



# PHP LOOPS & FUNCTION

---

**COURSE INSTRUCTOR**

**ABDULLAH AL NOMAN PRINCE**

**SOFTWARE ENGINEER**

**SUBJECT MATTER EXPERT & SUPPORT INSTRUCTOR, OSTAD**

# TOPICS WILL BE COVERD

---

**CONCEPT OF LOOPS**

**CONCEPT OF FUNCTION**

**SOME LOOP & FUNCTION EXERCISE**

# POST-INCREMENT & PRE-INCREMENT

```
1 <?php
2
3 $x = 1;
4 $y = 1;
5
6 echo $x . PHP_EOL;
7 echo $x++ . PHP_EOL;
8 echo $x . PHP_EOL;
9 echo $y . PHP_EOL;
10 echo ++$y . PHP_EOL;
11 echo $y . PHP_EOL;
```

```
1 1
2 1
3 2
4 1
5 2
6 2
```

# POST-DECREMENT & PRE-DECREMENT

```
1 <?php
2
3 $x = 1;
4 $y = 1;
5
6 echo $x . PHP_EOL;
7 echo $x-- . PHP_EOL;
8 echo $x . PHP_EOL;
9 echo $y . PHP_EOL;
10 echo --$y . PHP_EOL;
11 echo $y . PHP_EOL;
```

```
1 1
2 1
3 0
4 1
5 0
6 0
```

