

REG NUMBER: 21RP01204

ASSIGNMENT 2

* **Explain PHP programing**
* PHP(short for Hypertext PreProcessor) is the most widely used open source and general purpose server side scripting language used mainly in web development to create dynamic websites and applications.
* This server-side scripting language is commonly used in the back-end for tasks such as managing data on servers.
* PHP can be embedded in HTML code and was created to manage Lerdorf’s personal webpage at the time.
* In addition to having a large community and plenty of job opportunities, PHP is easy to learn and doesn’t require you to start from scratch. With so many available themes, plugins, and websites for WordPress, you can simply take some pre-existing code and edit it to create something new.
* PHP is also extremely quick to set up on your computer and won’t take days like some other programming languages.
* **Why do we need to use PHP programming**
* PHP is a great starting point for any new developer.
* PHP can actually do anything related to server-side scripting or more popularly known as the backend of a website.
* There are also many hash functions available in PHP to encrypt user’s data that makes PHP secure and reliable to be used as a server-side scripting language.
* PHP can run on all major operating systems like Windows, Linux, Unix, Mac OS X etc.
* PHP allows web developers to create dynamic content and interact with databases.
* PHP is known for its simplicity, speed, and flexibility — features that have made it a cornerstone in the web development world.
* PHP is freely available to download and use.
* **the latest PHP version we have today and list the updated features for the latest 3 release**

**PHP 8.2:** PHP 8.2, is the latest version of PHP and is a major update of the PHP language. It contains many new features, including read-only classes, null, false, and true as stand-alone types, deprecated dynamic properties, performance improvements, and more.

**updated features for the latest 3 release:**

* Scalar type declarations
* Return type declarations
* Null coalescing operator
* Spaceship operator
* Constant arrays using define()
* Unicode codepoint escape syntax
* Closure::call()
* Filtered unserialize()
* Group use declarations
* preg\_replace\_callback\_array()
* **Difference between new release vs stable release of a software product**
* Normal users choose the latest RELEASE. STABLE is for developers that improve the current product that will be the next RELEASE (most of this work is bug fixes and sometimes new features that won’t break the userland and 3rd-party applications).
* A stable release is a version that has been tested as thoroughly as possible and is as reliable as we can make it. It does not have all the new features of a beta release and it does not have the latest fixes for problems.
* The new release is a version that has been tested internally and is being tested by the wider community. It usually has fixes for bugs in the stable version, and has new features that are subject to change and need testing and may have their own bugs or limitations.
* **the main features of PHP programming**
* **Simple:** It is very simple and easy to use, compare to other scripting language it is very simple and easy, this is widely used all over the world.
* **Interpreted:** It is an interpreted language, i.e. there is no need for compilation
* **Open Source:** it means you no need to pay for use PHP, you can free download and use.
* **Case Sensitive:** PHP is case sensitive scripting language at the time of variable declaration. In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive. Platform Independent: PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.
* **Error Reporting:** PHP has some predefined error reporting constants to generate a warning or error notice.
* **Real-Time Access Monitoring:** PHP provides access logging by creating a summary of recent accesses for the user.
* **Availability of Information on memory and CPU utilization:** is available through PHP functions like memory get usage() and memory get peak usage(), which can assist programmers to optimize their code.
* **With a help of examples explain why PHP is case sensitive**

<?php

$num = 8;

$NUM = 6;

echo $num." + ".$NUM; // output : 8 + 6

?>

* **What and why do we use comments while writing php codes, with a help of examples explain**

**different types of php comments.**

* A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code. Comments can be used to: Let others understand your code.
* Comments are used in a programming language to document the program and remind programmers of what tricky things they just did with the code, or to warn later generations of programmers stuck with maintaining some spaghetti code.

**Comments can be used to:**

* Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code.
* Comments are employed to notify, warn, and remind others that didn't write the code [and your future self] of important things that the code is doing.

**The help of examples explains different types of PHP comments:**

* **single-line comments:** Single-line PHP comments are useful for short notes before a code block or for explaining a single line of code**.** type two forward slashes (//) followed by your comment text.

<!DOCTYPE html>

<html>

<body>

<?php

// HELLO WORLD

# This is also a single-line comment

?>

</body>

</html>

* **multiple-line comments:** in case you want to comment out a larger section of code or leave a more descriptive comment. Start your multiline comment by writing /\* and end it by writing \*/.

**<**!DOCTYPE html>

<html>

<body>

<?php

/\* i like being alone

i like to be happy on inside

hello me

\*/

?>

</body>

</html>

* **php output functions:**
* **ECHO() VS PRINT()**

**ECHO** has no return value

**ECHO** can take multiple parameters (although such usage is rare )

***ECHO*** is marginally faster than print..

<!DOCTYPE html>

<html>

<body>

<?php

echo "<h2>PHP is Fun!</h2>";

echo "Hello world!<br>";

echo "I'm about to learn PHP!<br>";

echo "This ", "string ", "was ", "made ", "with multiple parameters.";

?>

</body>

</html>

**PRINT** has a return value of 1 so it can be used in expressions.

**PRINT** can take one argument.

<!DOCTYPE html>

<html>

<body>

<?php

print "Hello world!";

?>

</body>

</html>

**PRINT() VS PRINTF()**

**print()**: The print() is used to create a PHP print statement to print given data to the browser. It accepts single data and prints it on the browser.

