summray\sumBroCode.php

```

<!-- php tag -->
    <?php
    ?>

<!-- display a message  -->
    <?php
    echo "your text";
    ?>
<!-- break line  -->
    <?php
    echo "text<br>"
    ?>

<!-- comments  -->
<?php
    // one line
    # one line

    /*
    multiline

    */
    ?>

<!-- variables  -->
    <?php
    $variableName='value';
    ?>

 <!-- data types -->
    PHP is a loosely typed language
    /*
    integer
    float
    string
    boolean
    array
    object
    null
    */

<!-- constant  -->
    <!--
    if you wanna to use constant var you  don't need to add $ :
     -->
    <?php define("VariableName",value); ?>


<!-- variable functions -->
    <?php
    var_dump($var);  # check data type  with more details :
    isset($var);     #Checks if a variable is set and is not null.
```

```php
    unset($var);     # Destroys a variable.
    empty($ar);      # Checks if a variable is empty.
    gettype($var);   # Returns the data type of a variable.
    settype($var,"newType")
    ?>

<!-- global keyword  -->
    <?php
    $name = "ayoub";

    function testing() {
    global $name; // Access the global variable $name
    echo $name;
    }

    testing(); // This will correctly output "ayoub"
    ?>

<!-- variable data type conversion : -->  --------------------
    <!--
        implicit Type Conversion: PHP will automatically convert data types when necessary.
    -->
<!-- Type Casting: -->
    <?php
    (int) or (integer)  # Casts a value to an integer data type.
    (float) or (double) # Casts a value to a floating-point (decimal) number.
    (string) # Casts a value to a string.
    (array  # Casts a value to an array.
    (object) # Casts a value to an object.
    (bool) or (boolean) # Casts a value to a boolean (true or false).
    ?>

<!-- Explicit Type Conversion Functions: -->
    <?php
    intval($str);   #Converts a value to an integer.
    floatval($str); # or doubleval(): Converts a value to a floating-point number.
    strval($nbr);   #Converts a value to a string.
    boolval($str);  #Converts a value to a boolean.
    ?>

<!-- concat string with variables  -->  ------------------------
        <?php
        echo "text $variableName";
        echo "text {$variableName}";
        echo "text" .$variableName . "text";
        ?>
    <!-- example  -->
        <?php
        $name="bro Code";
        $food="pizza";
        $email="ayoubmajid71@gmail.com<br>";
        $age=21;
        $users=2;
        $quantity=3;
        $gpa=2.5;
        $prise=4.99;
```

```
110
111         $forSale=true;
112         echo "hello $name<br>";
113         echo "you like $food<br>";
114         echo "you email is  $email<br>";
115         echo "you are $age years old<br>";
116         echo "there are $users users online<br>";
117         echo "you would like to by $quantity items ";
118         echo "your  gpa is $gpa <br>";
119         echo "your pizza is  \"$$prise\" <br>";
120         echo "For sale status : $forSale <br>";
121         $total=$quantity* $prise ;
122
123         echo "the total is : $$total <br>";
124         ?>
125
126 <!-- escape character  -->  ----------------------------
127     In PHP, an escape character is a backslash (\)
128     that is used to indicate that the character following it
129     should be treated specially. Escape characters are used to
130     represent characters that have a special meaning in PHP,
131     such as quotation marks, newline characters, and others
132     <!--example   -->
133         <?php
134         echo "hello world \"ayoub \"<br>";
135         ?>
136
137 <!-- arithmetic operators  --> ----------------
138     <?php
139     //  arithmetic operators :
140         // + - * / ** %
141
142     // operator precedence :
143         // ()
144         // **
145         //* /  %
146         // + -
147     ?>
148
149 <!-- increment operators  --> ---------------------------
150     <?php
151     // increment operators :
152     // postincrement :
153         $variableName++;
154
155     // preincrement :
156         ++$variableName;
157     ?>
158
159 <!-- decrement operators  -->  ------------------
160     <?php
161     // decrement operators :
162     // postdecrement  :
163         $variableName--;
164
165     // predecrement  :
```

```php
        --$variableName;
    ?>

<!-- $_GET, $_POST --> ------------------------------------
     // $_GET, $_POST = special variables used to collect data from an HTML form
    //data is sent to the file in the action attribute of <form>
    <form action="some_file.php" method="get">

    $_Get = Data is appended to the url
            NOT SECURE
            char limit
            Bookmark is possible w/ values
            GET requests can be cached
            Better for a search page

    //$ POST = Data is packaged inside the body of the HTTP request
            MORE SECURE
            No data limit
            Cannot bookmark
            GET requests are not cached

    <!-- get info   when the form   submitted   -->  -------------------------
        <?php
        $_GET["inputNameAttribute"]; /* it will return the value of input
        when the form submitted */

        $_POST["inputNameAttribute"]; /* it will return the value
        of input when the form submitted*/
        ?>

    <!-- example  -->
        <form action="index.php" method="GET">
            <input type="text" name="userName" placeholder="enter your name">
            <input type="submit">
        </form>
<?php
echo $_GET["userName"];
?>

<!-- THE difference  between get and post mode  -->  ------------------

        <!--
            get  :  send info as a query params
            post : send info as a body params

            using :
            get insensible  information : (small )
            post sensible information like password user  token ( maybe  big data )
        -->
    <!-- example  -->
        <form action="index.php" method="post">
            <input type="number"  placeholder="enter the quantity" name="quantity">
            <input type="submit" value="log in ">
        </form>
        <?php
        $item="pizza";
```

```
        $price=5.99;
        $quantity=$_POST['quantity'];
        $total=$quantity*$price;
        echo "you have ordered $quantity *$price <br>";
        echo "the total is $$total <br>";
        // get  user info :
        ?>

<!-- functions  --> ------------------------------
    <!-- get max of  a range of values  -->
        <?php max(val1,val2,valeN); ?>

