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

```php
<?php

$x = 1;
$y = 1;

echo $x . PHP_EOL;
echo $x++ . PHP_EOL;
echo $x . PHP_EOL;
echo $y . PHP_EOL;
echo ++$y . PHP_EOL;
echo $y . PHP_EOL;
```

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

POST-DECREMENT & PRE-DECREMENT

```php
<?php

$x = 1;
$y = 1;

echo $x . PHP_EOL;
echo $x-- . PHP_EOL;
echo $x . PHP_EOL;
echo $y . PHP_EOL;
echo --$y . PHP_EOL;
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**

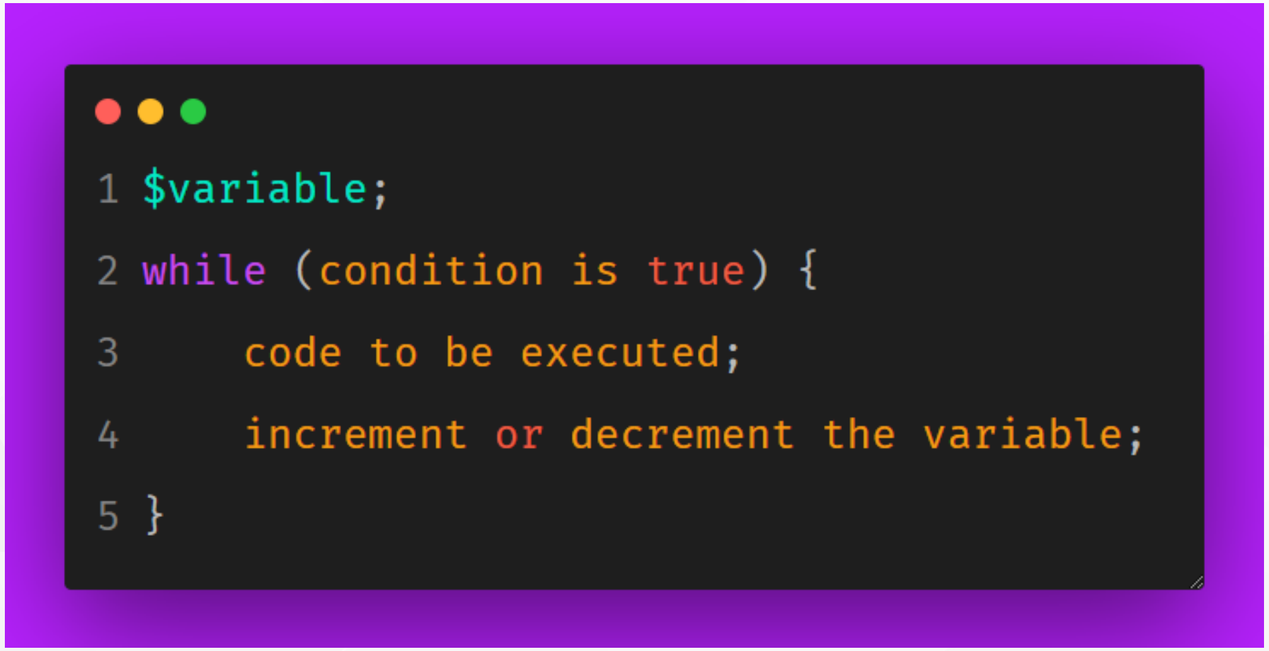
LOOP PARAMETERS

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.



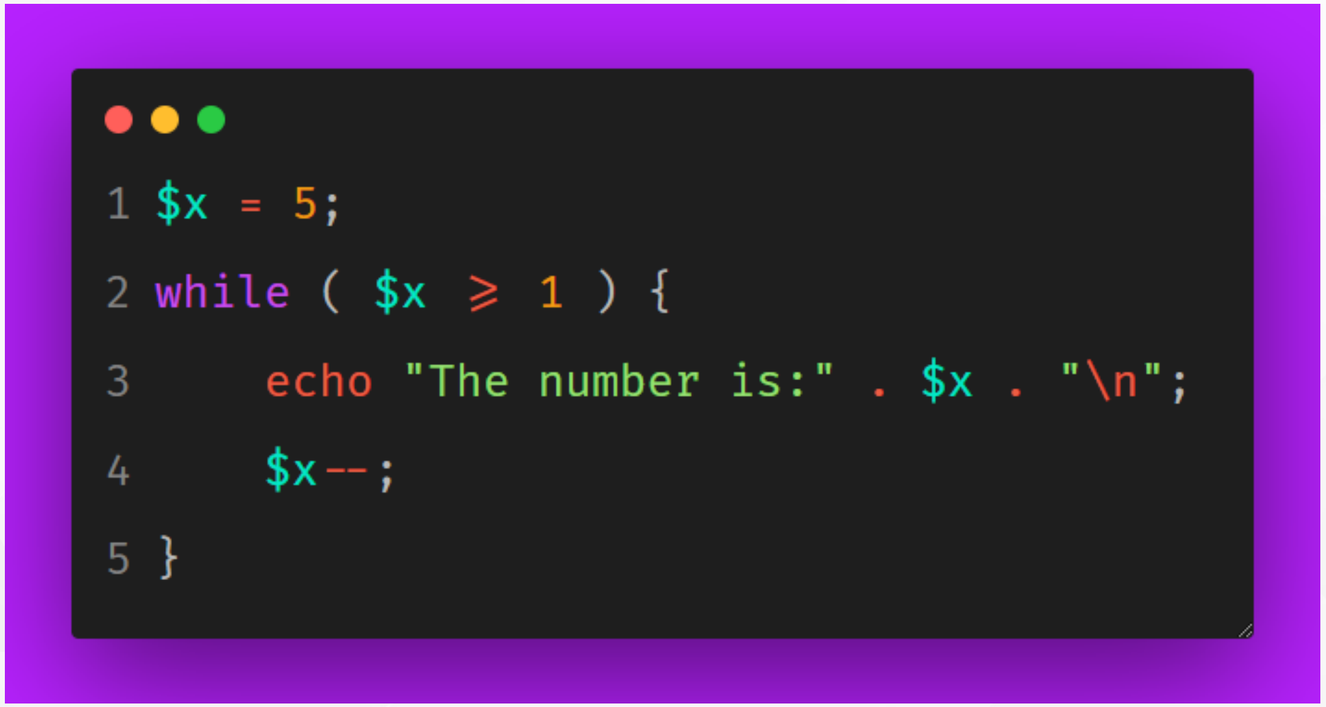
WHILE LOOP EXAMPLE

Increment way:

```php
$x = 1;
while ( $x <= 5 ) {
    echo "The number is:" . $x . "\n";
    $x++;
}
```

WHILE LOOP EXAMPLE

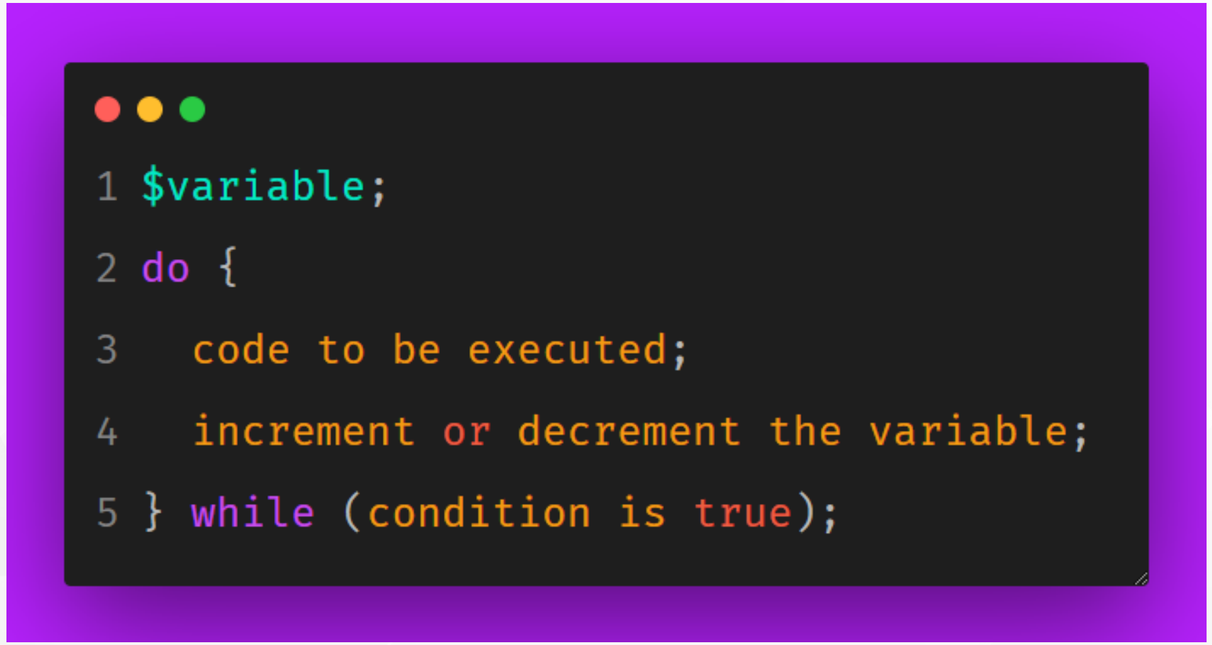
Decrement way:



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

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.



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

DO WHILE LOOP

First do then check condition