<?php

print "Apple";

// (or)

print("Apple");

?>

**printf(string\_format, values)** : This method is used to print the formatted output by using the values passed as the parameter of this function. So, this function accepts the output string format and the values to be added.

<?php

printf('We are expected to score above %d%% for distinction', 85);

// Output: We are expected to score above 85%

// for distinction

?>

**Printf() vs print\_r()**

* **printf(string\_format, values)** : This method is used to print the formatted output by using the values passed as the parameter of this function. So, this function accepts the output string format and the values to be added.

<?php

printf('We are expected to score above %d%% for distinction', 85);

// Output: We are expected to score above 85%

// for distinction

?>

* **print\_r():** This function is used to print the compound data like PHP array or objects.

<?php

$my\_var = array(“a”=>”10”,”b”=>”78”);

Print\_r($my\_var);

//Output: Array ( [a] => 10 [b]=> 78)

?>

**Print\_r vs var\_dump()**

* **print\_r():** This function is used to print the compound data like PHP array or objects.

<?php

$my\_var = array(“a”=>”10”,”b”=>”78”);

Print\_r($my\_var);

//Output: Array ( [a] => 10 [b]=> 78)

?>

* **var\_dump() –** var\_dump() also prints array data in structured manner. It gives additional data, like, the data type, the length, values and more.

<?php

var\_dump(false); // prints bool(false)

print\_r(false); // returns empty string

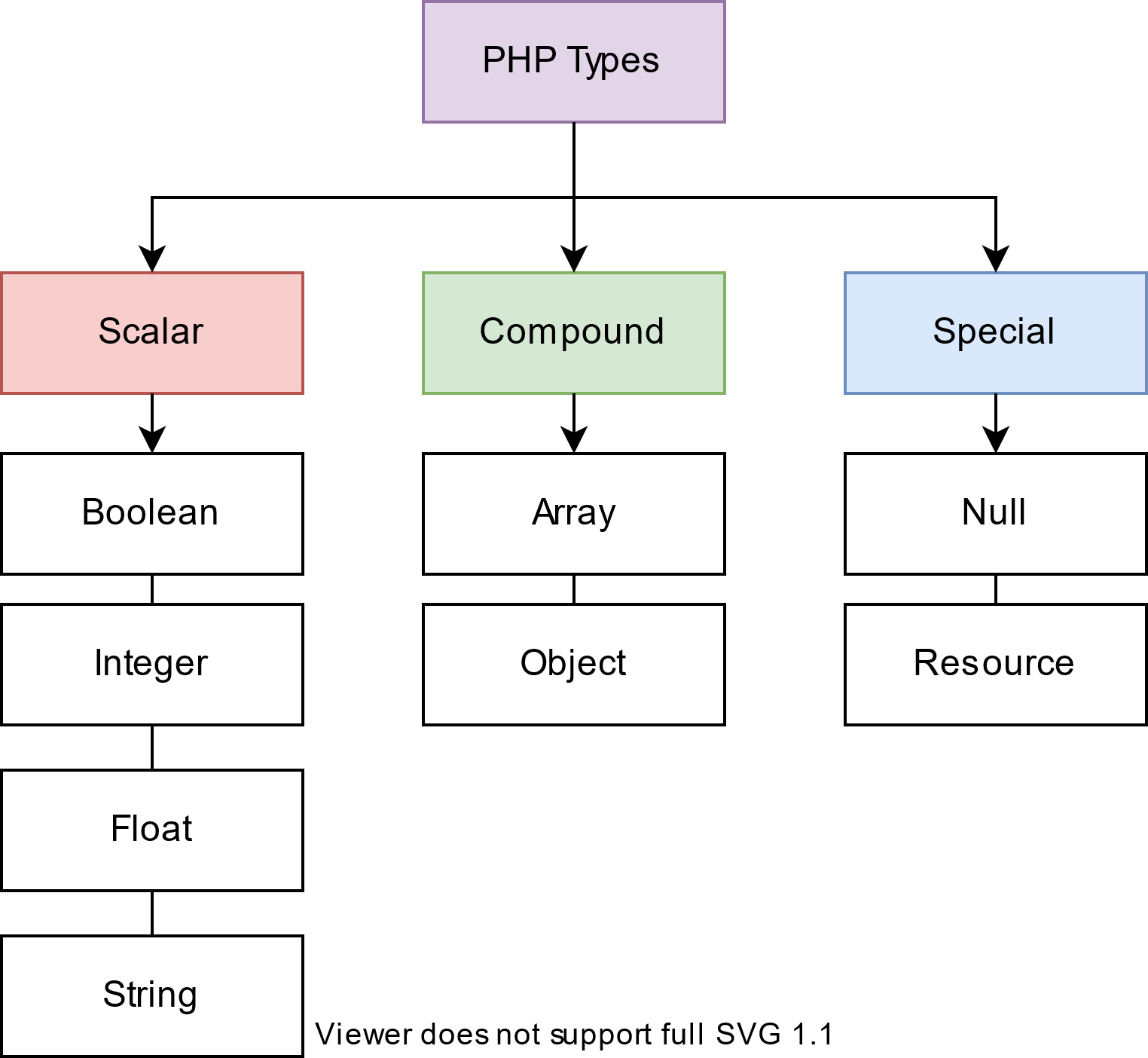
?>

* **List and describe different data-type we have in php by categorizing them in scalar, compound**

**and special datatypes.**

A **type** specifies the amount of memory that allocates to a value associated with it. A type also determines the operations that you can perform on it.