    <!-- get min fo a range of values  -->
        <?php min(val1,val2,valN); ?>

    <!-- get random   number  -->
        <?php rand(min,max); ?>

    <!-- get the sqrt of number  -->
        <?php sqrt(val); ?>

    <!-- get the ceil value of a number  -->
        <?php ceil(val); ?>

    <!-- get the floor value of a number  -->
        <?php floor(val); ?>

    <!-- get the round of a number   -->
        <?php round(val); ?>

    <!-- identify number of decimal after the comma  -->
        <?php round(number,numberAfterComma); ?>

    <!-- get the pi value  -->
        <?php pi(); ?>

    <!-- application  -->
        <form action="index.php" method="post">
            <label for="r">radius</label>
            <input type="text"  placeholder="enter the radius " name="r" id="r"><br>
            <input type="submit" value="calculate">
        </form><br>
        <?php
        // $x=$_POST['x'];
        // $y=$_POST['y'];
        $radius=$_POST['r'];

        // circumference :  2PR
        // aria          : PR**2
        // volume        : (4/3 )PR^3

        $circumference=round(2*pi()*$radius,2);
        $aria=round(pi()*$radius**2,2);
        $volume = round((4/3)*pi() *$radius**3,2);
        echo "the circumference is :  {$circumference}cm <br>" ;
        echo "the aria is :  {$aria}cm^2 <br>";
```

```
        echo "the volume is :  {$volume}cm^3 <br>" ;
        ?>

<!-- if statement  -->  ------------------------------
        if  some condition is true do something
        if condition is not true don't do it
        <?php
        if(condition){
        // do some statements
        }
        else if (condition){
        // do some statements
        }
        else{
            // do some statements  :
        }
        ?>

    <!-- example  1 -->
        <?php
        $age = 99;
        if($age>=100){
        echo "you're to old to enter this site : ";
        }
        else if ($age >= 18) {
            echo "You may enter this site <br>";
        } elseif ($age <= 0) {
            echo "That was not a valid age ";
        } else {
            echo "you must be 18+ to enter ";
        }
        ?>
    <!-- example 2  -->
        <?php
        $hours = 50;
        $rate = 15;
        $weeklyPay = 0;
        if ($hours < 0) {
            $weeklyPay = 0;
        } else if ($hours <= 40) {
            $weeklyPay = $hours * $rate;
        } else {
            $weeklyPay = ($rate * 40) + ($hours - 40) * ($rate * 1.5);
        }
        echo "you made $$weeklyPay this week<br>";

        ?>

<!-- relational operators
    > grater than
    < lower than
    >= grater than or equal
    <= lower than or equal
    !=  differ than
    !== differ than + (differ type also )
    ==  => equal to
```

```
    === => equal to + (the same data type)
    -->

<!--  logical operator
    &&  => (and)
    || => (or )
    !  => (not)
    -->
<!-- example  1 -->
    <?php
    $age = 25;
    $citizen = true;
    if ($age >= 18 && $citizen) {
        echo "You can vote <br>";
    } else {
        echo "You can not vote <br>";
    }
    ?>
<!-- example 2  -->
    <?php
    $child = false;
    $senior = false;
    $ticket = null;
    if ($child || $senior) {
        $ticket = 10;
    } else {
        $ticket = 15;
    }
    echo "the ticket prise is $$ticket<br>";
    ?>

<!-- switch -->   ----------------------------
        replacement to using many elseif statements more efficient
        <?php
        switch($variableName){

        case val1:
            #code
            break;

        case val2:
            #code
            break;

        case val3:
            #code
            break;
            //....

        default :
            #code
        }
        ?>
    <!-- example   -->
        <?php
        $grade = "A";
```

```
        switch ($grade) {
            case 'A':
                echo  "You did great";
                break;

            case 'B':
                echo  "You did good";
                break;

            case 'C':
                echo  "You did okay";
                break;

            case 'D':
                echo  "You did poorly";
                break;
            case 'F':
                echo  "You failed";
                break;
            default:
                echo "invalid grade ";
        }
        ?>

<!-- For loop -->  ------------------------------------------------------------------
        repeat some code a certain # of times

        <?php
        for ($i=start ; condition ; step){
            //code
        }
        ?>
    <!-- example  1-->
        <?php
        for ($i = 1; $i <= 10; $i++) {
            echo "$i-Hello <br>";
        }
        ?>
    <!-- example 2 -->
        <form action="index.php" method="post">
                <label for="counter">enter a number to count to</label><br>
                <input type="number" name="counter" m><br>
                <input type="text" placeholder="the name will be counted " name="name"><br>
                <input type="submit" value="counter">
            </form><br>

        <?php
        $counter = 0;
        $name = "";
        if (isset($_POST['counter']) && isset($_POST['name'])) {
            $counter = $_POST['counter'];
            if ($counter < 0) $counter = 0;
            $name = $_POST['name'];
        }
        for ($i = 1; $i <= $counter; $i++) {
            echo "$i -$name <br>";
```

```
        }
        ?>

    <!-- while condition :  -->
        <?php
        while(condition){
        //code;
        }
        ?>
    <!-- example -->
        <form action="index.php" method="post">
                <input type="submit" value="stop">
            </form><br>
        <?php
        $seconds = 0;
        $running = true;
        while ($running) {
            $seconds++;
            if (isset($_POST['stop'])) break;
            echo "the current amount of seconds $seconds <br>\n";
        }
        ?>

    <!-- do while  -->
        <?php
        do {
        // code
        }while(condition);
        ?>

    <!-- foreach  -->
        <?php
        foreach( $elements as $item){
        // code
        };
        ?>
    <!-- example  -->
        <?php
        foreach($family as $item){
        echo $item . "<br>";

        }
        ?>

<!-- array in php  --> ---------------------------------------
        pointer that point at sequence of memory cases
    <?php
        $arrayName=array(value1,value2,value3,/*...*/,valueN);
        ?>
        <!-- or  -->
