Code

Printing

```
echo "<h1>Hello World</h1>");
print "<hr/>";                                  // returns 1
echo  "<p>This is a php tutorial</p>";
```

Variables and Data Types

```
/*
Names are case-sensitive and must start with '$' then:
   letters, _
After, may include
   letters, numbers, _
Convention says
   Start with a lowercase word, then additional words are
capitilized
   ex. myFirstVariable
*/
$name = "Mike";     // Strings
$age = 30;          // Integer
$gpa = 3.5;         // Decimal
$isTall = true;     // Boolean -> true/false

$name = "John";

echo  "Your name is $name <br>";
```

Casting and Converting

```
echo ((int)3.14)."<br>";
echo ((float)3)."<br>";
echo ((int)"80" + (float)"60.5")."<br>";
echo  intval("80") + floatval("60.5");
```

Strings

```
$greeting = "Hello";
//indexes:    01234
```

```
echo  strlen($greeting)."<br>";
echo  $greeting[0]."<br>";
echo  $greeting[-1]."<br>";
echo  str_replace("l", "Z", $greeting)."<br>";
echo  strchr($greeting, "ll")."<br>";
```

Numbers

```
echo (2 * 3)."<br>";                    // Basic Arithmetic: +, -
, /, *
echo (2**3)."<br>";                     // Basic Arithmetic: +, -
, /, *
echo (10 % 3)."<br>";                   // Modulus Op. : returns
remainder of 10/3
echo (1 + 2 * 3)."<br>";                // order of operations
echo (10 / 3.0)."<br><br>";             // int's and doubles
```

```
$num = 10;
$num += 100;                            // +=, -=, /=, *=
echo $num."<br>";
```

```
$num++;
echo $num."<br><br>";
```

```
// useful math methods
echo max(2, 3)."<br>";
echo sqrt(144)."<br>";
echo round(2.7)."<br>";
```

User Input GET

```
<form action="App.php" method="GET">
     Username: <input type="text" name="username">
</form>
```

```
<?php
echo $_GET["username"];
echo "</br>";
$age = $_GET["age"];
echo $age;
?>
```

User Input POST

```
<form action="App.php" method="POST">
     Username: <input type="text" name="username">
</form>
```

```
<?php
echo $_POST["username"];
?>
```

Arrays

```
// $lucky_numbers = [];
// $lucky_numbers = array(4, 8, "fifteen", 16, 23, 42.0);
$lucky_numbers = [4, 8, "fifteen", 16, 23, 42.0];
//       indexes   0  1          2       3   4   5
```

```
$lucky_numbers[0] = 90;
echo $lucky_numbers[0]."<br>";
echo $lucky_numbers[1]."<br>";
echo count($lucky_numbers)."<br>";
```

2 Dimensional Arrays

```
$number_grid = [ [1, 2], [3, 4] ];
$number_grid[0][1] = 99;
```

```
echo $number_grid[0][0]."<br>";
echo $number_grid[0][1]."<br>";
```

Array Functions

```
$friends = [];
array_push($friends, "Oscar", "Angela");
array_push($friends, "Kevin");

// array_pop($friends);
echo "$friends[0], $friends[1], $friends[2] <br>";
sort($friends);
echo "$friends[0], $friends[1], $friends[2] <br>";
echo in_array("Oscar", $friends);
```

Associative Arrays

```
test_grades = {
    "Andy" => "B+",
    "Stanley" => "C",
    "Ryan" => "A",
    3 => 95.2
}
echo  test_grades["Andy"]."<br>";
echo  test_grades["Ryan"]."<br>";
echo  test_grades[3]."<br>";
```

Functions

```
function addNumbers($num1, $num2=99){
     return $num1 + $num2;
}

$sum = addNumbers(4, 3);
echo $sum;
```

If Statements

```
$isStudent = false;
$isSmart = false;

if($isStudent && $isSmart){
     echo "You are a student";
} elseif($isStudent && !$isSmart){
     echo "You are not a smart student";
} else {
     echo "You are not a student and not smart";
}
echo "<br>";

// >, <, >=, <=, !=, ==
if(1 > 3){
     echo "number comparison was true";
}
echo "<br>";

if("dog" == "cat"){
   echo "string comparison was true";
}
```