PHP has ten primitive types including four scalar types, four compound types, and two special types:



* **Scalar types**

A variable is a scalar when it holds a single value of the type integer, float, string, or Boolean.

* **Integer**

Integers are whole numbers defined in the set {…-3,-2-,-1,0,1,2,3…}. The size of the integer depends on the platform where PHP runs.

The constant PHP\_INT\_SIZE specifies the size of the integer on a specific platform. PHP uses the int keyword to denote the integer type.

Example:

<?php

$count = 0;

$max = 1000;

$page\_size = 10;

* **Float**

**Floats** are floating-point numbers, which are also known as floats, doubles, or real numbers. A float (floating point number) is a number with a decimal point or a number in exponential form.  floats have limited precision.

Example:

<?php

$price = 10.25;

$tax = 0.08;

* **Boolean**

**Boolean** represents a truth value that can be either true or false. PHP uses the bool keyword to represent the Boolean type.

The bool type has two value**s true** and **false**. Since keywords are case-insensitive, you can use true, True, TRUE, false, False, and False to indicate Boolean values.

Example:

<?php

$is\_admin = true;

$is\_user\_logged\_in = false;

* **String**

<?php

$str = 'PHP scalar type';

$message = "PHP data types";

* **Compound types**

**Compound data** includes values that contain more than one value. PHP has two compound types including array and object.

* **Array**

An array is an ordered map that associates keys with values.

For example:

<?php

$carts = [ 'laptop', 'mouse', 'keyboard' ];

**The $carts** array contains three string values. It maps the index 0, 1, and 2 to the values 'laptop', 'mouse', and 'keyboard'. The $carts are called an **indexed array** because it uses numeric indexes as keys.

**To access a value in an array, you use the square brackets:**

<?php

echo $carts[0]; // 'laptop'

echo $carts[1]; // 'mouse'

echo $carts[2]; // 'keyboard'

* **Object**

An object is an instance of a class. It’s a central concept in object-oriented programming.

An object has properties. For example, a person object may have the first name, last name, and age properties.

An object also has behaviors, which are known as methods. For example, a person object can have a method called getFullName() that returns the full name.

* **Special types**

PHP has two special types**: null and resource.**

* **Null:** The null type has one value called null that represents a variable with no value.
* **Resource:** The resource type holds a reference to an external resource, e.g. a filehandle or a database connection.
* **php variable, list the variable naming rules you have to obey while defining a variable in PHP.**

A PHP variable is a memory zone identified by a name and a descriptive name that holds information in php codes for later use. A variable can store numeric values, characters, character strings, or memory addresses so that they can be used in any part of the program.

Rules to follow while naming php variable

* Variable names cannot contain spaces.
* Variable names cannot start with a number.
* Variable names cannot be a reserved word in PHP.
* Variable names must start with a letter or an underscore (\_).
* Variable names can only contain letters, numbers, and underscores.
* Variable names are case-sensitive, meaning that $myVar and $myvar are considered to be different variables.
* **List and explain at least 10 super global variables**
* $\_SERVER: An associative array that contains information about the server and the current script, such as the server name, the script name, and the client's IP address.
* $\_FILES: An associative array that contains information about uploaded files.
* $\_ENV: An associative array that contains information about the environment variables.
* $\_COOKIE: An associative array that contains information about cookies
* $\_SESSION: An associative array that contains session variables.
* $GLOBALS: An associative array that contains all global variables.
* $argv, $argc: store the command-line arguments passed to the script. $argc contains the number of arguments, and $argv contains an array of the arguments.
* $\_GET: An associative array that contains data passed in the URL via the GET method.
* $\_POST: An associative array that contains data passed in the HTTP body via the POST method.
* $\_REQUEST: An associative array that contains data passed via either the GET, POST, or COOKIE methods

<https://www.techtarget.com/searchsoftwarequality/definition/release>

<https://www.sitesbay.com/php/php-features-of-php>

<https://phppot.com/php/php-print-statements/>

<https://www.phptutorial.net/php-tutorial/php-data-types/>

<https://www.netsolutions.com/insights/what-is-php/#:~:text=PHP%20allows%20web%20developers%20to,in%20the%20web%20development%20world>.

<https://www.geeksforgeeks.org/what-is-php-and-why-we-use-it/>

<https://www.php.net/manual/en/migration70.new-features.php>