        <?php
        $arrayName=[value1,value2,valueN];
        ?>

    <!-- access to arrays  element  -->
        <?php
```

```
        $arrayName[index]=value;
        ?>

    <!-- display array using foreach  -->
        <?php
        $foods = array("apple", "orange", "banana", "coconut");
        foreach ($foods as $food) {
            echo  $food . "<br>";
        }
        #or
        print_r($foods);
        ?>

    <!-- get length of array  -->
        count($arrayName);

    <!-- push element the the end on an array  -->
        <?php
        array_push($arrayName,"value");
        #or
        $arrayName[]="value";
        ?>

    <!-- pop element from array  -->
        <?php
        array_pop($arrayName);
        ?>

    <!-- shift left array  -->
        <?php array_shift($arrayName); ?>

    <!-- get reversed version from your array  -->
        <?php  $reversedArray= array_reverse($arrayName);  ?>

    <!-- check if an element in array  -->
        <?php in_array("value", $arrayName)?>

    <!-- searching with key  -->
        <?php array_key_exists("key", $arrayName) ?>

    <!-- convert array to string  -->
        <?php $string = implode("separator", $arrayName);  ?>

<!-- associative Array    ------------------------------------------
    provide you to use associative names rather than indexes :
    -->
        <?php $arrayName=["assName1"=>Value1,"assName2"=>value2]; ?>
        <!-- you can using index just with elements that does not has an
        associative name
        (index did not start form first element in array => start from first not
        associative item)
    -->
    <!-- example 1 -->
        <?php
        $capitals = [
            "USA" => "Washington",
```

```
            "JAPAN" => "kyoto",
            "SOUTH KOREA" => "Seoul",
            "MOROCCO" => "Rabat"
        ];
        foreach ($capitals as $key  => $value) {
            echo $key . "=$value " . "<br>";
        }

        $capitals["Algeria"] = "Algeria";
        foreach ($capitals as $key  => $value) {
            echo $key . "=$value " . "<br>";
        }
        ?>
    <!-- example 2 -->
        <?php
        $studentName = "ayoub";
        $score = [
            "ayoub" =>
            [
                "score" => "19",
                "grade" => "A"
            ],
            "adam" =>
            [
                "score" => "15",
                "grade" => "B"
            ],
            "youness" =>
            [
                "score" => "14",
                "grade" => "C"
            ]
        ];
        echo  "mark  :" . $score[$studentName]["score"] . "<br>";
        echo  "score : ", $score[$studentName]["grade"];
        ?>

<!-- get array keys  -->
    <?php $keys=array_keys($arrayName);  ?>

<!-- key array values  -->
    <?php $values=array_values($arrayName);  ?>

<!-- flip between key and values -->
    <?php $flippedArray=array_flip($arrayName);  ?>

<!-- example 3  -->
    <form action="index.php" method="post">
            <label for="">enter a country </label>
            <input type="text" name="capital">
            <input type="submit" value="submit">
    </form><br>
        <?php
    $capitals = [
        "USA" => "Washington",
        "JAPAN" => "kyoto",
```

```
        "SOUTH KOREA" => "Seoul",
        "MOROCCO" => "Rabat"
    ];
    if (isset($_POST['capital'])) {
        $capital = strtoupper($_POST['capital']);
        if (array_key_exists($capital, $capitals))
            echo "the capital is : " . $capitals[$capital] . "<br>";
        else {
            echo "this capital not found ";
        }
    }
    ?>

<!-- isset($variableName)  -->
    returns true if variable is declared and not null

<!-- empty($variableName)  -->
    return true if a variable is not declared || false || null || ""\

<!-- unset($variableName)  -->
    removes the reference to the array,
    and the variable $myArray will no longer be defined

<!-- create an array when setting the name of inputs    -->
    <input type="text" name="name[]">

<!-- example   1 -->
    <form action="index.php" method="post">
        <label for="">username :</label>
        <input type="text" name="username"><br>
        <label for="">password :</label>
        <input type="text" name="password"><br>
        <input type="submit" value="Log in" name="login">
    </form><br>
    <?php
    foreach ($_POST as $key => $value) {
        echo "$key = $value <br>";
    }
    if (isset($_POST["login"])) {

        $username = $_POST["username"];
        $password = $_POST["password"];
        if (empty($username)) {
            echo "Username is missing <br>";
        } else if (empty($password)) {
            echo "Password is missing <br>";
        } else {
            echo  "\"hello\" $username <br>";
            echo  "\"hello\" $password <br>";
        }
    }
    ?>
<!-- example 2  -->
    <form action="index.php" method="post">

    <input type="radio" name="creditCard" value="Visa" id="Visa" required>
```

```
    <label for="Visa">Visa</label><br>

    <input type="radio" name="creditCard" value="Mastercard" id="Mastercard" required>
    <label for="Mastercard">Mastercard</label><br>

    <input type="radio" name="creditCard" value="American Express" id="American Express" required>
    <label for="American Express">American Express</label><br><br>
    <hr><input type="submit" name="submit">

    </form><br>
    <?php

    if (isset($_POST['submit'])) {

        $userCard = $_POST['creditCard'];

        echo "You selected  :  $userCard <br>";
    } else {
    }

    ?>
<!-- functions  -->  ----------------------------------------
write some code once, reuse when you need it
        <?php
        function functionName(){
        // code

        }
        ?>
    <!-- example 1 -->
        <?php
        function happyBirthday($name, $age)
        {
            echo "happy Birthday dear $name! <br>";
            echo "happy Birthday dear $name! <br>";
            echo "happy Birthday dear $name! <br><br>";
            echo "you are $age years old <br>";
        }
        happyBirthday("ayoub", 20);
        happyBirthday("amine", 25);
        happyBirthday("amal", 18);
        ?>
    <!-- example 2 -->
        <?php
        function isEven(int $number){

        return $number%2 ? "is even" : "is odd";
        }
        echo isEven(5);
        ?>
    <!-- example 2 with strict typing  -->
        PHP 7 introduced scalar type declarations and return type declarations.
