

sumrray\sumBroCode.php

```
1
2 <!-- php tag -->
3 <?php
4 ?>
5
6 <!-- display a message -->
7 <?php
8 echo "your text";
9 ?>
10 <!-- break line -->
11 <?php
12 echo "text<br>"
13 ?>
14
15 <!-- comments -->
16 <?php
17 // one line
18 # one line
19
20 /*
21 multiline
22
23 */
24 ?>
25
26 <!-- variables -->
27 <?php
28 $variableName='value';
29 ?>
30
31 <!-- data types -->
32 PHP is a loosely typed language
33 /*
34 integer
35 float
36 string
37 boolean
38 array
39 object
40 null
41 */
42
43 <!-- constant -->
44 <!--
45 if you wanna to use constant var you don't need to add $ :
46 -->
47 <?php define("VariableName",value); ?>
48
49
50 <!-- variable functions -->
51 <?php
52 var_dump($var); # check data type with more details :
53 isset($var); #Checks if a variable is set and is not null.
```

```

54  unset($var);      # Destroys a variable.
55  empty($ar);       # Checks if a variable is empty.
56  gettype($var);    # Returns the data type of a variable.
57  settype($var,"newType")
58  ?>
59
60  <!-- global keyword -->
61  <?php
62  $name = "ayoub";
63
64  function testing() {
65  global $name; // Access the global variable $name
66  echo $name;
67  }
68
69  testing(); // This will correctly output "ayoub"
70  ?>
71
72  <!-- variable data type conversion : --> -----
73  <!--
74      implicit Type Conversion: PHP will automatically convert data types when necessary.
75  -->
76  <!-- Type Casting: -->
77  <?php
78  (int) or (integer) # Casts a value to an integer data type.
79  (float) or (double) # Casts a value to a floating-point (decimal) number.
80  (string) # Casts a value to a string.
81  (array) # Casts a value to an array.
82  (object) # Casts a value to an object.
83  (bool) or (boolean) # Casts a value to a boolean (true or false).
84  ?>
85
86  <!-- Explicit Type Conversion Functions: -->
87  <?php
88  intval($str); #Converts a value to an integer.
89  floatval($str); # or doubleval(): Converts a value to a floating-point number.
90  strval($nbr); #Converts a value to a string.
91  boolval($str); #Converts a value to a boolean.
92  ?>
93
94  <!-- concat string with variables --> -----
95  <?php
96  echo "text $variableName";
97  echo "text {$variableName}";
98  echo "text" . $variableName . "text";
99  ?>
100 <!-- example -->
101 <?php
102 $name="bro Code";
103 $food="pizza";
104 $email="ayoubmajid71@gmail.com<br>";
105 $age=21;
106 $users=2;
107 $quantity=3;
108 $gpa=2.5;
109 $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 :

```

```

166         --$variableName;
167     ?>
168
169 <!-- $_GET, $_POST --> -----
170     // $_GET, $_POST = special variables used to collect data from an HTML form
171     //data is sent to the file in the action attribute of <form>
172     <form action="some_file.php" method="get">
173
174     $_Get = Data is appended to the url
175             NOT SECURE
176             char limit
177             Bookmark is possible w/ values
178             GET requests can be cached
179             Better for a search page
180
181     //$ POST = Data is packaged inside the body of the HTTP request
182             MORE SECURE
183             No data limit
184             Cannot bookmark
185             GET requests are not cached
186
187     <!-- get info when the form submitted --> -----
188     <?php
189         $_GET["inputNameAttribute"]; /* it will return the value of input
190         when the form submitted */
191
192         $_POST["inputNameAttribute"]; /* it will return the value
193         of input when the form submitted*/
194     ?>
195
196     <!-- example -->
197     <form action="index.php" method="GET">
198         <input type="text" name="userName" placeholder="enter your name">
199         <input type="submit">
200     </form>
201 <?php
202 echo $_GET["userName"];
203 ?>
204
205 <!-- THE difference between get and post mode --> -----
206
207     <!--
208         get : send info as a query params
209         post : send info as a body params
210
211         using :
212         get insensible information : (small )
213         post sensible information like password user token ( maybe big data )
214     -->
215     <!-- example -->
216     <form action="index.php" method="post">
217         <input type="number" placeholder="enter the quantity" name="quantity">
218         <input type="submit" value="log in ">
219     </form>
220 <?php
221 $item="pizza";

```

```

222     $price=5.99;
223     $quantity=$_POST['quantity'];
224     $total=$quantity*$price;
225     echo "you have ordered $quantity *$price <br>";
226     echo "the total is $$total <br>";
227     // get user info :
228     ?>
229
230 <!-- functions --> -----
231     <!-- get max of a range of values -->
232         <?php max(val1,val2,valeN); ?>
233
234     <!-- get min fo a range of values -->
235         <?php min(val1,val2,valN); ?>
236
237     <!-- get random number -->
238         <?php rand(min,max); ?>
239
240     <!-- get the sqrt of number -->
241         <?php sqrt(val); ?>
242
243     <!-- get the ceil value of a number -->
244         <?php ceil(val); ?>
245
246     <!-- get the floor value of a number -->
247         <?php floor(val); ?>
248
249     <!-- get the round of a number -->
250         <?php round(val); ?>
251
252     <!-- identify number of decimal after the comma -->
253         <?php round(number,numberAfterComma); ?>
254
255     <!-- get the pi value -->
256         <?php pi(); ?>
257
258     <!-- application -->
259         <form action="index.php" method="post">
260             <label for="r">radius</label>
261             <input type="text" placeholder="enter the radius " name="r" id="r"><br>
262             <input type="submit" value="calculate">
263         </form><br>
264         <?php
265             // $x=$_POST['x'];
266             // $y=$_POST['y'];
267             $radius=$_POST['r'];
268
269             // circumference : 2PR
270             // aria          : PR**2
271             // volume        : (4/3 )PR^3
272
273             $circumference=round(2*pi()*$radius,2);
274             $aria=round(pi()*$radius**2,2);
275             $volume = round((4/3)*pi() *$radius**3,2);
276             echo "the circumference is : {$circumference}cm <br>" ;
277             echo "the aria is : {$aria}cm^2 <br>";