```php
$x = 6;
do {
    echo "The number is: $x \n";
    $x++;
} while ( $x ⩽ 5 );
```

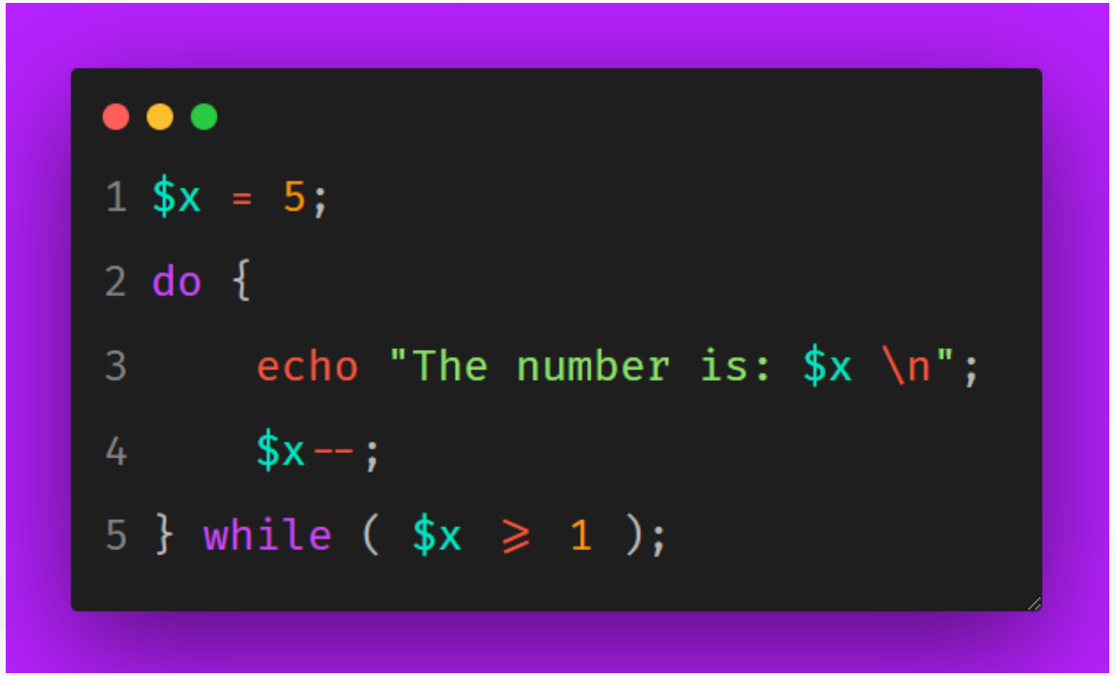
DO WHILE LOOP EXAMPLE

Increment way:

```php
$x = 1;
do {
    echo "The number is: $x \n";
    $x++;
} while ( $x <= 5 );
```

DO WHILE LOOP EXAMPLE

Decrement way:



```php
$x = 5;
do {
    echo "The number is: $x \n";
    $x--;
} while ( $x ⩾ 1 );
```

FOR LOOP(IMPORTANT)

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

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

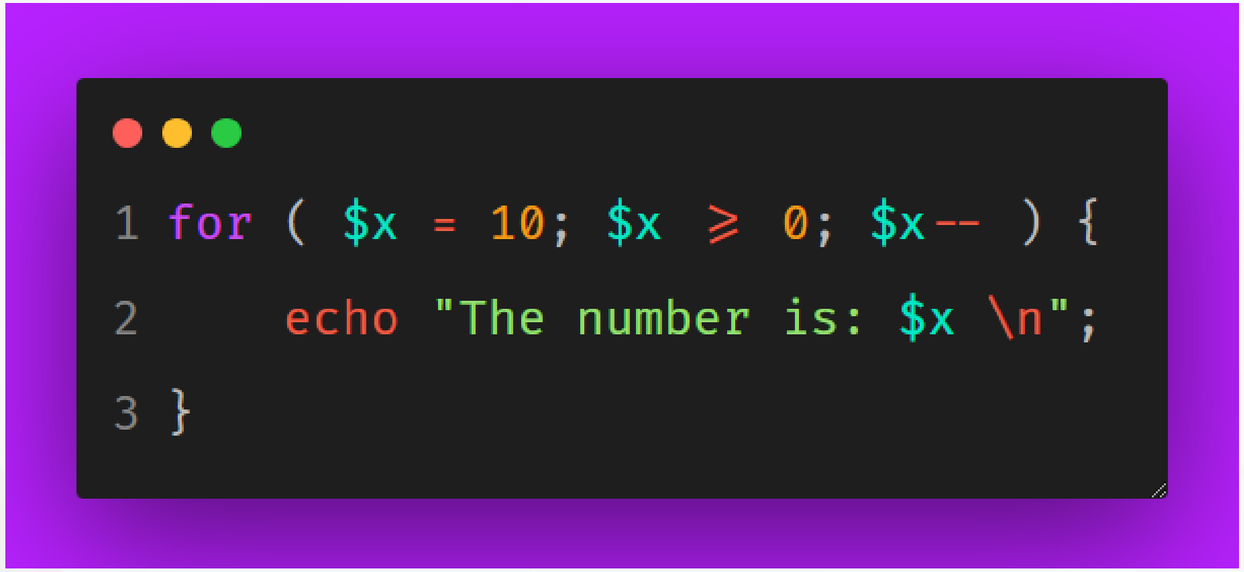
FOR LOOP EXAMPLE

Increment way:

```php
for ( $x = 0; $x <= 10; $x++ ) {
    echo "The number is: $x \n";
}
```

FOR LOOP EXAMPLE

Decrement way:



```php
for ( $x = 10; $x ⩾ 0; $x-- ) {
    echo "The number is: $x \n";
}
```

FOR LOOP EXAMPLE

Nested Way:

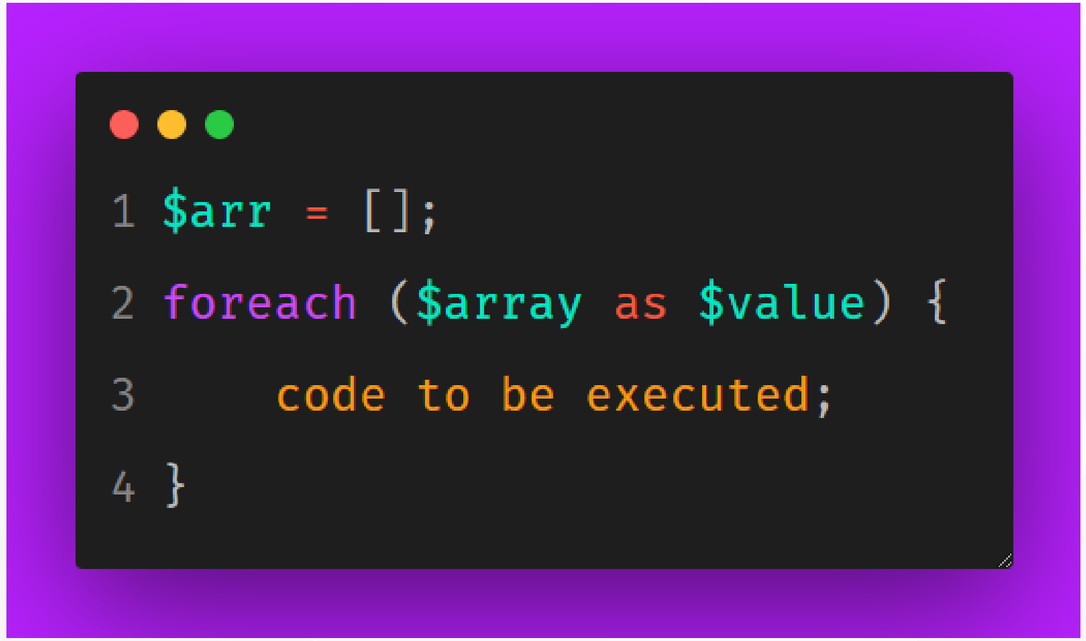
```php
for ( $i = 1; $i <= 2; $i++ ) {
    echo "Times Table of : $i \n";
    for ( $j = 1; $j <= 10; $j++ ) {
        $product = $i * $j;
        echo "$i x $j = $product\n";
    }
    echo "\n";
}
```

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

```
Times Table of : 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
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.



```php
$arr = [];
foreach ($array as $value) {
    code to be executed;
}
```

FOREACH LOOP EXAMPLE

```php
$colors = array( "red", "green", "blue", "yellow" );

foreach ( $colors as $value ) {
    echo "$value \n";
}
```

PHP BREAK

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

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

```
The number is: 0
The number is: 1
The number is: 2
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.

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

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 5
The number is: 6
The number is: 7
The number is: 8
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");

=> 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**

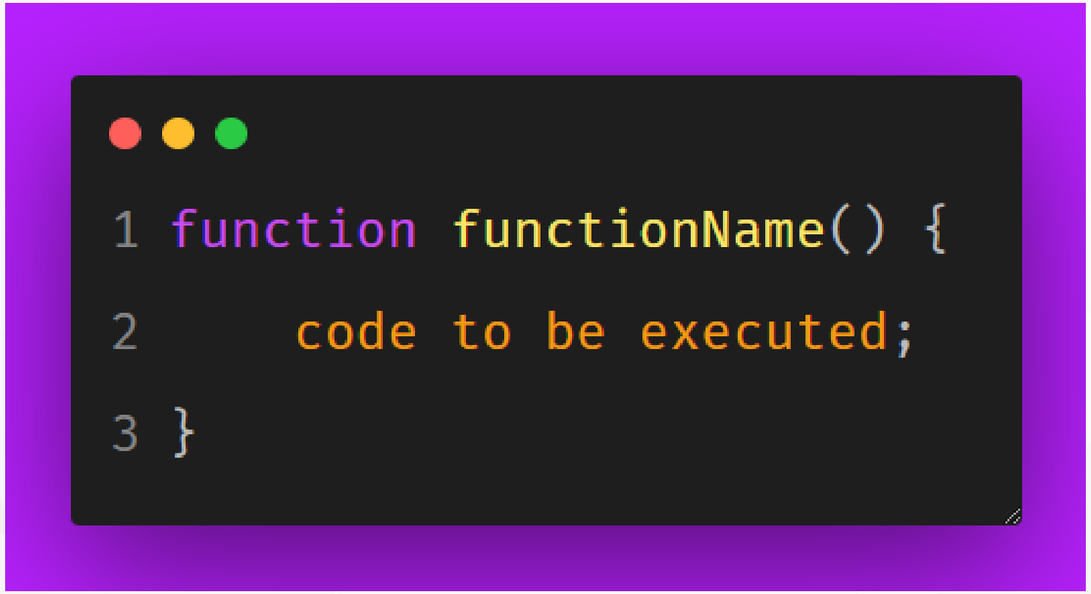
(https://www.w3schools.com/php/php_ref_overview.asp**)**

1. **User Defined Functions**

FUNCTION

Create a User Defined Function in PHP

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



FUNCTION

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