        By using strict typing, you can enforce that only values of the
        correct type are allowed. This can be done by declaring the types
        for function arguments and return values explicitly:
```

```php
        <?php
        declare(strict_types=1);
        function isEven(int $number){
            return $number%2 ? "is yes" : "is odd";
            }
            echo isEven('5'); // it will give strict typing error :
    ?>

    <!-- return multiple values  -->
        <?php
        function functionName(){

        return [val1,val2,valN];
        }
        ?>

    <!-- get multiple table from functions  -->
        <?php
        list($var1,$var2,$varN)=functionName();
        #or
        [$var1,$var2,$varN]=functionName();
        ?>

    <!--example -->
        <?php
        function getMultipleValues() {
            $value1 = "Hello";
            $value2 = "World";
            $value3 = 42;
            // Return an array with multiple values
            return array($value1, $value2, $value3);
        }
        // Call the function and receive multiple values
            list($result1, $result2, $result3) = getMultipleValues();

        // Output the results
            echo $result1 . " " . $result2 . " " . $result3;
        ?>

<!-- example 3  -->
    <form action="index.php" method="post" onsubmit=" return checkCheckedFood(); ">
        <input type="checkbox" name="Pizza" value="Pizza" id="Pizza">
        <label for="Pizza">Pizza</label><br>
        <input type="checkbox" name="Hamburger" value="Hamburger" id="Hamburger">
        <label for="Hamburger">Hamburger</label><br>
        <input type="checkbox" name="Hotdog" value="Hotdog" id="Hotdog">
        <label for="Hotdog">Hotdog</label><br><br>
        <input type="checkbox" name="Taco" value="Taco" id="Taco">
        <label for="Taco">Taco</label><br><br>
        <hr><input type="submit" name="submit">
    </form><br>
    <script>
        function checkCheckedFood() {
            let arrFoods = document.querySelectorAll(" form > input[type='checkbox'] ");
            arrFoods = Array.from(arrFoods);
            let isChecked = false;
```

```
            arrFoods.forEach(element => {
                if (element.checked) return isChecked = true;
            });
            if (!isChecked) alert("please select at least one food");
            return isChecked;
        }
    </script>
    <?php
    function getListOfFoodSelected()
    {
        $strFoods = "";
        foreach ($_POST as $key => $value) {
            if ($key != "submit") $strFoods .= "  " . $value;
        }
        return $strFoods;
    }
    if (isset($_POST['submit'])) {
        $strFoods = getListOfFoodSelected();
        echo "You like  : $strFoods";
    }
    ?>

<!-- static variable  -->
<?php static $variableName=value; ?>
    <!-- example  -->
        <?php
        function testing(){
        static $number2=10;
        echo $number2++ . "\n";
        }
        ?>
<!-- string functions  --> ------------------------------------------
    <!-- convert to  lower case  -->
        <?php strtolower($string);  ?>

    <!-- convert to upper case -->
        <?php strtoupper($string); ?>

    <!-- know the length of a variable  -->
        <?php
        strlen($string);
        ?>

    <!-- get specific  index -->
        <?php
        $string[index];
        ?>

    <!-- assign a value of an index  -->
        <?php
        $string[index]=newValue;
        ?>

    <!-- replace string  -->
        <?php
        $str=str_replace(replaceSubStr,newSubStr,$string);
```

```
        ?>

    <!-- sub string  -->
        <?php
        substr($string,startIndex,length) // default length=strlen($string)
        ?>

    <!-- get length of str -->
        <?php strlen($str);  ?>

    <!-- count number of words  -->
        <?php str_word_count($str); ?>

    <!-- reverse a string  -->
        <?php strrev($str); ?>

    <!-- get position of a substr -->
        <?php strpos($str,"searchedSubStr"); ?>

    <!-- replace empty character  -->
        <?php  str_pad($str,length,"replaceValue") ?>

    <!-- shuffle string  خلط -->
        <?php str_shuffle($str) ?>

    <!-- compare between two string  -->
        <?php strcmp($str) ?>

    <!-- str to array  -->
        <?php $arrStr=explode("separator",$str);?>

    <!-- array to str -->
        <?php $Str=implode("separator",$arrStr);?>

    <!-- filter input  -->  -------------------------
    <?php
    // filter special character :
        filter_input(INPUT_POST,"name",FILTER_SANITIZE_SPECIAL_CHARS);

    // filter int :remove character  from right  and left
        filter_input(INPUT_POST,"name",FILTER_SANITIZE_NUMBER_INT);

    // filter float :
        filter_input(INPUT_POST,"name",FILTER_SANITIZE_NUMBER_FLOAT);

    // valid email :
        filter_input(INPUT_POST,"name",FILTER_VALIDATE_EMAIL);

    // accept just valid int
        filter_input(INPUT_POST,"name",FILTER_VALIDATE_INT);

    // accept just valid float :
        filter_input(INPUT_POST,"name",FILTER_VALIDATE_FLOAT);
?>
<!-- example -->
    <form action="index.php" method="post" >
```

```
        <input type="text" name="username" placeholder="enter your name ">
        <input type="text" name="age" placeholder="enter your age ">
        <input type="text" name="email" placeholder="enter your email ">
        <input type="submit" value="log in">
    </form><br>
    <?php
    if (isset($_POST['username']) && isset($_POST['age']) && isset($_POST['email'])) {
        $username = filter_input(INPUT_POST, "username", FILTER_SANITIZE_SPECIAL_CHARS);
        $age = filter_input(INPUT_POST, "age", FILTER_VALIDATE_FLOAT);
        $email = filter_input(INPUT_POST, "email", FILTER_VALIDATE_EMAIL);
        echo "Hello $username <br>";
        echo "your age is  $age <br>";
        echo "your age is  $email <br>";
    }
    ?>

<!-- include  -->
    copy the content of a file (php/html/txt) then includes it in your file
    <?php
    include "path";
    #or
    include("path");
    ?>
    <!-- or  -->
    <?php
    require "path"; // stop all rest of code
    ?>

<!-- cookie -->
    information about a user stored in a user's web-browser
    targeted advertisements,browsing preferences,
    other non-sensitive data
    it's a piece of information stored inside our system
    we should never set username and  password using cookies

    <!-- set a cookie -->
        setcookie("key","value",ExpireDay,filePath);
        // time() :return the current day
        // 86400 : one day

<!-- example  -->
    <?php
    setcookie("favoriteFood","pizza",time() +86400,"/");
    setcookie("favoriteDrink","coffe",time() +86400*2,"/");
    ?>

<!-- print all cookies -->
    <?php
    foreach ($_COOKIE as $key => $value) {
        echo "$key = $value <br>";
    }
    ?>

<!-- session --> ------------------------
    session =SGB used to store information on a user to be used
    across multiple pages a user is assigned a session-id
```

```
    <!-- start session  -->
        <?php
        session_start();
        ?>
    <!-- create new key in a session -->
        <?php
        $_SESSION['key']='value';
        ?>
    <!-- get key value  -->
        <?php
        echo  $_SESSION['key'];
        ?>
    <!-- get the current session id  -->
        <?php
        session_id();
        ?>
    <!-- set session timeout  -->
        The ini_set('session.gc_maxlifetime', minutes); line is used to set the
        maximum lifetime of a session. It determines how long the server
        should keep session data before it's considered expired.
        This setting works in conjunction with the session timeout
        mechanism. If a user is inactive for the specified number of minutes,
        their session may expire.
        <?php
        ini_set('session.gc_maxlifetime', minutes);
        ?>
    <!-- regenerate session id : -->
        The session_regenerate_id() function is used to generate a new session ID
        and replace the current session ID with the new one.