```

```

278     echo "the volume is : {$volume}cm^3 <br>" ;
279     ?>
280
281 <!-- if statement --> -----
282     if some condition is true do something
283     if condition is not true don't do it
284     <?php
285     if(condition){
286     // do some statements
287     }
288     else if (condition){
289     // do some statements
290     }
291     else{
292         // do some statements :
293     }
294     ?>
295
296 <!-- example 1 -->
297     <?php
298     $age = 99;
299     if($age>=100){
300     echo "you're to old to enter this site : ";
301     }
302     else if ($age >= 18) {
303         echo "You may enter this site <br>";
304     } elseif ($age <= 0) {
305         echo "That was not a valid age ";
306     } else {
307         echo "you must be 18+ to enter ";
308     }
309     ?>
310 <!-- example 2 -->
311     <?php
312     $hours = 50;
313     $rate = 15;
314     $weeklyPay = 0;
315     if ($hours < 0) {
316         $weeklyPay = 0;
317     } else if ($hours <= 40) {
318         $weeklyPay = $hours * $rate;
319     } else {
320         $weeklyPay = ($rate * 40) + ($hours - 40) * ($rate * 1.5);
321     }
322     echo "you made $$weeklyPay this week<br>";
323
324     ?>
325
326 <!-- relational operators
327     > grater than
328     < Lower than
329     >= grater than or equal
330     <= lower than or equal
331     != differ than
332     != differ than + (differ type also )
333     == => equal to

```

```

334     === => equal to + (the same data type)
335     -->
336
337 <!-- Logical operator
338     && => (and)
339     || => (or )
340     !  => (not)
341     -->
342 <!-- example 1 -->
343 <?php
344     $age = 25;
345     $citizen = true;
346     if ($age >= 18 && $citizen) {
347         echo "You can vote <br>";
348     } else {
349         echo "You can not vote <br>";
350     }
351     ?>
352 <!-- example 2 -->
353 <?php
354     $child = false;
355     $senior = false;
356     $ticket = null;
357     if ($child || $senior) {
358         $ticket = 10;
359     } else {
360         $ticket = 15;
361     }
362     echo "the ticket prise is $$ticket<br>";
363     ?>
364
365 <!-- switch --> -----
366     replacement to using many elseif statements more efficient
367     <?php
368     switch($variableName){
369
370     case val1:
371         #code
372         break;
373
374     case val2:
375         #code
376         break;
377
378     case val3:
379         #code
380         break;
381         //....
382
383     default :
384         #code
385     }
386     ?>
387 <!-- example -->
388 <?php
389     $grade = "A";

```

```

390     switch ($grade) {
391         case 'A':
392             echo "You did great";
393             break;
394
395         case 'B':
396             echo "You did good";
397             break;
398
399         case 'C':
400             echo "You did okay";
401             break;
402
403         case 'D':
404             echo "You did poorly";
405             break;
406         case 'F':
407             echo "You failed";
408             break;
409         default:
410             echo "invalid grade ";
411     }
412     ?>
413
414 <!-- For Loop --> -----
415     repeat some code a certain # of times
416
417     <?php
418     for ($i=start ; condition ; step){
419         //code
420     }
421     ?>
422 <!-- example 1-->
423 <?php
424 for ($i = 1; $i <= 10; $i++) {
425     echo "$i-Hello <br>";
426 }
427 ?>
428 <!-- example 2 -->
429 <form action="index.php" method="post">
430     <label for="counter">enter a number to count to</label><br>
431     <input type="number" name="counter" m><br>
432     <input type="text" placeholder="the name will be counted " name="name"><br>
433     <input type="submit" value="counter">
434 </form><br>
435
436 <?php
437 $counter = 0;
438 $name = "";
439 if (isset($_POST['counter']) && isset($_POST['name'])) {
440     $counter = $_POST['counter'];
441     if ($counter < 0) $counter = 0;
442     $name = $_POST['name'];
443 }
444 for ($i = 1; $i <= $counter; $i++) {
445     echo "$i -$name <br>";

```



```

446     }
447     ?>
448
449     <!-- while condition : -->
450     <?php
451         while(condition){
452             //code;
453         }
454     ?>
455
456     <!-- example -->
457     <form action="index.php" method="post">
458         <input type="submit" value="stop">
459     </form><br>
460     <?php
461     $seconds = 0;
462     $running = true;
463     while ($running) {
464         $seconds++;
465         if (isset($_POST['stop'])) break;
466         echo "the current amount of seconds $seconds <br>\n";
467     }
468     ?>
469
470     <!-- do while -->
471     <?php
472     do {
473         // code
474     }while(condition);
475     ?>
476
477     <!-- foreach -->
478     <?php
479     foreach( $elements as $item){
480         // code
481     };
482     ?>
483
484     <!-- example -->
485     <?php
486     foreach($family as $item){
487         echo $item . "<br>";
488     }
489     ?>
490
491     <!-- array in php --> -----
492     pointer that point at sequence of memory cases
493     <?php
494     $arrayName=array(value1,value2,value3,/*...*/valueN);
495     ?>
496     <!-- or -->
497     <?php
498     $arrayName=[value1,value2,valueN];
499     ?>
500
501     <!-- access to arrays element -->
502     <?php

```

```

502     $arrayName[index]=value;
503     ?>
504
505 <!-- display array using foreach -->
506 <?php
507     $foods = array("apple", "orange", "banana", "coconut");
508     foreach ($foods as $food) {
509         echo $food . "<br>";
510     }
511     #or
512     print_r($foods);
513     ?>
514
515 <!-- get length of array -->
516     count($arrayName);
517
518 <!-- push element the the end on an array -->
519 <?php
520     array_push($arrayName,"value");
521     #or
522     $arrayName[]="value";
523     ?>
524
525 <!-- pop element from array -->
526 <?php
527     array_pop($arrayName);
528     ?>
529
530 <!-- shift left array -->
531 <?php array_shift($arrayName); ?>
532
533 <!-- get reversed version from your array -->
534 <?php $reversedArray= array_reverse($arrayName); ?>
535
536 <!-- check if an element in array -->
537 <?php in_array("value", $arrayName)?>
538
539 <!-- searching with key -->
540 <?php array_key_exists("key", $arrayName) ?>
541
542 <!-- convert array to string -->
543 <?php $string = implode("separator", $arrayName); ?>
544
545 <!-- associative Array -----
546 provide you to use associative names rather than indexes :
547 -->
548 <?php $arrayName=["assName1"=>Value1,"assName2"=>value2]; ?>
549 <!-- you can using index just with elements that does not has an
550 associative name
551 (index did not start form first element in array => start from first not
552 associative item)
553 -->
554 <!-- example 1 -->
555 <?php
556 $capitals = [
557     "USA" => "Washington",