```php
function greetingMsg()
{
    echo "Hello world!";
}
// Both are treated as same
greetingMsg(); // call the function
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**

```php
function familyName( $fname )
{
    echo "$fname Colins.\n";
}

familyName( "Jeny" );
familyName( "Robert" );
```

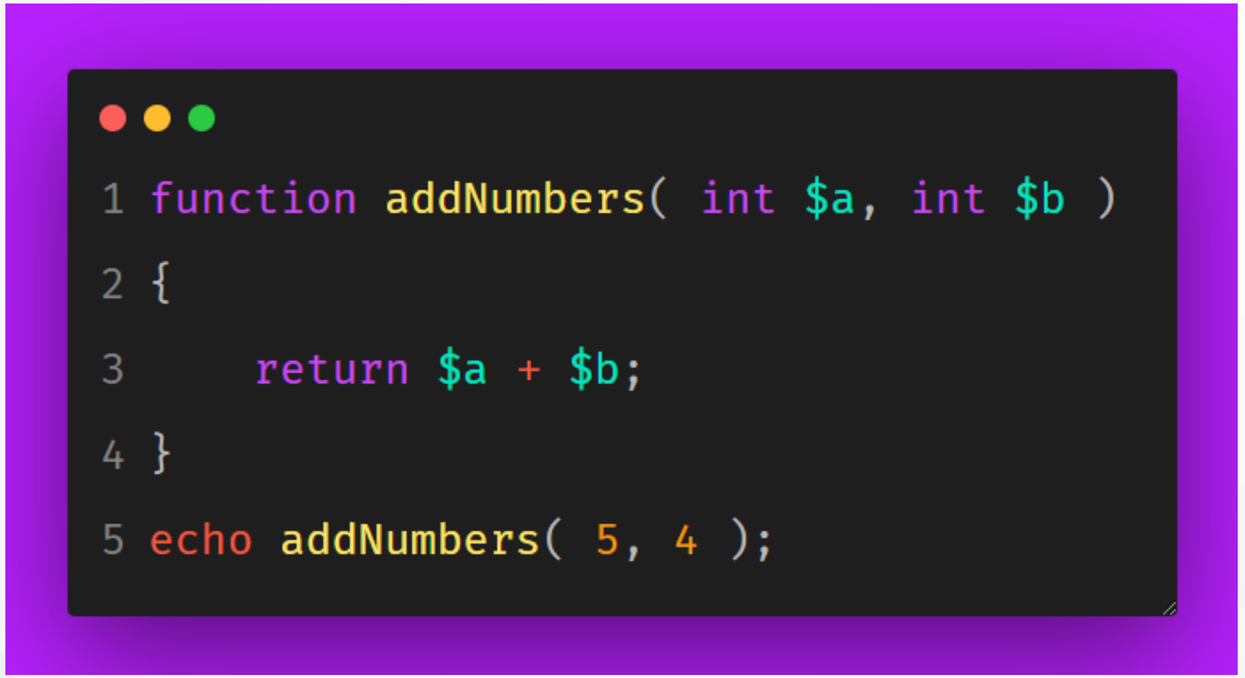
PHP FUNCTION MULTIPLE ARGUMENTS & PARAMETERS

```php
function familyName( $fname, $year )
{
    echo "$fname Refsnes. Born in $year \n";
}

familyName( "Hege", "1975" );
familyName( "Stale", "1978" );
familyName( "Kai Jim", "1983" );

```

PHP FUNCTION DEFAULT VALUE OF ARGUMENTS

```php
function setHeight( $minheight = 50 )
{
    echo "The height is : $minheight \n";
}

setHeight( 350 );
setHeight(); // will use the default value of 50
setHeight( 135 );
```

