NAME: UMUHOZA Bayingana Denyse

Class: IT B

RegNo: 21RP03648

Q1. What is PHP

The name PHP stands for Hypertext Preprocessor and denotes a server-side scripting language, which means that the applications written on it run on web servers and do not depend on the web browser. However, over the years, its area of use has shifted, and nowadays the PHP coding language is ranked among the best and most popular programming tools for web development due to its many virtues which will be the focus of this article. It is considered a very effective technology that offers a convenient development process with many additional tools to aid it. In fact, according to the Popularity of Programming Language Index (PYPL), PHP is the fifth most popular coding language in the world.

Q2. reasons for using PHP:

1. many available specialists;
2. a large base of reference and educational materials;
3. better loading speed of websites;
4. more options for database connectivity;
5. a large collection of open-source addons;
6. inexpensive website hosting;
7. great synergy with HTML;
8. excellent flexibility and combinability;
9. various benefits provided by cloud solutions.
10. Now, let’s review the listed benefits of PHP more closely in order to discover how they can be applied in practice to improve the quality, functionality, and profitability of your web app.

Q3.the latest php version we have today is **8.2.0** and the updated features for the latest 3  
are **8.0.26, 8.1.13**

Q4. **A new release: is** a change or set of changes that is created to be delivered to the production environment. **while**

**A stable release** is a version of a software package that has been tested and verified. It is the latest of a program that is considered safe for public use.

Q5. **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.

**Flexibility:** PHP is known for its flexibility and embedded nature as it can ba well integrated with HTML ,XML ,Javascript and many more.

**Interpreted:** It is an interpreted language, i.e. there is no need for compilation.

**Faster and efficient performance:** users generally prefer fast loading website.

**Open Source:** Open source means you no need to pay for use php, you can free download and use.

**Platform Independent:** PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.

**Maintenance**: when dealing with big project, maintenance of code is also an important aspect of the web development process.

**Case-sensitive:** PHP is a partially case-sensitive language. Although functions names are not case-sensitive, other things in PHP are case-sensitiveQ6. Case Sensitive

Q6. PHP classes are a mix between variables and functions, so they are partially case-sensitive. Such as: the variables $num and $NUM can have different values.

But when you declare two functions with the same name, PHP give or produce a fatal error: cannot redeclare the function.

Q7. 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. Examples

1. **This is a single-line comment**

<!DOCTYPE html>

<html>

<body>

<?php

// This is a single-line comment

# This is also a single-line comment

?>

</body>

</html>

2. <! DOCTYPE html>

<html>

<body>

2. **multiple-lines comment**

<? php

/\*

This is a multiple-lines comment block

that spans over multiple

lines

\*/

?>

</body>

</html>

Q8. A) **Echo:** In PHP, Echo acts as a statement that is used to show the output. It does not return any value and has the ability to pass multiple strings split by comma (,) in echo. We can use the Echo statement with and without parentheses, and it is faster in nature.

* **Print: In** PHP, the Print statement is also used to show the output. We can use it as an alternative to Echo. However, it is slower than Echo and returns an integer value 1. Also, in the Print statement we cannot pass multiple arguments.

|  |  |  |
| --- | --- | --- |
| **1.** | In Echo, we can pass multiple arguments separated by commas. | In Print, we cannot pass multiple arguments. |
| **2.** | In Echo, we can exhibit the outputs of one or more strings separated by commas. | Through the Print statement, we can only show the strings. |
| **3** | Echo can be used with or without parentheses | Print can also be used with or without parentheses. |
| **4.** | It never returns any value. | It always returns the integer value that is 1. |
| **5.** | This statement is fast as compared to the print statement. | It is slow as compared to the echo statement. |

b) print vs printf

printf() outputs a formatted string whereas print() outputs one or more strings.

example: <?php

print "Hello, world!";

?>

output: Hello, world!

example: <?php

$number = 8;

$str = "Solar System";

printf("There are %u planets in the %s.", $number, $str);

?>

output: There are 8 planets in the Solar System.

C) printf() vs print\_r()

The printf( ) function builds a formatted string by inserting values into a template. The print\_r( ) function is useful for debugging—it prints the contents of arrays, objects, and other things, in a more-or-less human-readable form.

Example 1: <?php

$number = 8;

$str = "Solar System";

printf("There are %u planets in the %s.", $number, $str);

?>

output: There are 8 planets in the Solar System.

Example 2:

class P {

var $name = 'nat';

// ...

}

$p = new P;

print\_r($p);

output

Object

(

[name] => nat

)

d) print\_r() vs var\_dump()

The var\_dump() function displays structured information about variables/expressions including its type and value. Whereas The print\_r() displays information about a variable in a way that's readable by humans.

$arr = array ('xyz', false, true, 99, array('50'));

output for print\_r($arr)

Array

(

[0] => xyz

[1] =>

[2] => 1

[3] => 99

[4] => Array

(

[0] => 50

)

)

output for var\_dump($arr)

array(5) {

[0]=>

string(3) "xyz"

[1]=>

bool(false)

[2]=>

bool(true)

[3]=>

int(100)

[4]=>

array(1) {

[0]=>

string(2) "50"

}

}

Q9. **Data Types:** define the type of data a variable can store.

* **Integer:** Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point
* **String:** Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration
* **Double**: Can hold numbers containing fractional or decimal parts including positive and negative numbers or a number in exponential form
* **Array:** Array is a compound data type that can store multiple values of the same data type. Below is an example of an array of integers
* **Boolean:** Boolean data types are used in conditional testing. Hold only two values, either TRUE (1) or FALSE(0). Successful events will return true and unsuccessful events return false

Q10**. A variable:** Is A Memory Zone to store data. A variable can have a short name (like x and y) or a more descriptive name (age, car name, total volume)

**Rules for PHP variables:**

* A variable start with the $ sign, followed by the name of the variable
* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and …)
* Variable names are case-sensitive ($age and $AGE are two different variables)

Q11.

All of the super global variables act as associative arrays that use a string value as a key to access values. The following is a list of super global variables in PHP:

* **＄GLOBALS** is the super global variable that stores all user-defined global variables. The global variable names act as keys to their values.
* **＄\_SERVER** contains data about headers, scripts, and paths. The keys to the values in this array are predefined.
* **＄\_REQUEST** stores data input in the form of HTTP POST, GET and Cookies. The keys to this array are defined in the HTTP requests.
* **＄\_POST** stores data input in the form of POST requests. The keys to this array are defined in the HTTP POST request.
* **＄\_GET** has data input in the form of GET requests. The keys to this array are defined in the HTTP GET request.
* **＄\_FILES** is a two-dimensional associative array that contains a list of files that were uploaded to the script using the POST method. The keys to this array are the names of the fields uploading the files and the data being accessed
* **＄\_COOKIES** keeps data input via HTTP Cookies. The keys to this array are defined when the cookies are set.
* **＄\_SESSION** holds session variables. Session variables can be accessed on multiple pages. This array’s keys are defined by the users when they define session variables.
* **＄\_ENV** contains information about the environment that PHP is running in. The keys to the values in this array are predefined.

# References

<https://www.jobsity.com/blog/8-reasons-why-php-is-still-so-important-for-web-development>

<https://www.php.net/releases/index.php>