```

```

558         "JAPAN" => "kyoto",
559         "SOUTH KOREA" => "Seoul",
560         "MOROCCO" => "Rabat"
561     ];
562     foreach ($capitals as $key => $value) {
563         echo $key . "=$value " . "<br>";
564     }
565
566     $capitals["Algeria"] = "Algeria";
567     foreach ($capitals as $key => $value) {
568         echo $key . "=$value " . "<br>";
569     }
570     ?>
571 <!-- example 2 -->
572 <?php
573     $studentName = "ayoub";
574     $score = [
575         "ayoub" =>
576         [
577             "score" => "19",
578             "grade" => "A"
579         ],
580         "adam" =>
581         [
582             "score" => "15",
583             "grade" => "B"
584         ],
585         "youness" =>
586         [
587             "score" => "14",
588             "grade" => "C"
589         ]
590     ];
591     echo "mark : " . $score[$studentName]["score"] . "<br>";
592     echo "score : ", $score[$studentName]["grade"];
593     ?>
594
595 <!-- get array keys -->
596 <?php $keys=array_keys($arrayName); ?>
597
598 <!-- key array values -->
599 <?php $values=array_values($arrayName); ?>
600
601 <!-- flip between key and values -->
602 <?php $flippedArray=array_flip($arrayName); ?>
603
604 <!-- example 3 -->
605 <form action="index.php" method="post">
606     <label for="">enter a country </label>
607     <input type="text" name="capital">
608     <input type="submit" value="submit">
609 </form><br>
610 <?php
611 $capitals = [
612     "USA" => "Washington",
613     "JAPAN" => "kyoto",

```

```

614     "SOUTH KOREA" => "Seoul",
615     "MOROCCO" => "Rabat"
616 ];
617 if (isset($_POST['capital'])) {
618     $capital = strtoupper($_POST['capital']);
619     if (array_key_exists($capital, $capitals))
620         echo "the capital is : " . $capitals[$capital] . "<br>";
621     else {
622         echo "this capital not found ";
623     }
624 }
625 ?>
626
627 <!-- isset($variableName) -->
628     returns true if variable is declared and not null
629
630 <!-- empty($variableName) -->
631     return true if a variable is not declared || false || null || ""\
632
633 <!-- unset($variableName) -->
634     removes the reference to the array,
635     and the variable $myArray will no longer be defined
636
637 <!-- create an array when setting the name of inputs -->
638     <input type="text" name="name[]">
639
640 <!-- example 1 -->
641     <form action="index.php" method="post">
642         <label for="">username :</label>
643         <input type="text" name="username"><br>
644         <label for="">password :</label>
645         <input type="text" name="password"><br>
646         <input type="submit" value="Log in" name="login">
647     </form><br>
648     <?php
649     foreach ($_POST as $key => $value) {
650         echo "$key = $value <br>";
651     }
652     if (isset($_POST["login"])) {
653
654         $username = $_POST["username"];
655         $password = $_POST["password"];
656         if (empty($username)) {
657             echo "Username is missing <br>";
658         } else if (empty($password)) {
659             echo "Password is missing <br>";
660         } else {
661             echo "\"hello\" $username <br>";
662             echo "\"hello\" $password <br>";
663         }
664     }
665     ?>
666 <!-- example 2 -->
667     <form action="index.php" method="post">
668
669     <input type="radio" name="creditCard" value="Visa" id="Visa" required>

```

```

670     <label for="Visa">Visa</label><br>
671
672     <input type="radio" name="creditCard" value="Mastercard" id="Mastercard" required>
673     <label for="Mastercard">Mastercard</label><br>
674
675     <input type="radio" name="creditCard" value="American Express" id="American Express"
required>
676     <label for="American Express">American Express</label><br><br>
677     <hr><input type="submit" name="submit">
678
679 </form><br>
680 <?php
681
682 if (isset($_POST['submit'])) {
683
684     $userCard = $_POST['creditCard'];
685
686     echo "You selected : $userCard <br>";
687 } else {
688 }
689
690 ?>
691 <!-- functions --> -----
692 write some code once, reuse when you need it
693     <?php
694     function functionName(){
695         // code
696
697     }
698     ?>
699 <!-- example 1 -->
700     <?php
701     function happyBirthday($name, $age)
702     {
703         echo "happy Birthday dear $name! <br>";
704         echo "happy Birthday dear $name! <br>";
705         echo "happy Birthday dear $name! <br><br>";
706         echo "you are $age years old <br>";
707     }
708     happyBirthday("ayoub", 20);
709     happyBirthday("amine", 25);
710     happyBirthday("amal", 18);
711     ?>
712 <!-- example 2 -->
713     <?php
714     function isEven(int $number){
715
716         return $number%2 ? "is even" : "is odd";
717     }
718     echo isEven(5);
719     ?>
720 <!-- example 2 with strict typing -->
721     PHP 7 introduced scalar type declarations and return type declarations.
722     By using strict typing, you can enforce that only values of the
723     correct type are allowed. This can be done by declaring the types
724     for function arguments and return values explicitly:

```

```

725     <?php
726     declare(strict_types=1);
727     function isEven(int $number){
728         return $number%2 ? "is yes" : "is odd";
729     }
730     echo isEven('5'); // it will give strict typing error :
731     ?>
732
733     <!-- return multiple values -->
734     <?php
735     function functionName(){
736
737         return [val1,val2,valN];
738     }
739     ?>
740
741     <!-- get multiple table from functions -->
742     <?php
743     list($var1,$var2,$varN)=functionName();
744     #or
745     [$var1,$var2,$varN]=functionName();
746     ?>
747
748     <!--example -->
749     <?php
750     function getMultipleValues() {
751         $value1 = "Hello";
752         $value2 = "World";
753         $value3 = 42;
754         // Return an array with multiple values
755         return array($value1, $value2, $value3);
756     }
757     // Call the function and receive multiple values
758     list($result1, $result2, $result3) = getMultipleValues();
759
760     // Output the results
761     echo $result1 . " " . $result2 . " " . $result3;
762     ?>
763
764     <!-- example 3 -->
765     <form action="index.php" method="post" onsubmit=" return checkCheckedFood(); ">
766     <input type="checkbox" name="Pizza" value="Pizza" id="Pizza">
767     <label for="Pizza">Pizza</label><br>
768     <input type="checkbox" name="Hamburger" value="Hamburger" id="Hamburger">
769     <label for="Hamburger">Hamburger</label><br>
770     <input type="checkbox" name="Hotdog" value="Hotdog" id="Hotdog">
771     <label for="Hotdog">Hotdog</label><br><br>
772     <input type="checkbox" name="Taco" value="Taco" id="Taco">
773     <label for="Taco">Taco</label><br><br>
774     <hr><input type="submit" name="submit">
775     </form><br>
776     <script>
777     function checkCheckedFood() {
778         let arrFoods = document.querySelectorAll(" form > input[type='checkbox'] ");
779         arrFoods = Array.from(arrFoods);
780         let isChecked = false;