TYPE HINTING ARGUMENTS



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

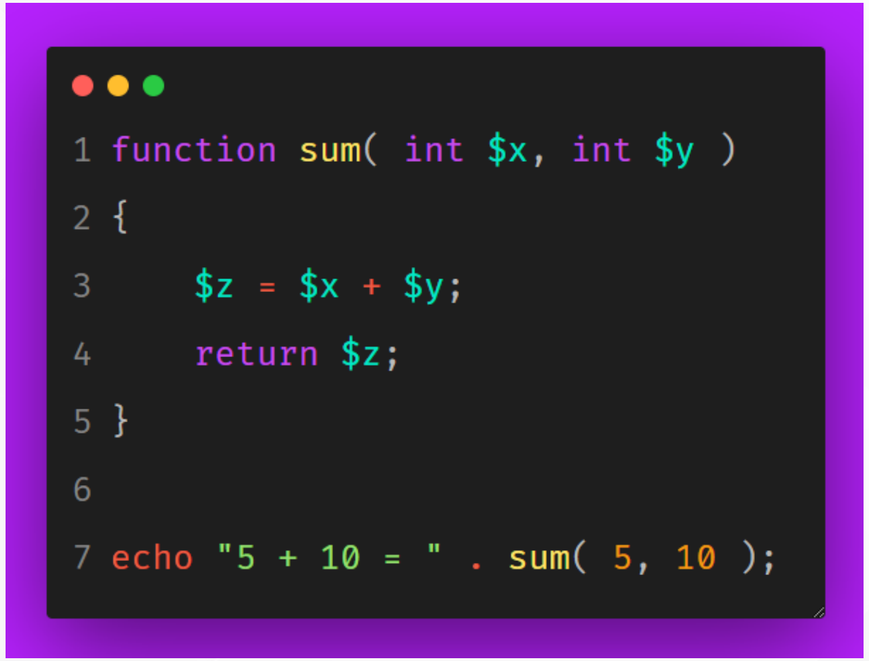
TYPE HINTING ARGUMENTS

- **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.**

```php
<?php declare(strict_types=1); // strict requirement

function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```

PHP FUNCTIONS – RETURNING VALUES

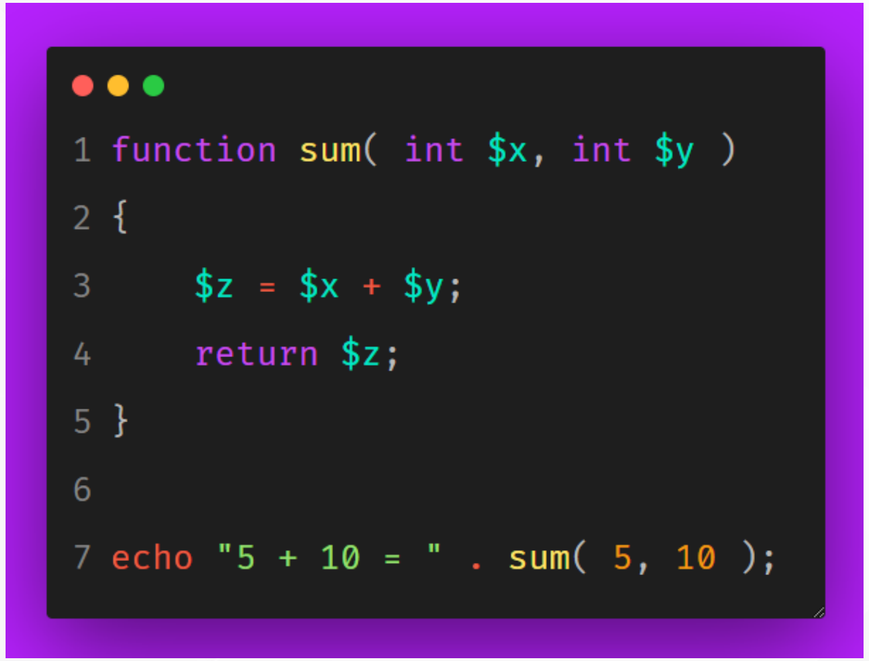
- **To let a function return a value, use the return statement**