# **WHAT IS LOOP**

**Loops are used to execute the same block of code again and again, as long as a certain condition is true.**

## **PHP LOOPS**

- ➡ while**
- ➡ do while**
- ➡ for**
- ➡ foreach**

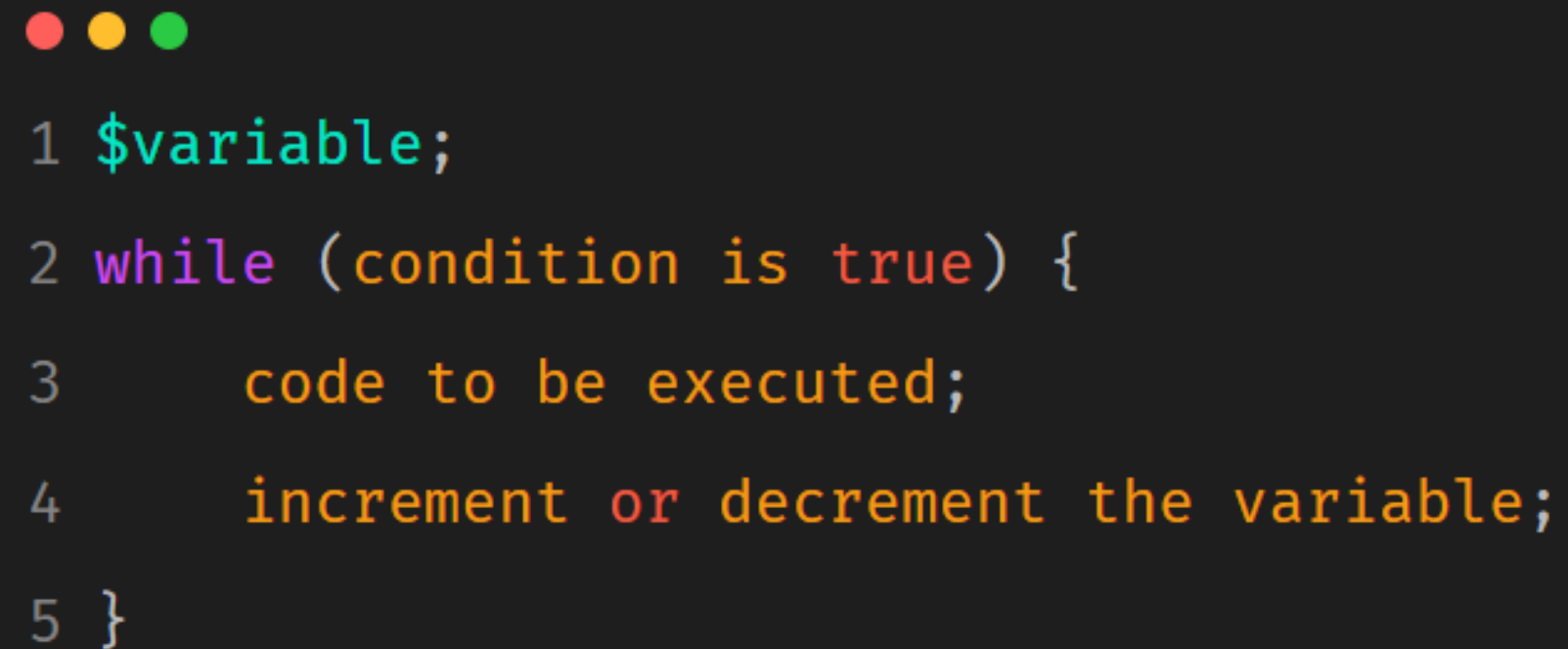
# **LOOP PARAMETERS**

## **Parameters**

- ➡ **variable initialization**
- ➡ **condition**
- ➡ **change (increment/decrement)**

# WHILE LOOP

**The while loop executes a block of code as long as the specified condition is true.**



```
1 $variable;  
2 while (condition is true) {  
3     code to be executed;  
4     increment or decrement the variable;  
5 }
```

# WHILE LOOP EXAMPLE

**Increment way:**

```
● ● ●  
1 $x = 1;  
2 while ( $x ≤ 5 ) {  
3     echo "The number is:" . $x . "\n";  
4     $x++;  
5 }
```



# WHILE LOOP EXAMPLE

**Decrement way:**

```
● ● ●
1 $x = 5;
2 while ( $x ≥ 1 ) {
3     echo "The number is:" . $x . "\n";
4     $x--;
5 }
```

# DO WHILE LOOP

---

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

```
1 $variable;  
2 do {  
3     code to be executed;  
4     increment or decrement the variable;  
5 } while (condition is true);
```

# DO WHILE LOOP

**First do then check condition**

```
● ● ●  
1 $x = 6;  
2 do {  
3     echo "The number is: $x \n";  
4     $x++;  
5 } while ( $x ≤ 5 );
```

# DO WHILE LOOP EXAMPLE

**Increment way:**

```
1 $x = 1;  
2 do {  
3     echo "The number is: $x \n";  
4     $x++;  
5 } while ( $x ≤ 5 );
```


# DO WHILE LOOP EXAMPLE

**Decrement way:**

```
1 $x = 5;  
2 do {  
3     echo "The number is: $x \n";  
4     $x--;  
5 } while ( $x ≥ 1 );
```

# FOR LOOP(IMPORTANT)

**The for loop is used when you know in advance how many times the script should run.**



```
1 $counter;  
2 for (init counter; test counter; increment counter) {  
3     code to be executed for each iteration;  
4 }
```

# FOR LOOP EXAMPLE

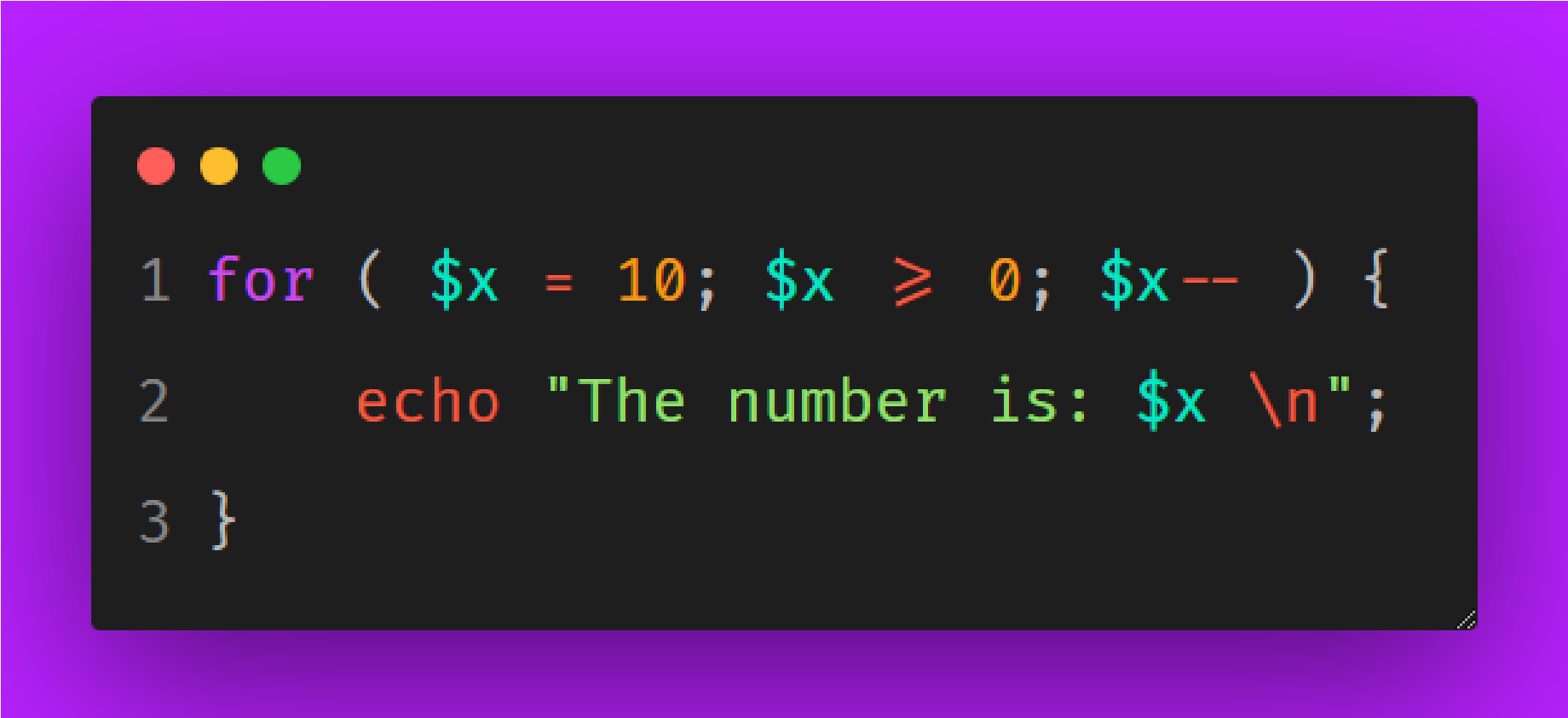
**Increment way:**



```
1 for ( $x = 0; $x ≤ 10; $x++ ) {  
2     echo "The number is: $x \n";  
3 }
```