```

```

781         arrFoods.forEach(element => {
782             if (element.checked) return isChecked = true;
783         });
784         if (!isChecked) alert("please select at least one food");
785         return isChecked;
786     }
787 </script>
788 <?php
789 function getListOfFoodSelected()
790 {
791     $strFoods = "";
792     foreach ($_POST as $key => $value) {
793         if ($key != "submit") $strFoods .= " " . $value;
794     }
795     return $strFoods;
796 }
797 if (isset($_POST['submit'])) {
798     $strFoods = getListOfFoodSelected();
799     echo "You like : $strFoods";
800 }
801 ?>
802
803 <!-- static variable -->
804 <?php static $variableName=value; ?>
805 <!-- example -->
806 <?php
807 function testing(){
808     static $number2=10;
809     echo $number2++ . "\n";
810 }
811 ?>
812 <!-- string functions --> -----
813 <!-- convert to lower case -->
814 <?php strtolower($string); ?>
815
816 <!-- convert to upper case -->
817 <?php strtoupper($string); ?>
818
819 <!-- know the length of a variable -->
820 <?php
821 strlen($string);
822 ?>
823
824 <!-- get specific index -->
825 <?php
826 $string[index];
827 ?>
828
829 <!-- assign a value of an index -->
830 <?php
831 $string[index]=newValue;
832 ?>
833
834 <!-- replace string -->
835 <?php
836 $str=str_replace(replaceSubStr,newSubStr,$string);

```

```

837     ?>
838
839     <!-- sub string -->
840     <?php
841         substr($string,startIndex,length) // default length=strlen($string)
842     ?>
843
844     <!-- get length of str -->
845     <?php strlen($str); ?>
846
847     <!-- count number of words -->
848     <?php str_word_count($str); ?>
849
850     <!-- reverse a string -->
851     <?php strrev($str); ?>
852
853     <!-- get position of a substr -->
854     <?php strpos($str,"searchedSubStr"); ?>
855
856     <!-- replace empty character -->
857     <?php str_pad($str,length,"replaceValue") ?>
858
859     <!-- shuffle string <del>bl</del> -->
860     <?php str_shuffle($str) ?>
861
862     <!-- compare between two string -->
863     <?php strcmp($str) ?>
864
865     <!-- str to array -->
866     <?php $arrStr=explode("separator",$str);?>
867
868     <!-- array to str -->
869     <?php $Str=implode("separator",$arrStr);?>
870
871     <!-- filter input --> -----
872     <?php
873     // filter special character :
874         filter_input(INPUT_POST,"name",FILTER_SANITIZE_SPECIAL_CHARS);
875
876     // filter int :remove character from right and left
877         filter_input(INPUT_POST,"name",FILTER_SANITIZE_NUMBER_INT);
878
879     // filter float :
880         filter_input(INPUT_POST,"name",FILTER_SANITIZE_NUMBER_FLOAT);
881
882     // valid email :
883         filter_input(INPUT_POST,"name",FILTER_VALIDATE_EMAIL);
884
885     // accept just valid int
886         filter_input(INPUT_POST,"name",FILTER_VALIDATE_INT);
887
888     // accept just valid float :
889         filter_input(INPUT_POST,"name",FILTER_VALIDATE_FLOAT);
890     ?>
891     <!-- example -->
892     <form action="index.php" method="post" >

```



```

893     <input type="text" name="username" placeholder="enter your name ">
894     <input type="text" name="age" placeholder="enter your age ">
895     <input type="text" name="email" placeholder="enter your email ">
896     <input type="submit" value="log in">
897 </form><br>
898 <?php
899 if (isset($_POST['username']) && isset($_POST['age']) && isset($_POST['email'])) {
900     $username = filter_input(INPUT_POST, "username", FILTER_SANITIZE_SPECIAL_CHARS);
901     $age = filter_input(INPUT_POST, "age", FILTER_VALIDATE_FLOAT);
902     $email = filter_input(INPUT_POST, "email", FILTER_VALIDATE_EMAIL);
903     echo "Hello $username <br>";
904     echo "your age is $age <br>";
905     echo "your email is $email <br>";
906 }
907 ?>
908
909 <!-- include -->
910 copy the content of a file (php/html/txt) then includes it in your file
911 <?php
912 include "path";
913 #or
914 include("path");
915 ?>
916 <!-- or -->
917 <?php
918 require "path"; // stop all rest of code
919 ?>
920
921 <!-- cookie -->
922 information about a user stored in a user's web-browser
923 targeted advertisements,browsing preferences,
924 other non-sensitive data
925 it's a piece of information stored inside our system
926 we should never set username and password using cookies
927
928 <!-- set a cookie -->
929 setcookie("key","value",ExpireDay,filePath);
930 // time() :return the current day
931 // 86400 : one day
932
933 <!-- example -->
934 <?php
935 setcookie("favoriteFood","pizza",time() +86400,"/");
936 setcookie("favoriteDrink","coffe",time() +86400*2,"/");
937 ?>
938
939 <!-- print all cookies -->
940 <?php
941 foreach ($_COOKIE as $key => $value) {
942     echo "$key = $value <br>";
943 }
944 ?>
945
946 <!-- session --> -----
947 session =SGB used to store information on a user to be used
948 across multiple pages a user is assigned a session-id

```

```

949
950 <!-- start session -->
951 <?php
952     session_start();
953 ?>
954 <!-- create new key in a session -->
955 <?php
956     $_SESSION['key']='value';
957 ?>
958 <!-- get key value -->
959 <?php
960     echo $_SESSION['key'];
961 ?>
962 <!-- get the current session id -->
963 <?php
964     session_id();
965 ?>
966 <!-- set session timeout -->
967     The ini_set('session.gc_maxlifetime', minutes); line is used to set the
968     maximum lifetime of a session. It determines how long the server
969     should keep session data before it's considered expired.
970     This setting works in conjunction with the session timeout
971     mechanism. If a user is inactive for the specified number of minutes,
972     their session may expire.
973 <?php
974     ini_set('session.gc_maxlifetime', minutes);
975 ?>
976 <!-- regenerate session id : -->
977     The session_regenerate_id() function is used to generate a new session ID
978     and replace the current session ID with the new one.
979     This is often done as a security measure to prevent session
980     fixation attacks. A session fixation attack occurs
981     when an attacker sets a user's session ID to a known value,
982     allowing them to hijack the user's session.
983 <!-- destroy session -->
984 <?php
985     session_destroy();
986 ?>
987 <!-- change to another page -->
988 <?php
989     header("Location:pageName");
990 ?>
991
992 <!-- example login page --> -----
993
994 <!-- index.php -->
995 <?php
996     session_start();
997
998     if(isset($_POST['login'])){
999         if(!empty($_POST['username']) && !empty($_POST['password'])) {
1000             $_SESSION["username"] = $_POST['username'];
1001             $_SESSION["password"] = $_POST['password'];
1002             header("location: home.php");
1003             exit();
1004         } else {