        This is often done as a security measure to prevent session
        fixation attacks. A session fixation attack occurs
        when an attacker sets a user's session ID to a known value,
        allowing them to hijack the user's session.
    <!-- destroy session -->
        <?php
        session_destroy();
        ?>
<!-- change to another page  -->
    <?php
    header("Location:pageName");
    ?>

<!-- example login page --> ------------------------

<!-- index.php -->
    <?php
    session_start();

    if(isset($_POST['login'])){
        if(!empty($_POST['username']) && !empty($_POST['password'])) {
            $_SESSION["username"] = $_POST['username'];
            $_SESSION["password"] = $_POST['password'];
            header("location: home.php");
            exit();
        } else {
```

```
            echo "Missing username or password";
        }
    }
    ?>
    <form action="index.php" method="post">
            <input type="text" placeholder="username"  name="username"><br>
            <input type="password" placeholder="password"  name="password"><br>
            <input type="submit" value="login"  name="login"><br>
        </form>

<!-- home.php -->
    <?php
    session_start();

    if(!isset($_SESSION['username']) || !isset($_SESSION['password'])){
        header("location: index.php");
        exit();
    }
    ?>
    This is the home page.
        <form action="home.php" method="post">
            <input type="submit" value="log out" name="logout">
        </form>

        <?php
    if(isset($_POST['logout'])){
        session_destroy();
        header("location: index.php");
        exit();
    }
    ?>

<!-- example 2   Shopping Cart Implementation  -->
    <?php
    session_start();

    // Add item to the cart
    if (!isset($_SESSION['cart'])) {
        $_SESSION['cart'] = array();
    }
    $item = array('id' => 1, 'name' => 'Product A', 'price' => 19.99);
    $_SESSION['cart'][] = $item;

    // Display the shopping cart
    foreach ($_SESSION['cart'] as $item) {
        echo $item['name'] . " - $" . $item['price'] . "<br>";
    }
    ?>

<!-- date --> ------------------------------
    <!-- y/m/d -->
        echo date("y/m/d");
        echo date("y-m-d");

    <!-- print the current day name -->
        echo date("l");
```

```
1061
1062     <!-- h:i:sa -->
1063         echo date("h:i:sa");
1064         <!--
1065         Day:
1066
1067         d: Day of the month, two digits with leading zeros (01 to 31).
1068         j: Day of the month without leading zeros (1 to 31).
1069         Month:
1070
1071         m: Numeric representation of the month, two digits with leading
1072         zeros (01 to 12).
1073         n: Numeric representation of the month without leading zeros
1074         (1 to 12).
1075         M: A short textual representation of a month
1076         (e.g., Jan, Feb).
1077
1078         Year:
1079
1080         Y: Four-digit representation of the year (e.g., 2023).
1081         y: Two-digit representation of the year (e.g., 23).
1082         Time:
1083
1084         H: 24-hour format of an hour with leading zeros (00 to 23).
1085         h: 12-hour format of an hour with leading zeros (01 to 12).
1086         i: Minutes with leading zeros (00 to 59).
1087         s: Seconds with leading zeros (00 to 59).
1088         a: Lowercase Ante meridiem (am) or Post meridiem (pm).
1089         Day of the Week:
1090
1091         D: A short textual representation of a day (e.g., Mon, Tue).
1092         l: A full textual representation of the day of the week
1093         (e.g., Monday, Tuesday).
1094         Timezone:
1095
1096         e: Timezone identifier (e.g., UTC, GMT).
1097         -->
1098
1099 <!-- server  -->  --------------------------
1100     =SGB that contains headers,paths ,and script locations
1101     the entries in this array are created by the web server
1102     shows nearly everything you need to know about the current
1103     web pag
1104
1105 <!-- display server SGV -->
1106     <?php
1107     foreach($_SERVER as $key=>$value){
1108         echo "$key = $value<br>";
1109     }
1110
1111     ?>
1112
1113 <!-- php_self   -->
1114     RETURN THE CURRENT PATH
1115     <?php
1116     $_SERVER['PHP_SELF'];
```

```
    ?>
<!-- REQUEST METHOD -->
    return the current request method (get is the default)
    <?php
    $_SERVER['REQUEST_METHOD'];
    ?>
    <!-- control + click  see the code source-->

<!--  hashing --> ------------------
    hashing : transforming sensitive data (password)
    into letters ,numbers ,and/or symbols
    via a mathematical process (similar to encryption)
    hides the original data from 3rd parties


    In PHP, a hash is a one-way function that takes an input (or 'message')
    and produces a fixed-size string of characters,
    which is typically a hexadecimal number. The purpose of hashing is
    to generate a unique identifier (hash value) for a given input.
    Here are some key aspects of hashing in PHP:

    Hash Functions:
    PHP provides several built-in hash functions that you can
    use for various purposes. Some commonly used hash functions include:

    md5(): Produces a 32-character hexadecimal number.
    sha1(): Produces a 40-character hexadecimal number.
    hash(): A flexible function that supports multiple algorithms (e.g., MD5, SHA-256, SHA-
512).

<!-- example  -->
    <?php
    $data = 'Hello, World!';
    $md5Hash = md5($data);
    $sha1Hash = sha1($data);
    $customHash = hash('sha256', $data);
    ?>

<!-- Password Hashing: --> ------------------------
    When dealing with passwords, it's essential to use secure hashing
    methods to protect user credentials. PHP provides the password_hash()
    function for this purpose. This function uses a strong, adaptive hashing
    algorithm and automatically handles the generation of a salt.
    Example:
<?php
    $password = 'user_password';
    $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
    ?>

<!--  hash a string  -->
    <?php
    $hash=password_hash($password,PASSWORD_DEFAULT);
    ?>

<!-- check if string ==hash version  -->
    <?php
```

```
    if(password_verify($password,$hash)){

        echo "you logged in ";
    }
    else{
    echo "incorrect password";

    }
    ?>

<!-- different between hashing and encryption -->
    Encryption and hashing are both cryptographic techniques,
    but they serve different purposes and have distinct characteristics:

    1. **Purpose:**
    - **Encryption:** The primary purpose of encryption is
        to protect data confidentiality. It involves transforming data
        (plaintext) into an unreadable format (ciphertext) using an algorithm
        and a secret key. The goal is to ensure that only authorized parties with
        the correct key can decrypt and access the original data.