# FOR LOOP EXAMPLE

**Decrement way:**



```
1 for ( $x = 10; $x ≥ 0; $x-- ) {  
2     echo "The number is: $x \n";  
3 }
```



# FOR LOOP EXAMPLE

## Nested Way:

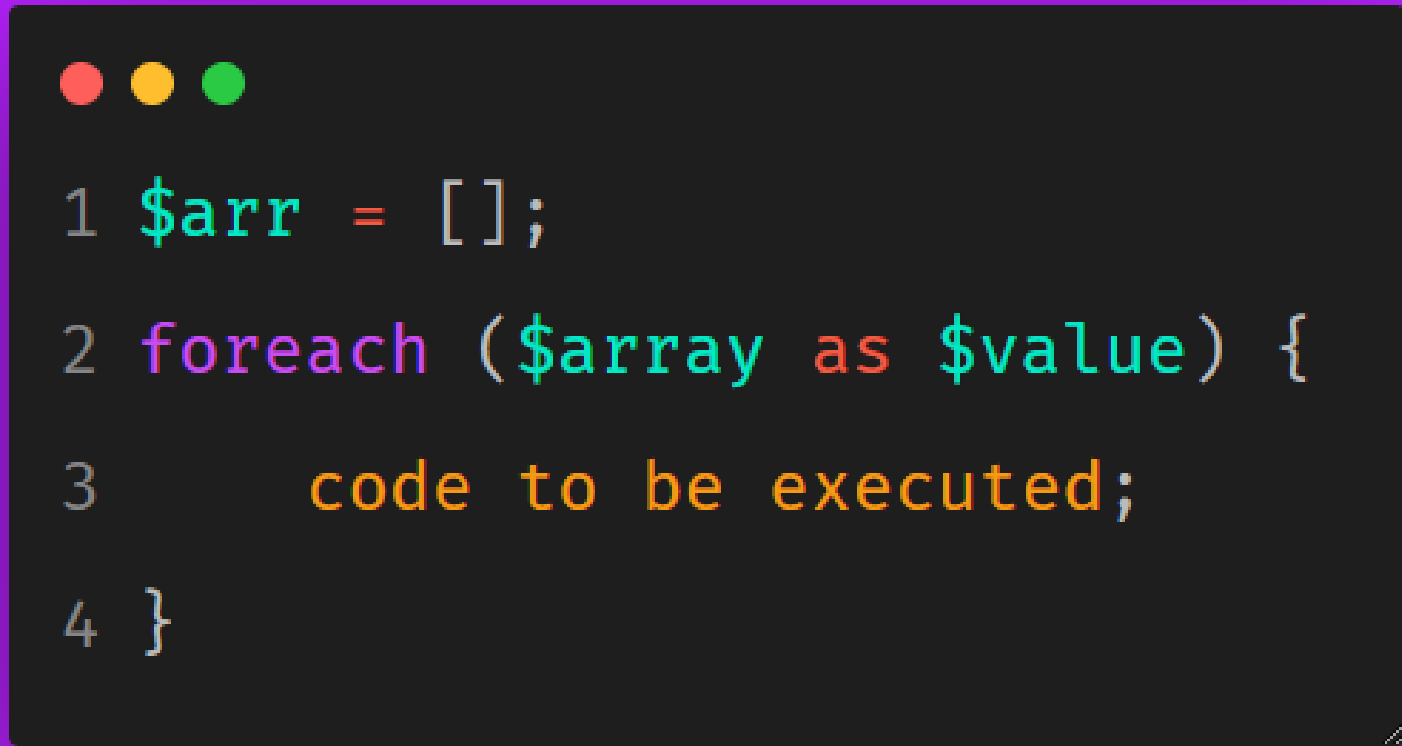
```
1 for ( $i = 1; $i ≤ 2; $i++ ) {  
2     echo "Times Table of : $i \n";  
3     for ( $j = 1; $j ≤ 10; $j++ ) {  
4         $product = $i * $j;  
5         echo "$i x $j = $product\n";  
6     }  
7     echo "\n";  
8 }
```

```
1 Times Table of : 1  
2 1 x 1 = 1  
3 1 x 2 = 2  
4 1 x 3 = 3  
5 1 x 4 = 4  
6 1 x 5 = 5  
7 1 x 6 = 6  
8 1 x 7 = 7  
9 1 x 8 = 8  
10 1 x 9 = 9  
11 1 x 10 = 10
```

```
13 Times Table of : 2  
14 2 x 1 = 2  
15 2 x 2 = 4  
16 2 x 3 = 6  
17 2 x 4 = 8  
18 2 x 5 = 10  
19 2 x 6 = 12  
20 2 x 7 = 14  
21 2 x 8 = 16  
22 2 x 9 = 18  
23 2 x 10 = 20
```