```

```

1005         echo "Missing username or password";
1006     }
1007 }
1008 ?>
1009 <form action="index.php" method="post">
1010     <input type="text" placeholder="username" name="username"><br>
1011     <input type="password" placeholder="password" name="password"><br>
1012     <input type="submit" value="login" name="login"><br>
1013 </form>
1014
1015 <!-- home.php -->
1016 <?php
1017     session_start();
1018
1019     if(!isset($_SESSION['username']) || !isset($_SESSION['password'])){
1020         header("location: index.php");
1021         exit();
1022     }
1023 ?>
1024 This is the home page.
1025     <form action="home.php" method="post">
1026         <input type="submit" value="log out" name="logout">
1027     </form>
1028
1029     <?php
1030     if(isset($_POST['logout'])){
1031         session_destroy();
1032         header("location: index.php");
1033         exit();
1034     }
1035 ?>
1036
1037 <!-- example 2 Shopping Cart Implementation -->
1038 <?php
1039     session_start();
1040
1041     // Add item to the cart
1042     if (!isset($_SESSION['cart'])) {
1043         $_SESSION['cart'] = array();
1044     }
1045     $item = array('id' => 1, 'name' => 'Product A', 'price' => 19.99);
1046     $_SESSION['cart'][] = $item;
1047
1048     // Display the shopping cart
1049     foreach ($_SESSION['cart'] as $item) {
1050         echo $item['name'] . " - $" . $item['price'] . "<br>";
1051     }
1052 ?>
1053
1054 <!-- date --> -----
1055 <!-- y/m/d -->
1056     echo date("y/m/d");
1057     echo date("y-m-d");
1058
1059 <!-- print the current day name -->
1060     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'];

```

```

1117     ?>
1118 <!-- REQUEST METHOD -->
1119     return the current request method (get is the default)
1120     <?php
1121         $_SERVER['REQUEST_METHOD'];
1122     ?>
1123     <!-- control + click see the code source-->
1124
1125 <!-- hashing --> -----
1126     hashing : transforming sensitive data (password)
1127     into letters ,numbers ,and/or symbols
1128     via a mathematical process (similar to encryption)
1129     hides the original data from 3rd parties
1130
1131
1132     In PHP, a hash is a one-way function that takes an input (or 'message')
1133     and produces a fixed-size string of characters,
1134     which is typically a hexadecimal number. The purpose of hashing is
1135     to generate a unique identifier (hash value) for a given input.
1136     Here are some key aspects of hashing in PHP:
1137
1138     Hash Functions:
1139     PHP provides several built-in hash functions that you can
1140     use for various purposes. Some commonly used hash functions include:
1141
1142     md5(): Produces a 32-character hexadecimal number.
1143     sha1(): Produces a 40-character hexadecimal number.
1144     hash(): A flexible function that supports multiple algorithms (e.g., MD5, SHA-256, SHA-
1145 512).
1146 <!-- example -->
1147     <?php
1148         $data = 'Hello, World!';
1149         $md5Hash = md5($data);
1150         $sha1Hash = sha1($data);
1151         $customHash = hash('sha256', $data);
1152     ?>
1153
1154 <!-- Password Hashing: --> -----
1155     When dealing with passwords, it's essential to use secure hashing
1156     methods to protect user credentials. PHP provides the password_hash()
1157     function for this purpose. This function uses a strong, adaptive hashing
1158     algorithm and automatically handles the generation of a salt.
1159     Example:
1160     <?php
1161         $password = 'user_password';
1162         $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
1163     ?>
1164
1165 <!-- hash a string -->
1166     <?php
1167         $hash=password_hash($password,PASSWORD_DEFAULT);
1168     ?>
1169
1170 <!-- check if string ==hash version -->
1171     <?php

```

```

1172 if(password_verify($password,$hash)){
1173
1174     echo "you logged in ";
1175 }
1176 else{
1177     echo "incorrect password";
1178 }
1179 }
1180 ?>
1181
1182 <!-- different between hashing and encryption -->
1183 Encryption and hashing are both cryptographic techniques,
1184 but they serve different purposes and have distinct characteristics:
1185
1186 1. Purpose:
1187 - Encryption: The primary purpose of encryption is
1188   to protect data confidentiality. It involves transforming data
1189   (plaintext) into an unreadable format (ciphertext) using an algorithm
1190   and a secret key. The goal is to ensure that only authorized parties with
1191   the correct key can decrypt and access the original data.
1192
1193
1194   - Hashing: The primary purpose of hashing is to create a fixed-size,
1195     irreversible representation (hash value) of data.
1196     Hash functions are commonly used to verify data integrity,
1197     generate unique identifiers, and securely store passwords.
1198
1199 2. Reversibility:
1200 - Encryption: Encryption is a reversible process
1201   . It transforms data in such a way that it can be decrypted back to its
1202   original form using the appropriate decryption key.
1203 - Hashing: Hashing is a one-way process.
1204   Once data is hashed, it cannot be feasibly reversed to obtain
1205   the original input. Hash functions are designed to be irreversible
1206   for security reasons.
1207
1208 3. Key Usage:
1209 - Encryption: Encryption involves the use of keys
1210   for both encryption and decryption. There are symmetric
1211   key algorithms where the same key is used for both operations,
1212   and asymmetric key algorithms where different keys are used for
1213   encryption and decryption.
1214 - Hashing: Hashing typically does not involve the use of keys.
1215   A fixed-size hash value is generated based solely on the input
1216   data and the hashing algorithm.
1217
1218 4. Use Cases:
1219 - Encryption: Used for securing communication,
1220   protecting sensitive data at rest (e.g., file encryption),
1221   and ensuring privacy.
1222 - Hashing: Used for data integrity verification,
1223   creating digital signatures, generating unique identifiers,
1224   and securely storing passwords.
1225
1226 5. Examples:
1227 - Encryption: AES (Advanced Encryption Standard),

```

```

1228         RSA (Rivest-Shamir-Adleman), DES (Data Encryption Standard),
1229         and others.
1230     - Hashing: MD5, SHA-256, SHA-3, and others.
1231
1232     6. Security Considerations:
1233     - Encryption: Involves managing and securing keys,
1234     and the security relies on the strength of the encryption
1235     algorithm and the secrecy of the key.
1236     - Hashing: Focuses on collision resistance
1237     (the likelihood of two different inputs producing the same hash)
1238     and pre-image resistance (the difficulty of finding an input that
1239     produces a specific hash).
1240
1241     In summary, encryption is about keeping data confidential and
1242     reversible, while hashing is about creating fixed-size irreversible
1243     representations for various purposes like data integrity verification
1244     and password storage. Depending on the specific security requirements
1245     of a system, both encryption and hashing may be used in complementary ways.
1246
1247     <!-- my sql extension --> -----
1248
1249     <!-- connect to database -->
1250         <?php
1251             $dbServer = "localhost";
1252             $dbUser = "root";
1253             $dbPassword = "";
1254             $dbName = "test";
1255             $connection = "";
1256
1257             $connection = mysqli_connect(
1258                 $dbServer,
1259                 $dbUser,
1260                 $dbPassword,
1261                 $dbName
1262             );
1263             ?>
1264
1265     <!-- check connection error -->
1266         <?php
1267             $connection->connect_error();
1268             ?>
1269
1270     <!-- get result of a query -->
1271         <?php
1272             $query="sql code";
1273             $result = mysqli_query($connection, $query);
1274             ?>
1275
1276     <!-- count number of row from response -->
1277         <?php
1278             mysqli_num_rows($result)
1279             ?>
1280
1281     <!-- fetch response as an(associative Array) one row -->
1282         <?php
1283             $row=mysqli_fetch_assoc($result);