        - **Hashing:** The primary purpose of hashing is to create a fixed-size,
        irreversible representation (hash value) of data.
        Hash functions are commonly used to verify data integrity,
            generate unique identifiers, and securely store passwords.

    2. **Reversibility:**
    - **Encryption:** Encryption is a reversible process
    . It transforms data in such a way that it can be decrypted back to its
    original form using the appropriate decryption key.
    - **Hashing:** Hashing is a one-way process.
    Once data is hashed, it cannot be feasibly reversed to obtain
        the original input. Hash functions are designed to be irreversible
        for security reasons.

    3. **Key Usage:**
    - **Encryption:** Encryption involves the use of keys
    for both encryption and decryption. There are symmetric
        key algorithms where the same key is used for both operations,
        and asymmetric key algorithms where different keys are used for
        encryption and decryption.
    - **Hashing:** Hashing typically does not involve the use of keys.
        A fixed-size hash value is generated based solely on the input
        data and the hashing algorithm.

    4. **Use Cases:**
    - **Encryption:** Used for securing communication,
        protecting sensitive data at rest (e.g., file encryption),
        and ensuring privacy.
    - **Hashing:** Used for data integrity verification,
    creating digital signatures, generating unique identifiers,
        and securely storing passwords.

    5. **Examples:**
    - **Encryption:** AES (Advanced Encryption Standard),
```

```
        RSA (Rivest-Shamir-Adleman), DES (Data Encryption Standard),
        and others.
    - **Hashing:** MD5, SHA-256, SHA-3, and others.

    6. **Security Considerations:**
    - **Encryption:** Involves managing and securing keys,
    and the security relies on the strength of the encryption
        algorithm and the secrecy of the key.
    - **Hashing:** Focuses on collision resistance
    (the likelihood of two different inputs producing the same hash)
    and pre-image resistance (the difficulty of finding an input that
        produces a specific hash).

    In summary, encryption is about keeping data confidential and
    reversible, while hashing is about creating fixed-size irreversible
    representations for various purposes like data integrity verification
    and password storage. Depending on the specific security requirements
    of a system, both encryption and hashing may be used in complementary ways.

<!-- my sql extension --> ------------------------------

<!-- connect to database -->
    <?php
    $dbServer = "localhost";
    $dbUser = "root";
    $dbPassword = "";
    $dbName = "test";
    $connection = "";

    $connection = mysqli_connect(
        $dbServer,
        $dbUser,
        $dbPassword,
        $dbName
    );
    ?>

<!-- check connection error  -->
    <?php
    $connection->connect_error();
    ?>

<!-- get result of a query -->
    <?php
    $query="sql code";
    $result = mysqli_query($connection, $query);
    ?>

<!-- count number of row from response -->
    <?php
    mysqli_num_rows($result)
    ?>

<!-- fetch response as an(associative Array) one row -->
    <?php
    $row=mysqli_fetch_assoc($result);
```

```php
    ?>

<!-- close a connection -->
    <?php
    $connection->close();
    // or
    mysqli_close($connection);
    ?>

<!-- example  1 (database.php) -->
    <?php
    $dbServer = "localhost";
    $dbUser = "root";
    $dbPassword = "";
    $dbName = "test";
    $connection = "";
    try {
        $connection = mysqli_connect(
            $dbServer,
            $dbUser,
            $dbPassword,
            $dbName
        );
        if ($connection)
            if ($connection->connect_error) {
                // If connection fails, throw a mysqli_sql_exception
                throw new mysqli_sql_exception("Connection failed: " . $connection->
connect_error);
            }
        echo "you connected with success";
    } catch (mysqli_sql_exception $e) {

        echo $e->getMessage();
    }
    ?>
<!-- example 2 :  -->
    <?php
    include "database.php";
    $sql="insert into employee values ('adam','doma1')";
    try{
        mysqli_query($connection,$sql);
        echo "user is now registered";
    }catch(mysqli_sql_exception $e){

    echo "DataBase Error : " .$e->getMessage();
    }
    ?>
<!-- example 3 -->
    <?php
    include "database.php";
    $username = "ayoub";
    $password = "youbista123";
    $hasPassword = password_hash($password, PASSWORD_DEFAULT);
    $sql = "insert into employee values ('$username','$hasPassword')";

    // exception handling :
```

```
        try {
            mysqli_query($connection, $sql);
            echo "user is now registered";
        } catch (mysqli_sql_exception $e) {

            echo "DataBase Error : " . $e->getMessage();
        }

        ?>
<!-- example 4  (fetch data) -->
    <?php
    include "database.php";

    $sql = "select * from  employee where username='amina'";

    try {

        $result=mysqli_query($connection, $sql);
        if(mysqli_num_rows($result) >0){

    $row=mysqli_fetch_assoc($result);
    echo "username : " .$row['username'] ."<br>";
    echo "password :" . $row['password'] ."<br>";
        }
    } catch (mysqli_sql_exception $e) {

        echo "DataBase Error : " . $e->getMessage();
    }

    ?>
<!-- fetch multiple lines  -->
    <?php
    include "database.php";
    $sql = "select * from  employee where username='ayoub'";
    try {
        $result = mysqli_query($connection, $sql);
        if (mysqli_num_rows($result) > 0) {
          while ($row = mysqli_fetch_assoc($result)) {
                echo "username : " . $row['username'] . "<br>";
                echo "password :" . $row['password'] . "<br><br>";
            }
        }
    } catch (mysqli_sql_exception $e) {

        echo "DataBase Error : " . $e->getMessage();
    }
    ?>

<!-- file  -->
    <!--
    add enctype=enctype="multipart/form-data" in the form
    add input with type file :
    -->

 <!-- get file by name -->
    <?php
```

```
    $_FILES['fileName'];
    ?>

 <!-- get file name  -->
    <?php
    $file['name'];
    ?>

 <!-- get file full path  -->