# FOREACH LOOP(IMPORTANT)

The foreach loop works only on arrays, and is used to loop through each key/value pair in an **array**.



```
1 $arr = [];  
2 foreach ($array as $value) {  
3     code to be executed;  
4 }
```

# FOREACH LOOP EXAMPLE



```
1 $colors = array( "red", "green", "blue", "yellow" );  
2  
3 foreach ( $colors as $value ) {  
4     echo "$value \n";  
5 }
```

# PHP BREAK

---

The break statement is used to jump out of a loop.

```
1 for ( $x = 0; $x < 10; $x++ ) {  
2     if ( $x == 4 ) {  
3         break; // Stop the loop  
4     }  
5     echo "The number is: $x \n";  
6 }
```

```
1 The number is: 0  
2 The number is: 1  
3 The number is: 2  
4 The number is: 3
```

# PHP CONTINUE

**The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.**

```
1 for ( $x = 0; $x < 10; $x++ ) {  
2     if ( $x == 4 ) {  
3         continue; // Skip to the next iteration  
4     }  
5     echo "The number is: $x \n";  
6 }
```

```
1 The number is: 0  
2 The number is: 1  
3 The number is: 2  
4 The number is: 3  
5 The number is: 5  
6 The number is: 6  
7 The number is: 7  
8 The number is: 8  
9 The number is: 9
```

# **LOOP'S EXERCISE**

## **=> Easy**

**Exercise 1: Print the Even Number from 1 to 50**

**Exercise 2: Print the Odd Number from 1 to 50**

**Exercise 3: Print the Sum of Even Numbers from 1 to 50**

**Exercise 4: Print the Sum of Odd Numbers from 1 to 50**

**Exercise 5: Print the Sum of Numbers from 1 to 50**

**Exercise 6: Print the Square of Numbers from 1 to 10**

## **=> Medium**

**Exercise 7: Count Digits in a Number (Like \$numbers = 12345)**

**Exercise 8: Multiplication Table of 1 to 10;**

**Exercise 9: Reverse a Given String (\$string = "I love PHP"); [For Module 2, You can skip this problem]**

## **=> Hard**

**Exercise 10: Fibonacci Sequence of First 10 Numbers.**

**Exercise 11: Factorial Calculation of a Given Number.**

# **FUNCTION**

- **A function is a block of statements that can be used repeatedly in a program.**
- **A function will not execute automatically when a page loads.**
- **A function will be executed by a call to the function.**