```php
function sum( int $x, int $y )
{
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum( 5, 10 );
```

PHP FUNCTIONS – RETURNING VALUES

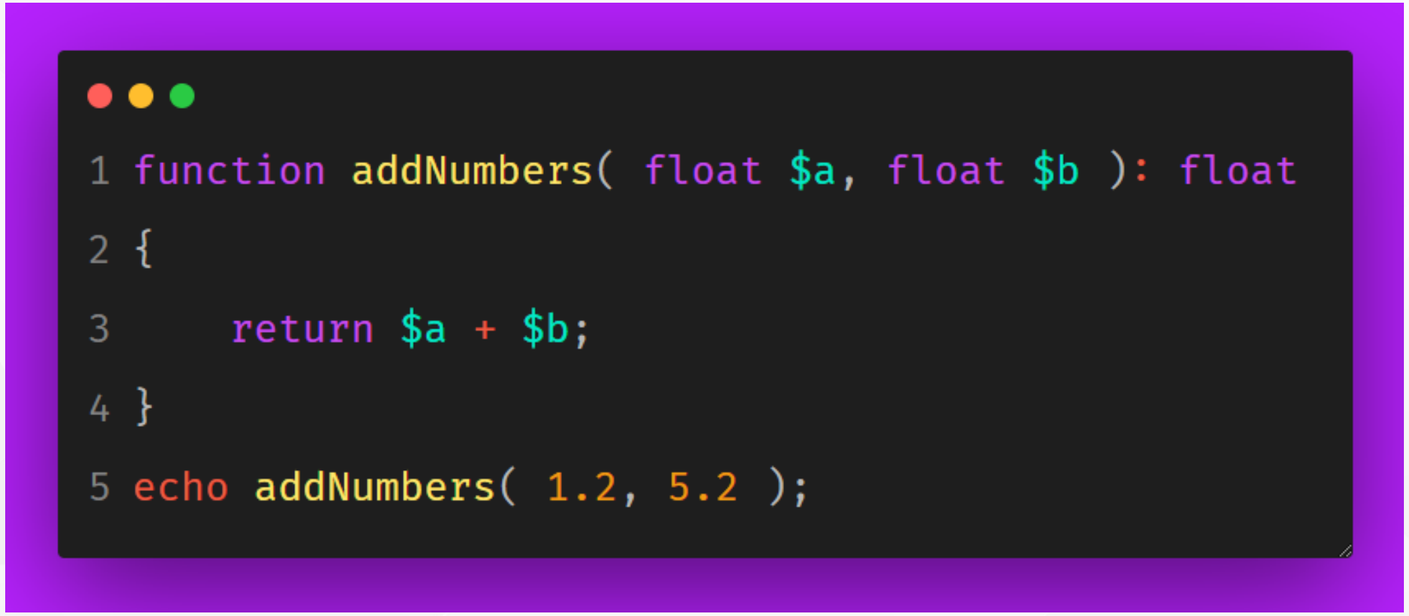
- **To let a function return a value, use the return statement**



```php
function sum( int $x, int $y )
{
    $z = $x + $y;
    return $z;
}

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.**



```php
function addNumbers( float $a, float $b ): float
{
    return $a + $b;
}
echo addNumbers( 1.2, 5.2 );
```

PHP RETURN TYPE DECLARATIONS

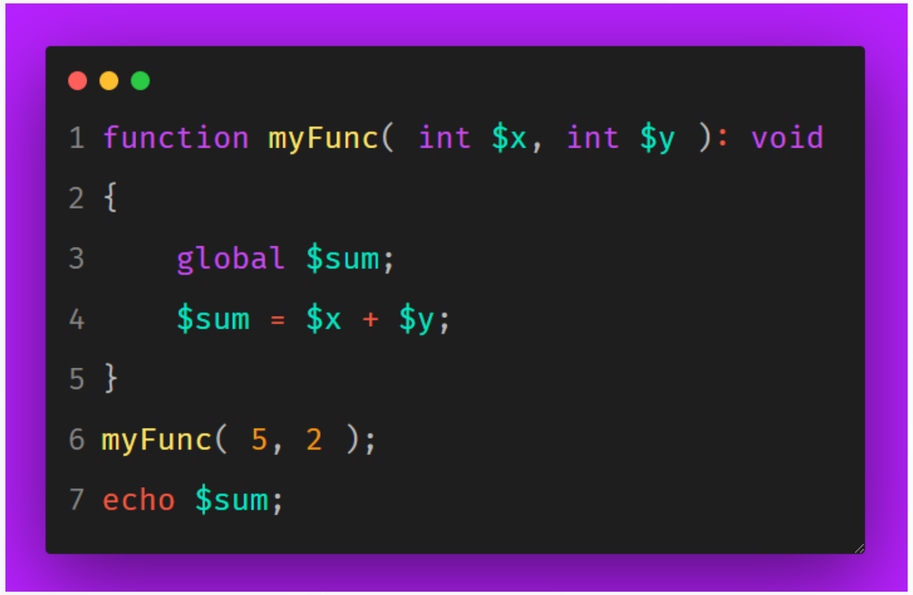
- **You can use type hinting while returning.**

```php
function addNumbers(float $a, float $b) : float {
    return (float)($a + $b);
}
echo addNumbers(1.2, 5.2);
```