```

```

1284     ?>
1285
1286 <!-- close a connection -->
1287 <?php
1288     $connection->close();
1289     // or
1290     mysqli_close($connection);
1291     ?>
1292
1293 <!-- example 1 (database.php) -->
1294 <?php
1295     $dbServer = "localhost";
1296     $dbUser = "root";
1297     $dbPassword = "";
1298     $dbName = "test";
1299     $connection = "";
1300     try {
1301         $connection = mysqli_connect(
1302             $dbServer,
1303             $dbUser,
1304             $dbPassword,
1305             $dbName
1306         );
1307         if ($connection)
1308             if ($connection->connect_error) {
1309                 // If connection fails, throw a mysqli_sql_exception
1310                 throw new mysqli_sql_exception("Connection failed: " . $connection->
connect_error);
1311             }
1312             echo "you connected with success";
1313         } catch (mysqli_sql_exception $e) {
1314
1315             echo $e->getMessage();
1316         }
1317     ?>
1318 <!-- example 2 : -->
1319 <?php
1320     include "database.php";
1321     $sql="insert into employee values ('adam','doma1')";
1322     try{
1323         mysqli_query($connection,$sql);
1324         echo "user is now registered";
1325     }catch(mysqli_sql_exception $e){
1326
1327         echo "DataBase Error : " . $e->getMessage();
1328     }
1329     ?>
1330 <!-- example 3 -->
1331 <?php
1332     include "database.php";
1333     $username = "ayoub";
1334     $password = "youbista123";
1335     $hasPassword = password_hash($password, PASSWORD_DEFAULT);
1336     $sql = "insert into employee values ('$username','$hasPassword')";
1337
1338     // exception handling :

```



```

1339     try {
1340         mysqli_query($connection, $sql);
1341         echo "user is now registered";
1342     } catch (mysqli_sql_exception $e) {
1343
1344         echo "DataBase Error : " . $e->getMessage();
1345     }
1346
1347     ?>
1348 <!-- example 4 (fetch data) -->
1349 <?php
1350     include "database.php";
1351
1352     $sql = "select * from employee where username='amina'";
1353
1354     try {
1355
1356         $result=mysqli_query($connection, $sql);
1357         if(mysqli_num_rows($result) >0){
1358
1359             $row=mysqli_fetch_assoc($result);
1360             echo "username : " . $row['username'] . "<br>";
1361             echo "password : " . $row['password'] . "<br>";
1362         }
1363     } catch (mysqli_sql_exception $e) {
1364
1365         echo "DataBase Error : " . $e->getMessage();
1366     }
1367
1368     ?>
1369 <!-- fetch multiple lines -->
1370 <?php
1371     include "database.php";
1372     $sql = "select * from employee where username='ayoub'";
1373     try {
1374         $result = mysqli_query($connection, $sql);
1375         if (mysqli_num_rows($result) > 0) {
1376             while ($row = mysqli_fetch_assoc($result)) {
1377                 echo "username : " . $row['username'] . "<br>";
1378                 echo "password : " . $row['password'] . "<br><br>";
1379             }
1380         }
1381     } catch (mysqli_sql_exception $e) {
1382
1383         echo "DataBase Error : " . $e->getMessage();
1384     }
1385     ?>
1386
1387 <!-- file -->
1388 <!--
1389     add enctype=enctype="multipart/form-data" in the form
1390     add input with type file :
1391     -->
1392
1393 <!-- get file by name -->
1394 <?php

```

```
1395     $_FILES['fileName'];
1396     ?>
1397
1398     <!-- get file name -->
1399     <?php
1400     $file['name'];
1401     ?>
1402
1403     <!-- get file full path -->
1404     <?php
1405     $file['tmp_name'];
1406     ?>
1407
1408     <!-- get file type -->
1409     <?php
1410     $file['type'];
1411     ?>
1412
1413     <!-- get file size -->
1414     <?php
1415     $file['size'];
1416     ?>
1417
1418     <!-- ensure return file name -->
1419     <?php
1420     basename($file['name'])
1421     ?>
1422
1423     <!-- specify dir to store a file -->
1424     <?php
1425     $targetPath= 'dirName/' . basename($file['name']);
1426     ?>
1427
1428     <!-- upload a file to a dir -->
1429     <?php
1430     //$filePath : $file['tmp_name'];
1431     move_uploaded_file($filePath,$targetPath);
1432     ?>
1433     <!-- check if file is already exists -->
1434     <?php
1435     file_exists($targetPath);
1436     ?>
1437
1438     <!-- get file content -->
1439     <?php
1440     file_get_contents($targetPath);
1441     ?>
1442
1443     <!-- escape html special character -->
1444     <?php
1445     echo htmlspecialchars("content");
1446     ?>
1447
1448     <!-- example -->
1449     <form action="index.php" method="post" enctype="multipart/form-data">
1450     <input type="file" name="file">
```

```

1451         <button type="submit" name="submit">Submit</button>
1452     </form>
1453     <?php
1454     if (isset($_POST['submit'])) {
1455         if (isset($_FILES['file'])) {
1456             $file = $_FILES['file'];
1457             $filePath = './uploaded/' . basename($file['name']);
1458             if (file_exists($filePath)) {
1459                 echo "File already exists.";
1460             } else {
1461                 echo $file['tmp_name'] . "<br>";
1462                 echo basename($file['name']) . "<br>";
1463                 if (move_uploaded_file($file['tmp_name'], $filePath)) {
1464                     echo "File uploaded successfully.";
1465                     $fileContent = file_get_contents($filePath);
1466                     echo "<pre>htmlspecialchars($fileContent)</pre>";
1467                 } else {
1468                     echo "Error uploading file. Check for errors: "
1469                     . $file['error'];
1470                 }
1471             }
1472         } else {
1473             echo "<br>No such file or directory.";
1474         }
1475     }
1476     ?>
1477
1478 <!-- PDO --> -----
1479
1480 <!-- create new connection -->
1481 <?php
1482 // Database credentials
1483 $host = 'your_database_host';
1484 $dbname = 'your_database_name';
1485 $username = 'your_username';
1486 $password = 'your_password';
1487
1488 // PDO connection
1489 try {
1490     $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
1491     echo "Connected successfully";
1492 } catch (PDOException $e) {
1493     echo "Connection failed: " . $e->getMessage();
1494 }
1495 ?>
1496
1497 <!-- exit script -->
1498 <?php
1499 die("message ");
1500 ?>
1501
1502 <!-- new connection example 2 -->
1503 <?php
1504
1505     $hostName = 'localhost';
1506     $dbName = 'test';

