];}}`
- foreach -> loops through an array. eg: `foreach ($myArray as $value) {echo $value;}`
- while -> loops while a condition is true. eg: `while ($i < 10) {echo $i; $i++;}`

```php
// for loop
for ($i = 0; $i < 5; $i++) {
    echo $i;
}
//for each loop
foreach ($myArray as $value) {
    echo $value;
}
// while loop
while ($i < 10) {
    echo $i;
    $i++;
}
// do while loop
do {
    echo $i;
    $i++;
} while ($i < 10);
```

# looping though an array in html

```
<?php
$users =[ ['name'=>"John", 'age'=>"30"],
          ['name'=>"Jane", 'age'=>"25"],
          ['name'=>"Bob", 'age'=>"40"]];
foreach($users as $user) {>
  <div>
    <h2> <?php echo user['name'] ?></h2>
    <p> <?php echo user['age'] ?></p>
  </div>
<?php} ?>
```

# boolean dataType

- boolean data type is used to store true or false values. eg: `$myBoolean = true;`
- this is because the values are converts into strings. eg: `echo true;` will give `1` and `echo false;` wont output anything
- some comparison operation that return boolean
  - `==`: checks if two values are equal. eg: `if ($myBoolean == true) {echo "true";}`
  - `===`: checks if two values are equal and of the same type. eg: `if ($myBoolean === true) {echo "true";}`
  - `!=`: checks if two values are not equal. eg: `if ($myBoolean != true) {echo "true";}`
  - `!==`: checks if two values are not equal and of the same type. eg: `if ($myBoolean !== true) {echo "true";}`
  - `>`: checks if the first value is greater than the second value. eg: `if ($myBoolean > true) {echo "true";}`
  - `<`: checks if the first value is less than the second value. eg: `if ($myBoolean < true) {echo "true";}`
  - `>=`: checks if the first value is greater than or equal to the second value. eg: `if ($myBoolean >= true) {echo "true";}`
  - `!`: negates a boolean value. eg: `if (!$myBoolean) {echo "true";}`
  - `&&`: checks if both values are true. eg: `if ($myBoolean && true) {echo "true";}`
  - `||`: checks if either value is true. eg: `if ($myBoolean || true) {echo "true";}`

# conditional statements using `if`

- a major part of programming is conditional statements. eg:
  `if ($myBoolean) {echo "true";}`
- example of conditional statement in real life is if you are driving and you are not allowed to drive. eg: `if ($myAge>18) {echo "true";} else {echo "false";}`
- example of nested conditional in real life is if only people with age above 18 and below 25 can drive. eg:
  ```
  if ($myAge>18) {if ($myAge<25) {echo "true";} else {echo "false";}} else {echo "false";}
  ```
- using `&&` `if ($myAge>18 && $myAge<25) {echo "true";} else {echo "false";}`
- using `||` `if ($myAge>18 || $myAge<25) {echo "true";} else {echo "false";}`

# break and continue

- break: breaks out of the loop. eg: `for ($i = 0; $i < 5; $i++) {if ($i == 3) {break;}}` this will stop the loop at the 3rd iteration.
- continue: continues to the next iteration of the loop. eg: `for ($i = 0; $i < 5; $i++) {if ($i == 3) {continue;}}` this will skip the 3rd iteration.

# functions

- functions are just block of code that can be used to perform a specific task and can be called anywhere eg: `function myFunction() {echo "hello";}`
- something to note is that functions are not hold in memory. they are just a block of code that can be called. eg: `myFunction();`
- we can pass arguments to functions. eg:
  `function myFunction($myArgument) {echo $myArgument;}`

```
function contactCard($name, $age, $location) {
  echo "<div class='card'>
    <div class='card-body'>
      <h5 class='card-title'>$name</h5>
      <p class='card-text'>Age: $age</p>
      <p class='card-text'>Location: $location</p>
    </div>";
}
contactCard("John", "30", "London");
```

# using function with associative array

```php
$users =[ ['name'=>"John", 'age'=>"30", 'location'=>"London", 'hobbies'=>["reading", "writing"]],
  ['name'=>"Jane", 'age'=>"25", 'location'=>"Paris", 'hobbies'=>["playing", "writing"]],
  ['name'=>"Bob", 'age'=>"40", 'location'=>"New York", 'hobbies'=>["walking", "writing"]]];

function contactCard2($user) {
  echo "<div class='card'>
    <div class='card-body'>
      <h5 class='card-title'>{$user['name']}</h5>
      <p class='card-text'>Age:{ $user['age']}</p>
      <p class='card-text'>Location:{ $user['location']}</p>
      <p class='card-text'>Hobbies: ";
      foreach ($user['hobbies'] as $hobby) {
        echo $hobby . " ";
      };
      echo "</p>
    </div>";
}
```

# variable scope in php

- scope refers to the visibility of variables. eg: `$myVariable` is global and can be used anywhere in the script.
- a variable can be local to a function or global to the script. eg of local variable: `function myFunction() {$myVariable = "hello";}` this variable cannot be used outside of the function.