## **1. Built-in Functions**

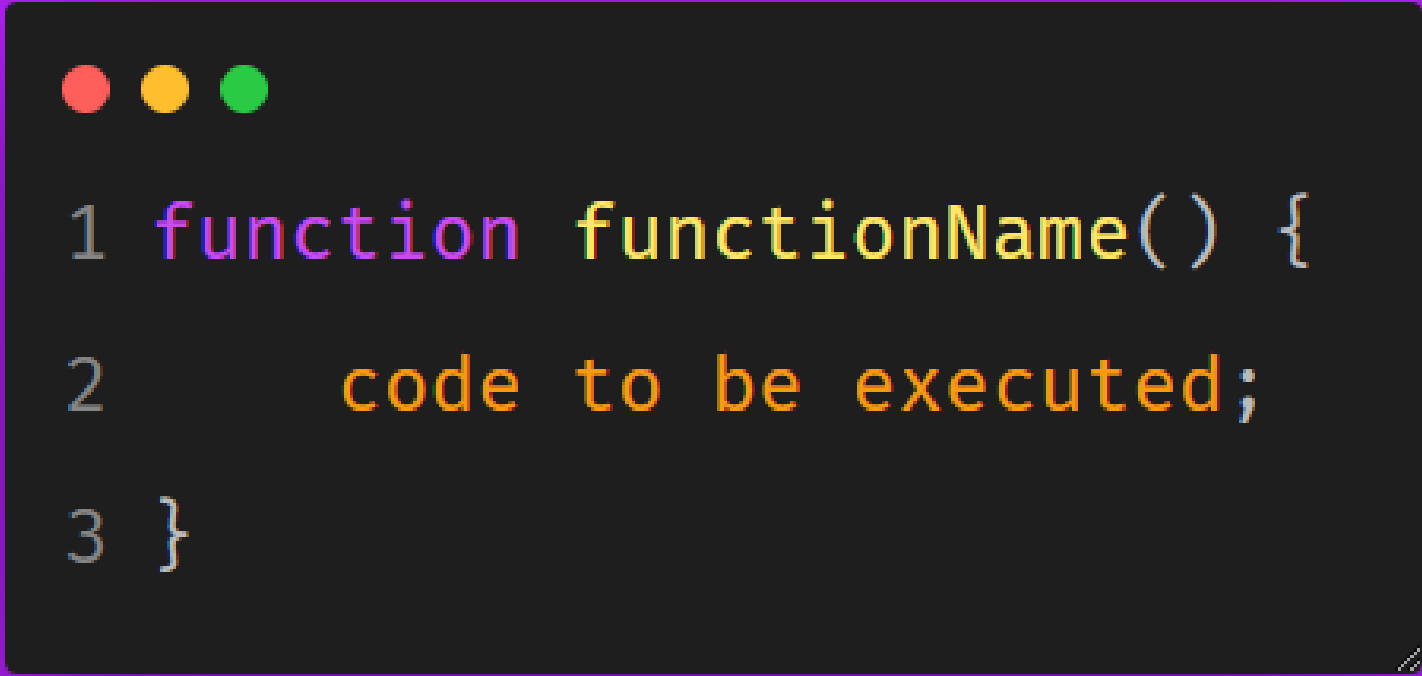
([https://www.w3schools.com/php/php\\_ref\\_overview.asp](https://www.w3schools.com/php/php_ref_overview.asp))

## **1. User Defined Functions**

# FUNCTION

## Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:



```
1 function functionName() {  
2     code to be executed;  
3 }
```



# FUNCTION

- A function name must start with a letter or an underscore.
- Function names are NOT case-sensitive.

```
1 function greetingMsg()  
2 {  
3     echo "Hello world!";  
4 }  
5 // Both are treated as same  
6 greetingMsg(); // call the function  
7 GreetingMsg(); // call the function
```

# PHP FUNCTION ARGUMENTS & PARAMETERS

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are passed while calling the function
- Parameters are passed while defining the function
- Here argument is “Jeny” & “Robert”
- Here parameter is `$fname`

```
1 function familyName( $fname )
2 {
3     echo "$fname Colins.\n";
4 }
5
6 familyName( "Jeny" );
7 familyName( "Robert" );
```