```

```

1507     $username = 'root';
1508     $password = '';
1509
1510     try{
1511
1512         $connection = new PDO(
1513             "mysql:host=$hostName;
1514             dbname=$dbName;
1515             charset=utf8",
1516             $username,
1517             $password
1518         );
1519
1520         echo "connected with success <br>";
1521     }catch(Exception $e){
1522
1523         die("Error : ". $e->getMessage());
1524     }
1525     ?>
1526 <!-- activate the details error -->
1527 <?php
1528     $connection->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
1529
1530 #OR ADD AS A PRAMS IN CONNECTION :
1531     $connection = new PDO(
1532         "mysql:host=$hostName;
1533         dbname=$dbName;
1534         charset=utf8",
1535         $username,
1536         $password,
1537         [PDO::ATTR_ERRMODE =>PDO::ERRMODE_EXCEPTION]
1538     );
1539     ?>
1540
1541 <!-- prepare the request in PDOStatement object -->
1542 <?php
1543     $query="query";
1544     $requestPdoObject=$connection->prepare($query);
1545     ?>
1546
1547 <!-- send the request to the database -->
1548 <?php
1549     $requestPdoObject->execute();
1550     ?>
1551
1552 <!-- get the response from the db : -->
1553 <?php
1554
1555     // get the first row of the response :
1556     $queryResponse=$requestPdoObject->fetch('fethMode');
1557
1558     //get all line in the response :
1559     $queryResponse=requestPdoObject->fetchAll();
1560     ?>
1561
1562 <!-- fetch modes -->

```

```

1563 <!--
1564     ⚠Tableau associatif : PDO::FETCH_ASSOC
1565     ⚠Tableau indexé : PDO::FETCH_NUM
1566     ⚠Les deux à la fois : PDO::FETCH_BOTH (par défaut)
1567     ⚠Objet : PDO::FETCH_OBJ
1568     -->
1569
1570 <!-- markers (placeholder) -->
1571 <!--
1572     using to replace a value in a request to avoid the sql injection
1573     ! you can't set the table name as a marker (placeholder)
1574     -->
1575 <?php
1576 $query=' ...:markerKey1..makerKey2';
1577
1578 $requestPdoObject->execute([
1579     'markerKey1'=>'value',
1580     'markerKey2'=>'value'
1581
1582     ])
1583     ?>
1584
1585 <!-- ? the same to marker -->
1586 <?php
1587 $query=' ...:?...?';
1588
1589 $requestPdoObject->execute(['value1','value2']);
1590     ?>
1591
1592 <!-- using bindPrms rather than passing array in execute -->
1593 <?php
1594 $requestPdoObject->bindPrms(':makerKey1','value');
1595 $requestPdoObject->bindPrms(':makerKey2','value');
1596
1597     ?>
1598
1599 <!-- case matching : -->
1600 preg_match is a PHP function that performs a regular expression match.
1601 It is used to check if a string matches a given pattern,
1602 specified by a regular expression. The function returns 1 if the pattern
1603 matches, 0 if it does not, or false if an error occurs.
1604 <?php
1605 preg_match(pattern, string);
1606
1607 // patten :
1608 // check if a string is in the word is in string :
1609 preg_match('#str#', string);
1610     ?>
1611
1612 <!-- patterns -->
1613 Regular Expressions (Regex):
1614 Regular expressions are a powerful tool for matching patterns
1615 in strings. They consist of a sequence of characters
1616 that define a search pattern. Regex patterns can include:
1617
1618 💎 Literal Characters: characters that match themselves.

```

```

1619             Metacharacters: Special characters with a reserved
1620             meaning, such as
1621             . (any character),
1622             * (zero or more occurrences),
1623             + (one or more occurrences),
1624             etc.
1625     ✦ Character Classes: Specify a set of characters to match, like
1626         [0-9] for any digit.
1627
1628     ✦ Quantifiers: Indicate the number of occurrences,
1629         such as * (zero or more),
1630         + (one or more),
1631         ? (zero or one).
1632
1633     ✦ Anchors: Define the position in the string, such as
1634         ^ (start of the line),
1635         $ (end of the line).
1636
1637     ✦ Groups and Capturing: Use parentheses to create groups,
1638         and captured groups store matched substrings.
1639
1640 <!-- example -->
1641 <?php
1642     $hostname = 'localhost';
1643     $dbName = 'test';
1644     $username = 'root';
1645     $password = '';
1646     try {
1647         $connection = new PDO(
1648             "mysql:host=$hostname;
1649             dbname=$dbName;
1650             charset=utf8",
1651             $username,
1652             $password
1653         );
1654         $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
1655         echo "connected with success <br>";
1656         $query = "select * from employee where username= :username";
1657         $requestObject = $connection->prepare($query);
1658         // send the query the mysql :
1659         $requestObject->execute([
1660
1661             'username'=>'ayoub'
1662         ]);
1663
1664         // get the response from the mysql :
1665         $response = $requestObject->fetchAll(PDO::FETCH_ASSOC);
1666         echo "<br>";
1667         print_r($response);
1668         foreach ($response as $key => $value) {
1669             echo "<br> </h1>" . $key . "</h1> <br> ";
1670
1671             foreach ($value as $SubKey => $value) {
1672
1673                 echo $SubKey . " = " . $value . "<br>";
1674             }

```

```

1675     }
1676 } catch (Exception $e) {
1677
1678     die("Error : " . $e->getMessage());
1679 }
1680
1681
1682     ?>
1683
1684 <!-- file handling ----- -->
1685
1686 <!-- read from file : -->
1687     readfile("path");
1688
1689 <!-- open file -->
1690     $fileName=fopen("path","mode");
1691     <!--
1692     mode :
1693     r: read mode
1694     w : write mode
1695     a : append mode
1696     -->
1697
1698 <!-- read from opened file -->
1699     echo fread($fileName,filesize("path"));
1700
1701 <!-- close file -->
1702     fclose($fileName);
1703
1704 <!-- read one line -->
1705     fgets($fileName);
1706
1707 <!-- read until you read end of file -->
1708     <?php
1709     echo fgets($myData);
1710
1711     while(!feof($myData)){
1712         echo fgets($myData) . "<br>";
1713     }
1714     ?>
1715
1716 <!-- get character from file -->
1717     fgetc($fileName);
1718
1719 <!-- return file pointer to start -->
1720     rewind($fileName);
1721
1722 <!-- create new file in write mode -->
1723     fopen("data.txt","w");
1724
1725 <!-- write text to file -->
1726     fwrite($fileName,$test);
1727
1728 <!-- append to a file -->
1729     file_put_contents($fileName,"New content", FILE_APPEND);
1730