    <?php
    $file['tmp_name'];
    ?>

<!-- get file type  -->
    <?php
    $file['type'];
    ?>

<!-- get file size  -->
    <?php
    $file['size'];
    ?>

<!-- ensure return file name  -->
    <?php
    basename($file['name'])
    ?>

<!-- specify dir to store a file   -->
    <?php
    $targetPath= 'dirName/' . basename($file['name']);
    ?>

<!-- upload a file to a dir   -->
    <?php
    //$filePath  : $file['tmp_name'];
    move_uploaded_file($filePath,$targetPath);
    ?>
<!-- check if file is already exits  -->
    <?php
    file_exists($targetPath);
    ?>

 <!-- get file content  -->
    <?php
    file_get_contents($targetPath);
    ?>

<!-- escape html special character  -->
    <?php
    echo htmlspecialchars("content");
    ?>

<!-- example   -->
        <form action="index.php" method="post" enctype="multipart/form-data">
            <input type="file" name="file">
```

```
            <button type="submit" name="submit">Submit</button>
        </form>
        <?php
    if (isset($_POST['submit'])) {
        if (isset($_FILES['file'])) {
            $file = $_FILES['file'];
            $filePath = './uploaded/' . basename($file['name']);
            if (file_exists($filePath)) {
                echo "File already exists.";
            } else {
                echo  $file['tmp_name'] . "<br>";
                echo  basename($file['name']) . "<br>";
                if (move_uploaded_file($file['tmp_name'], $filePath)) {
                    echo "File uploaded successfully.";
                    $fileContent = file_get_contents($filePath);
                    echo "<pre>htmlspecialchars($fileContent)</pre>";
                } else {
                echo "Error uploading file. Check for errors: "
                . $file['error'];
                }
            }
        } else {
            echo "<br>No such file or directory.";
        }
    }
        ?>

<!-- PDO  --> ------------------------------------

<!-- create new connection  -->
    <?php
    // Database credentials
    $host = 'your_database_host';
    $dbname = 'your_database_name';
    $username = 'your_username';
    $password = 'your_password';

// PDO connection
    try {
        $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
        echo "Connected successfully";
    } catch (PDOException $e) {
        echo "Connection failed: " . $e->getMessage();
    }
    ?>

<!-- exit  script  -->
    <?php
    die("message ");
    ?>

<!-- new connection example 2  -->
    <?php

        $hostName = 'localhost';
        $dbName = 'test';
```

```
        $username = 'root';
        $password = '';

        try{

            $connection = new PDO(
                "mysql:host=$hostName;
            dbname=$dbName;
            charset=utf8",
                $username,
                $password
            );

        echo "connected with success <br>";
        }catch(Exception $e){

            die("Error : ". $e->getMessage());
        }
        ?>
<!-- activate the details error  -->
    <?php
    $connection->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

#OR ADD AS A PRAMS IN CONNECTION :
    $connection = new PDO(
        "mysql:host=$hostName;
    dbname=$dbName;
    charset=utf8",
        $username,
        $password,
        [PDO::ATTR_ERRMODEP =>PDO::ERRMODE_EXCEPTION]
    );
    ?>

<!-- prepare the request in PDoStatement object  -->
    <?php
    $query="query";
    $requestPdoObject=$connection->prepare($query);
    ?>

<!-- send the request to the database  -->
    <?php
    $requestPdoObject->execute();
    ?>

<!-- get the response from the db :  -->
    <?php

    // get the first row of the response :
    $queryResponse=$requestPdoObject->fetch('fethMode');

    //get all line in the response :
    $queryResponse=requestPdoObject->fetchAll();
    ?>

<!-- fetch modes  -->
```

```
    <!--
    Tableau associatif : PDO::FETCH_ASSOC
    Tableau indexé : PDO::FETCH_NUM
    Les deux à la fois : PDO::FETCH_BOTH (par défaut)
    Objet : PDO::FETCH_OBJ
    -->

<!-- markers  (placeholder)  -->
  <!--
    using to replace a value in a request to avoid the sql injection
    ! you can't set the table name as a marker (placeholder)
    -->
    <?php
    $query=' ...:markerKey1..makerKey2';

    $requestPdoObject->execute([
        'markerKey1'=>'value',
        'markerKey2'=>'value'

        ])
    ?>

<!-- ? the same to marker  -->
    <?php
    $query=' ...:?..?';

    $requestPdoObject->execute(['value1','value2']);
    ?>

    <!-- using bindPrams rather than passing array in execute  -->
        <?php
        $requestPdoObject->bindPrams(':makerKey1','value');
        $requestPdoObject->bindPrams(':makerKey2','value');

        ?>

 <!-- case matching :  -->
    preg_match is a PHP function that performs a regular expression match.
    It is used to check if a string matches a given pattern,
    specified by a regular expression. The function returns 1 if the pattern
    matches, 0 if it does not, or false if an error occurs.
    <?php
    preg_match(pattern, string);

//  patten :
    // check if a string is in the word is in string :
        preg_match('#str#', string);
    ?>

<!-- patterns  -->
    Regular Expressions (Regex):
    Regular expressions are a powerful tool for matching patterns
    in strings. They consist of a sequence of characters
    that define a search pattern. Regex patterns can include:

    ✨Literal Characters:  characters that match themselves.
```

```
                        Metacharacters: Special characters with a reserved
                        meaning, such as
                . (any character),
                * (zero or more occurrences),
                + (one or more occurrences),
                etc.
    ✨Character Classes: Specify a set of characters to match, like
                    [0-9] for any digit.

    ✨Quantifiers: Indicate the number of occurrences,
                such as * (zero or more),
                + (one or more),
                ? (zero or one).

    ✨Anchors: Define the position in the string, such as
            ^ (start of the line),
            $ (end of the line).

    ✨Groups and Capturing: Use parentheses to create groups,
                        and captured groups store matched substrings.