# PHP FUNCTION MULTIPLE ARGUMENTS & PARAMETERS

```
1 function familyName( $fname, $year )
2 {
3     echo "$fname Refsnes. Born in $year \n";
4 }
5
6 familyName( "Hege", "1975" );
7 familyName( "Stale", "1978" );
8 familyName( "Kai Jim", "1983" );
9
```

# PHP FUNCTION DEFAULT VALUE OF ARGUMENTS

```
1 function setHeight( $minheight = 50 )
2 {
3     echo "The height is : $minheight \n";
4 }
5
6 setHeight( 350 );
7 setHeight(); // will use the default value of 50
8 setHeight( 135 );
```

# TYPE HINTING ARGUMENTS



```
1 function addNumbers( int $a, int $b )  
2 {  
3     return $a + $b;  
4 }  
5 echo addNumbers( 5, 4 );
```

# TYPE HINTING ARGUMENTS

---

- By declaring strict declaration
- To specify strict we need to set `declare(strict_types=1);`.
- This must be on the very first line of the PHP file.

```
1 <?php declare(strict_types=1); // strict requirement
2
3 function addNumbers(int $a, int $b) {
4     return $a + $b;
5 }
6 echo addNumbers(5, "5 days");
7 // since strict is enabled and "5 days" is not an integer, an error will be thrown
8 ?>
```

# PHP FUNCTIONS – RETURNING VALUES

- To let a function return a value, use the return statement

```
1 function sum( int $x, int $y )  
2 {  
3     $z = $x + $y;  
4     return $z;  
5 }  
6  
7 echo "5 + 10 = " . sum( 5, 10 );
```

# PHP FUNCTIONS – RETURNING VALUES

- To let a function return a value, use the return statement

```
1 function sum( int $x, int $y )  
2 {  
3     $z = $x + $y;  
4     return $z;  
5 }  
6  
7 echo "5 + 10 = " . sum( 5, 10 );
```



# PHP RETURN TYPE DECLARATIONS

---

- To declare a type for the function return, add a colon (:) and the type right before the opening curly ( { ) bracket when declaring the function.

```
1 function addNumbers( float $a, float $b ): float
2 {
3     return $a + $b;
4 }
5 echo addNumbers( 1.2, 5.2 );
```

# PHP RETURN TYPE DECLARATIONS

---

- You can use type hinting while returning.

```
1 function addNumbers(float $a, float $b) : float {  
2     return (float)($a + $b);  
3 }  
4 echo addNumbers(1.2, 5.2);
```

# PHP MULTIPLE TYPE HINTING & RETURN TYPE DECLARATIONS

---

```
1 <?php
2 declare ( strict_types = 1 );
3 function myFunc( int | float $n1, int | float $n2 ): int | float
4 {
5     return $n1 / $n2;
6 }
7 echo myFunc( 5, 2.2 );
```

# VOID RETURN TYPE

---

```
1 function myFunc( int $x, int $y ): void
2 {
3     global $sum;
4     $sum = $x + $y;
5 }
6 myFunc( 5, 2 );
7 echo $sum;
```

# PHP ANONYMOUS FUNCTIONS

---



```
1 $myFunction = function ( $x ) {  
2     return $x;  
3 };  
4  
5 echo $myFunction( 5 );
```

# PHP ACCESSING GLOBAL VARIABLE WITHIN FUNCTION

---

```
1 $x = 1;
2 function myFunc( int $y ): int
3 {
4     global $x;
5     return $x + $y;
6 }
7
8 echo myFunc( 2 ); // Outputs 3 (1 + 2)
```

```
1 $outerVar = 10;
2
3 $myFunction = function ( $x ) use ( $outerVar ) {
4     return $x + $outerVar;
5 };
6
7 $result = $myFunction( 5 ); // Calls the anonymous function
8 echo $result; // Outputs 15 (5 + 10)
```

# PASSING ARGUMENTS BY REFERENCE

---

- In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.
- When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used:

```
1 function add_five( &$value )  
2 {  
3     $value += 5;  
4 }  
5  
6 $num = 2;  
7 add_five( $num );  
8 echo $num;
```

# RECURSION IN PHP

```
1 function display( $number )
2 {
3     if ( $number ≤ 5 ) {
4         echo "$number \n";
5         display( $number + 1 );
6     }
7 }
8 display( 1 );
```

```
1 1
2 2
3 3
4 4
5 5
```



# **FUNCTION'S EXERCISE**

- **Will be added after the module 2, conceptual class 2.**

**THANK YOU**

---