Switch Statements

```
$myGrade = "A";
switch($myGrade){
     case "A":
          echo "You Pass";
          break;
     case "F":
          echo "You fail";
          break;
     default:
          echo "Invalid grade";
}
```

While Loops

```
$index = 1;
while ($index <= 5){
     echo $index;
     $index += 1;
}

$index = 1;
```

```
do{
	echo $index;
	$index += 1;
}while ($index <= 5);
```

For Loops

```
for($i = 0; $i < 5; $i++){
	echo $i;
}

$luckyNums = [4, 8, 15, 16, 23, 42];
foreach($luckyNums as $luckyNum){
	echo $luckyNum."<br>";
}
```

Exception Catching

```
try{
	throw new Exception('Something bad happened');
} catch(Exception $e){
	echo $e->getMessage();
} finally{
	echo "<br> This code gets executed no matter what";
}
```

Classes and Objects

```
class Book{
	var $title;
	public $author;
	public static $staticAttribute = "My Static
Attribute";

	function readBook(){
		echo "Reading $this->title by $this->author";
	}
};
```

```
$book1 = new Book;
$book1->title = "Harry Potter";
$book1->author = "JK Rowling";

echo $book1->title."<br>";
echo Book::$staticAttribute."<br>";
$book1->readBook();
```

Constructors

```
class Book{
      var $title;
      public $author;

      function __construct($title, $author){
            $this->title = $title;
            $this->author = $author;
      }

      function readBook(){
            echo "Reading $this->title by $this->author";
      }
};


$book1 = new Book("Harry Potter", "JK Rowling");
// $book1->title = "Half-Blood Prince";

echo $book1->title."<br>";
```

Getters and Setters

```
class Book{
      private $title;
      public $author;

      function __construct($title, $author){
            $this->setTitle($title);
            $this->author = $author;
      }
```

```
	function getTitle(){
		return $this->title;
	}
	function setTitle($title){
		$this->title = $title;
	}

	function readBook(){
		echo "Reading $this->title by $this->author";
	}
};


$book1 = new Book("Harry Potter", "JK Rowling");
$book1->setTitle("Half-Blood Prince");

echo $book1->getTitle();
```

Inheritance

```
class Chef{

	public $name;
	public $age;

	function __construct($name, $age){
		$this->name = $name;
		$this->age = $age;
	}

	function makeChicken(){
		echo "The chef makes chicken";
	}
	function makeSalad(){
		echo "The chef makes salad";
	}
	function makeSpecialDish(){
		echo "The chef makes bbq ribs";
	}
};
class ItalianChef extends Chef{
     public $countryOfOrigin;

     function __construct($name, $age, $countryOfOrigin){
          $this->countryOfOrigin = $countryOfOrigin;
          parent::__construct($name, $age);
     }

     function makePasta(){
          echo "The chef makes pasta";
     }
     function makeSpecialDish(){
          echo "The chef makes chicken parm";
     }
};


$chef = new Chef("Gordon Ramsay", 50);
$chef->makeChicken();
echo "<br>";
$italianChef = new ItalianChef("Massimo Bottura", 55,
"Italy");
$italianChef->makeChicken();
echo "<br> $italianChef->countryOfOrigin";
```

Abstract Classes and Methods

```
abstract class Vehicle{
     public abstract function move();
     public function getDescription(){
          echo "Vehicles are used for transportation";
     }
}

class Bicycle extends Vehicle{
     public function move(){
          echo "The bicycle pedals forward";
     }
}

class Plane extends Vehicle{
	public function move(){
		echo "The plane flys through the sky";
	}
}


$plane = new Plane();
$plane->move();
echo "<br>";
$plane->getDescription();
```

Interface Inheritance

```
interface Animal{
	public function speak();
}

class Cat implements Animal{
	public function speak(){
		echo "Meow Meow <br>";
	}
}

class Dog implements Animal{
	public function speak(){
		echo "Woof Woof <br>";
	}
}


$animals = [
	new Dog(),
	new Cat()
];

foreach($animals as $animal){
	$animal->speak();
}
```