## Variable Naming Conventions

- Variable names in PHP start with a dollar sign `($)`
- The name must begin with a letter or underscore `( _ )`, followed by letters, numbers, or underscores.
- Variable names are case-sensitive, meaning `$myVariable` and `$myvariable` are treated as different variables.
- PHP does not require you to declare a variable before using it; it automatically creates the variable when you assign a value to it.

```
$name="John";
$age=30;
```

---

## Variable Types

PHP is a loosely typed language, which means you don't need to declare the data type of a variable explicitly. PHP determines the data type based on the value assigned to it. Common data types in PHP include

- **Strings:** Textual data enclosed in single **('')** or double **("")** quotes.
- **Integers:** Whole numbers.
- **Floats (or Doubles):** Numbers with decimal points.
- **Booleans:** True or false values.
- **Arrays:** Ordered collections of data.
- **Objects:** Instances of user-defined classes.
- **Null:** Represents a variable with no value.

```

$name="John Smith";

$age=30;

$price= 19.99;

$isStudent=true;

$fruits=array(`"apple"`,` "banana"`,` "cherry"`);


class Person {
    public $name;
    public $age;
}

$person1 = new Person();
$person1->name = "Alice";
$person1->age = 25;


$noValue = null;
```

---

## Find variable memory location

```php
class Test {
    public $name = "Example";
}

$obj = new Test();
echo spl_object_id($obj);  // Prints a unique identifier for the object
```

### Why Memory Addresses Are Not Directly Exposed

- **Security & Abstraction**: PHP is a high-level language and abstracts away these details to prevent potential vulnerabilities.
- **Garbage Collection**: PHP uses a reference-counting garbage collector, so the actual memory address is not consistent or accessible in user space.

---

### Variable Scope

PHP variables have different scopes, which determine where the variable can be accessed. The main variable scopes in PHP are

- **Local Scope:** Variables declared within a function are only accessible within that function.
- **Global Scope:** Variables declared outside of any function can be accessed anywhere in the script.
- **Superglobals:** Special global arrays like $_GET, $_POST, $_SESSION, etc., which are accessible from anywhere in the script.

**Local Scope:** Variables declared within a function have local scope, meaning they are only accessible within that function.

```php
function greet() {
    $message = "Hello, World!";
    echo $message;
}

greet();
```

**Global Scope:** Variables declared outside of any function have global scope and can be accessed from anywhere in your script.

```php
$name = "John";

function greet() {
    global $name;
    echo "Hello, $name!";
}

greet(); // Outputs: Hello, John!

// You can also access $name outside of the function.
echo "Outside function: $name"; // Outputs: Outside function: John
```

---

#php