```

```

1731 <!-- get file size -->
1732     filesize($fileName);
1733
1734 <!-- check if file is exist or not -->
1735     file_exists($fileName);
1736
1737 <!-- get file type -->
1738     filetype($fileName);
1739
1740 <!-- get list of files and dir in a path -->
1741     $files = scandir("/path/to/directory");
1742     print_r($files);
1743
1744 <!-- delete a file in dir -->
1745     unlink($fileName);
1746
1747 <!-- rename a file -->
1748     rename("oldfile.txt", "newfile.txt");
1749
1750 <!-- get absolute path (full path) -->
1751     $absolute_path = realpath($fileName);
1752
1753 <!-- get relative path -->
1754     $directory = dirname("fullPath");
1755
1756 <!-- examples ----- -->
1757 <!-- 1. Reading from a File: -->
1758     <?php
1759     // Reading a file line by line
1760     $filename = "example.txt";
1761
1762     $file = fopen($filename, "r");
1763
1764     if ($file) {
1765         while (!feof($file)) {
1766             $line = fgets($file);
1767             echo $line;
1768         }
1769
1770         fclose($file);
1771     } else {
1772         echo "Error opening the file.";
1773     }
1774     ?>
1775
1776 <!-- 2. Writing to a File: -->
1777     <?php
1778     // Writing to a file
1779     $filename = "output.txt";
1780
1781     $file = fopen($filename, "w");
1782
1783     if ($file) {
1784         fwrite($file, "Hello, World!\n");
1785         fwrite($file, "This is a new line.");
1786

```



```
1787         fclose($file);
1788         echo "Content written to the file successfully.";
1789     } else {
1790         echo "Error opening the file for writing.";
1791     }
1792     ?>
1793
1794 <!-- 3. File Upload Handling: -->
1795 <?php
1796 // Handling file uploads
1797 $target_dir = "uploads/";
1798 $target_file = $target_dir . basename($_FILES["file"]["name"]);
1799
1800 if (move_uploaded_file($_FILES["file"]["tmp_name"], $target_file)) {
1801     echo "File uploaded successfully.";
1802 } else {
1803     echo "Error uploading file.";
1804 }
1805 ?>
1806
1807 <!-- 4. Checking if a File Exists: -->
1808 <?php
1809 // Checking if a file exists
1810 $filename = "example.txt";
1811
1812 if (file_exists($filename)) {
1813     echo "The file $filename exists.";
1814 } else {
1815     echo "The file $filename does not exist.";
1816 }
1817 ?>
1818
1819 <!-- 5. Deleting a File: -->
1820 <?php
1821 // Deleting a file
1822 $filename = "file_to_delete.txt";
1823
1824 if (unlink($filename)) {
1825     echo "File $filename has been deleted.";
1826 } else {
1827     echo "Error deleting the file.";
1828 }
1829 ?>
1830
1831 <!-- 6. Directory Listing: -->
1832 <?php
1833 // Listing files in a directory
1834 $directory = "path/to/files/";
1835
1836 $files = scandir($directory);
1837
1838 foreach ($files as $file) {
1839     if ($file != "." && $file != "..") {
1840         echo $file . "<br>";
1841     }
1842 }
```

```

1843     ?>
1844
1845 <!-- 7. Renaming a File: -->
1846 <?php
1847     // Renaming a file
1848     $old_name = "oldfile.txt";
1849     $new_name = "newfile.txt";
1850
1851     if (rename($old_name, $new_name)) {
1852         echo "File successfully renamed.";
1853     } else {
1854         echo "Error renaming the file.";
1855     }
1856     ?>
1857
1858 <!-- Handling File Paths: -->
1859 <?php
1860     // Working with file paths
1861     $absolute_path = realpath("file.txt");
1862     $directory = dirname("/path/to/file.txt");
1863
1864     echo "Absolute Path: $absolute_path<br>";
1865     echo "Directory: $directory";
1866     ?>
1867
1868 <!-- *poo --> -----
1869
1870 <!-- create a class -->
1871 <?
1872     class fruits{
1873 // <!-- default public -->
1874     public $name;
1875     public $colors;
1876
1877     public function setName($name){
1878         $this->name=$name;
1879     }
1880     public function getName(){
1881
1882         return $this->name;
1883     }
1884     }
1885     ?>
1886
1887 <!-- constructor : -->
1888 <?php
1889     class ClassName{
1890     public function __construct(){
1891         // code
1892     }
1893
1894 // !-- destructor -->
1895     public function __destruct(){
1896
1897         // code
1898     }

```

```

1899     };
1900     ?>
1901
1902     <!-- create an instance from a class -->
1903     <?php
1904         $apple = new fruits();
1905         $apple->setName("apple");
1906         $apple->getName();
1907     ?>
1908
1909     <!-- declare a static variable -->
1910     <?php
1911         class className{
1912             public static int $ObjectCounter = 0;
1913         };
1914     ?>
1915
1916     <!-- access to a static variable into the class -->
1917     <?php
1918         self::$StaticVarName++;
1919     ?>
1920     <!-- access to static var outside the class -->
1921     <?php
1922         className::$staticVarName;
1923     ?>
1924
1925     <!-- inheritance -->
1926     <?php
1927         class Parent{
1928
1929             };
1930
1931         class child extends Parent{
1932
1933             };
1934     ?>
1935     <!-- example 1 -->
1936     <?php
1937         class fruit
1938         {
1939             public $name;
1940             public $color;
1941
1942             public function getName()
1943             {
1944                 return $this->name;
1945             }
1946             public function getColor()
1947             {
1948                 return $this->color;
1949             }
1950             public function setName($name)
1951             {
1952                 $this->name = $name;
1953             }
1954             public function setColor($color)

```

```

1955         {
1956             $this->color = $color;
1957         }
1958     };
1959     $mongo = new fruit();
1960     $mongo->setColor("red");
1961     $mongo->setName("mongo");
1962
1963     echo "the name is : " . $mongo->getName() . "\n <br>";
1964     echo "the color is : " . $mongo->getColor() . "\n <br>";
1965     ?>
1966
1967
1968 <!-- using strict type -->
1969 <?php
1970     declare(strict_types=1);
1971     ?>
1972
1973 <!-- example 2 -->
1974 <?php
1975     class Pont {
1976         public float $width;
1977         public float $height;
1978         public static int $ObjectCounter = 0;
1979
1980         // increment constructor
1981         public function __construct() {
1982             // Increment the static counter when an object is created
1983             self::$ObjectCounter++;
1984         }
1985     }
1986
1987     // Creating an instance of the class
1988     $p1 = new Pont();
1989     $p1 = new Pont();
1990     $p1 = new Pont();
1991     // Displaying the number of objects
1992     echo "The number of objects in the class: " . Pont::$ObjectCounter;
1993     ?>
1994

```