PHP MULTIPLE TYPE HINTING & RETURN TYPE DECLARATIONS

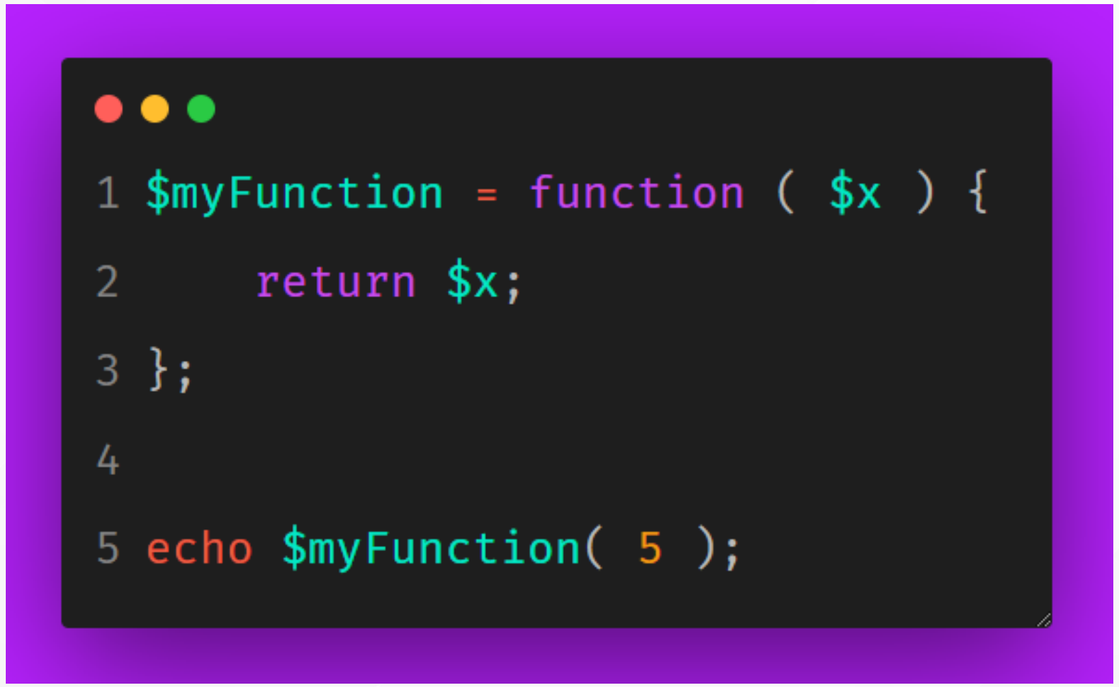
```php
<?php
declare ( strict_types = 1 );
function myFunc( int | float $n1, int | float $n2 ): int | float
{
    return $n1 / $n2;
}
echo myFunc( 5, 2.2 );
```

VOID RETURN TYPE



```php
function myFunc( int $x, int $y ): void
{
    global $sum;
    $sum = $x + $y;
}
myFunc( 5, 2 );
echo $sum;
```

PHP ANONYMOUS FUNCTIONS



```php
$myFunction = function ( $x ) {
    return $x;
};

echo $myFunction( 5 );
```

PHP ACCESSING GLOBAL VARIABLE WITHIN FUNCTION

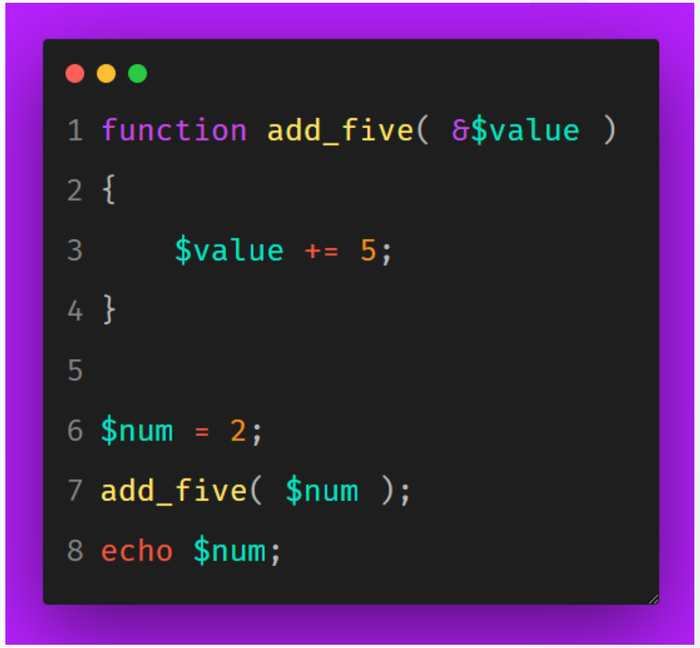
```php
$x = 1;
function myFunc( int $y ): int
{
    global $x;
    return $x + $y;
}

echo myFunc( 2 ); // Outputs 3 (1 + 2)
```

```php
$outerVar = 10;

$myFunction = function ( $x ) use ( $outerVar ) {
    return $x + $outerVar;
};

$result = $myFunction( 5 ); // Calls the anonymous function
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:



```php
function add_five( &$value )
{
    $value += 5;
}

$num = 2;
add_five( $num );
echo $num;
```

RECURSION IN PHP

```php
function display( $number )
{
    if ( $number ⩽ 5 ) {
        echo "$number \n";
        display( $number + 1 );
    }
}
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