<!-- example  -->
   <?php
       $hostName = 'localhost';
       $dbName = 'test';
       $username = 'root';
       $password = '';
       try {
           $connection = new PDO(
               "mysql:host=$hostName;
           dbname=$dbName;
           charset=utf8",
               $username,
               $password
           );
           $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
           echo "connected with success <br>";
           $query = "select * from employee where  username= :username";
           $requestObject = $connection->prepare($query);
           // send the query the mysql :
           $requestObject->execute([

           'username'=>'ayoub'
           ]);

           // get the response from the  mysql :
           $response = $requestObject->fetchAll(PDO::FETCH_ASSOC);
           echo "<br>";
           print_r($response);
           foreach ($response as $key => $value) {
               echo  "<br> </h1>" . $key . "</h1> <br> ";

               foreach ($value as $SubKey => $value) {

                   echo $SubKey .  " = " . $value .  "<br>";
               }
```

```
            }
        } catch (Exception $e) {

            die("Error : " . $e->getMessage());
        }


        ?>

<!-- file handling ----------------- -->

<!-- read from file :  -->
    readfile("path");

<!-- open file  -->
    $fileName=fopen("path","mode");
    <!--
    mode :
    r: read mode
    w : write mode
    a : append mode
    -->

 <!-- read from opened file  -->
    echo fread($fileName,filesize("path"));

<!-- close file -->
    fclose($fileName);

<!-- read one line  -->
    fgets($fileName);

 <!-- read until you read end of file -->
    <?php
    echo fgets($myData);

    while(!feof($myData)){
        echo fgets($myData) . "<br>";
    }
    ?>

<!-- get character from file -->
    fgetc($fileName);

<!-- return file pointer to start -->
    rewind($fileName);

<!-- create new file  in write mode -->
    fopen("data.txt","w");

<!-- write text to file  -->
    fwrite($fileName,$test);

<!-- append to a file  -->
    file_put_contents($fileName,"New content", FILE_APPEND);

```

```
<!-- get file size  -->
    filesize($fileName);

<!-- check if file is exist or not  -->
    file_exists($fileName);

<!-- get file type  -->
    filetype($fileName);

<!-- get list of files and dir in a path  -->
    $files = scandir("/path/to/directory");
    print_r($files);

<!-- delete a file in dir  -->
    unlink($fileName);

<!-- rename a file  -->
    rename("oldfile.txt", "newfile.txt");

<!-- get absolute  path (full path) -->
    $absolute_path = realpath($fileName);

<!-- get relative path  -->
    $directory = dirname("fullPath");

<!-- examples ------------------------------ -->
<!-- 1. Reading from a File: -->
    <?php
    // Reading a file line by line
    $filename = "example.txt";

    $file = fopen($filename, "r");

    if ($file) {
        while (!feof($file)) {
            $line = fgets($file);
            echo $line;
        }

        fclose($file);
    } else {
        echo "Error opening the file.";
    }
    ?>

<!-- 2. Writing to a File: -->
    <?php
    // Writing to a file
    $filename = "output.txt";

    $file = fopen($filename, "w");

    if ($file) {
        fwrite($file, "Hello, World!\n");
        fwrite($file, "This is a new line.");

```

```
        fclose($file);
        echo "Content written to the file successfully.";
    } else {
        echo "Error opening the file for writing.";
    }
    ?>

<!-- 3. File Upload Handling: -->
    <?php
    // Handling file uploads
    $target_dir = "uploads/";
    $target_file = $target_dir . basename($_FILES["file"]["name"]);

    if (move_uploaded_file($_FILES["file"]["tmp_name"], $target_file)) {
        echo "File uploaded successfully.";
    } else {
        echo "Error uploading file.";
    }
    ?>

<!-- 4. Checking if a File Exists:  -->
    <?php
    // Checking if a file exists
    $filename = "example.txt";

    if (file_exists($filename)) {
        echo "The file $filename exists.";
    } else {
        echo "The file $filename does not exist.";
    }
    ?>

<!-- 5. Deleting a File: -->
    <?php
    // Deleting a file
    $filename = "file_to_delete.txt";

    if (unlink($filename)) {
        echo "File $filename has been deleted.";
    } else {
        echo "Error deleting the file.";
    }
    ?>

<!-- 6. Directory Listing: -->
    <?php
    // Listing files in a directory
    $directory = "path/to/files/";

    $files = scandir($directory);

    foreach ($files as $file) {
        if ($file != "." && $file != "..") {
            echo $file . "<br>";
        }
    }
```

```
    ?>

<!-- 7. Renaming a File: -->
    <?php
    // Renaming a file
    $old_name = "oldfile.txt";
    $new_name = "newfile.txt";

    if (rename($old_name, $new_name)) {
        echo "File successfully renamed.";
    } else {
        echo "Error renaming the file.";
    }
    ?>

<!-- Handling File Paths: -->
    <?php
    // Working with file paths
    $absolute_path = realpath("file.txt");
    $directory = dirname("/path/to/file.txt");

    echo "Absolute Path: $absolute_path<br>";
    echo "Directory: $directory";
    ?>

<!--*poo  --> ------------------------------------------

<!-- create a class -->
    <?
    class fruits{
//  <!-- default public  -->
    public $name;
    public $colors;

    public function setName($name){
    $this->name=$name;
    }
    public function  getName(){

    return $this->name;
    }
    }
    ?>

<!-- constructor : -->
    <?php
    class ClassName{
    public function __construct(){
        // code
    }

// !-- destructor  -->
    public function __destruct(){

        // code
    }
```

```
    };
    ?>

<!-- create an instance from a class -->
    <?php
    $apple = new fruits();
    $apple->setName("apple");
    $apple->getName();
    ?>

<!-- declare a static variable -->
    <?php
    class className{
    public static int $ObjectCounter = 0;
    };
    ?>

<!-- access  to a static variable into the class -->
    <?php
    self::$StaticVarName++;
    ?>
<!-- access to static var outside the class -->
    <?php
    className:$staticVarName;
    ?>

<!-- inheritance  -->
    <?php
    class Parent{

    };

    class child extends Parent{

    };
    ?>
<!--   example  1   -->
  <?php
    class fruit
    {
        public $name;
        public $color;

        public function getName()
        {
            return $this->name;
        }
        public function getColor()
        {
            return $this->color;
        }
        public function setName($name)
        {
            $this->name = $name;
        }
        public function setColor($color)
```

```
        {
            $this->color = $color;
        }
    };
    $mongo = new fruit();
    $mongo->setColor("red");
    $mongo->setName("mongo");

    echo "the name is  : " . $mongo->getName() . "\n <br>";
    echo "the color is : " .  $mongo->getColor() . "\n <br>";
    ?>


<!-- using strict type -->
    <?php
    declare(strict_types=1);
    ?>

<!-- example 2  -->
    <?php
    class Pont {
        public float $width;
        public float $height;
        public static int $ObjectCounter = 0;

        // increment  constructor
        public function __construct() {
            // Increment the static counter when an object is created
            self::$ObjectCounter++;
        }
    }

// Creating an instance of the class
    $p1 = new Pont();
    $p1 = new Pont();
    $p1 = new Pont();
    // Displaying the number of objects
    echo "The number of objects in the class: " . Pont::$ObjectCounter;
    ?>

```