**NAME: KWIZERISEZERANO**

**REGNO: 21RP01262**

**Q1.PHP** is an acronym for "PHP: Hypertext Preprocessor", PHP is **a widely-used, open source ,** **interpreted, and object-oriented** , **a general-purpose scripting language** that you can use to make lots of projects, including Graphical User Interfaces (GUIs), web development and PHP scripts are executed on the server  generates the dynamic page. PHP is free to download and use.

**Q2.we need to use PHP programming because of the following reasons:**

* .PHP is amazing and popular language
* It is powerful enough to develop web applications.
* It is also easy enough to be a beginner's first server side language!
* It is used to generate dynamic page content
* It is used create,open,read,write,tdelete and close file on the server
* It is used collect form data
* It is used send and receive cookies
* It is used add,delete,modify data in your database
* It is used to control user access
* It is used to encrypt data PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
* PHP is compatible with almost all servers used today (Apache, IIS, etc.)
* PHP supports a wide range of databases
* PHP is free. Download it from the official PHP resource: [www.php.net](http://www.php.net/)
* PHP is easy to learn and runs efficiently on the server sid

**Q3.the latest PHP version we have to day is 8.2**

**The list of updated features for latest 3 release:**

**Php 8.2** is the latest PHP version which brings readonly classes, DNF types, null false and true types, sensitive parameter redaction support, a new random extension and new features along with with deprecations.

**PHP 8.1**, released in 2021, brings major new features such as [Enums](https://php.watch/versions/8.1/enums), [Fibers](https://php.watch/versions/8.1/fibers), [never return type](https://php.watch/versions/8.1/never-return-type), [Intersection Types](https://php.watch/versions/8.1/intersection-types), [readonly properties](https://php.watch/versions/8.1/readonly), and more, while ironing out some of its undesired legacy features by deprecating them.

**PHP 8.0**, on the 25th year of PHP history, brings several important features such as [Union Types](https://php.watch/versions/8.0/union-types), [JIT](https://php.watch/versions/8.0/JIT), [Constructor Property Promotion](https://php.watch/versions/8.0/constructor-property-promotion), [Match Syntax](https://php.watch/versions/8.0/match-expression), [Named Parameters](https://php.watch/versions/8.0/named-parameters), and several more performance, syntax, and quality-of-life improvements

Q4**. A new release of software**  is the distribution of the final version or the newest version of a [software](https://www.techtarget.com/searchapparchitecture/definition/software) application. A software release may be public or private.

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.

**Q5. PHP features are:**

**Performance:**

PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

**Open Source:**

PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

**Familiarity with syntax:**

PHP has easily understandable syntax. Programmers are comfortable coding with it.

**Embedded:**

PHP code can be easily embedded within HTML tags and script.

**Platform Independent:**

PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

**Database Support:**

PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

**Error Reporting -**

PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E\_ERROR, E\_WARNING, E\_STRICT, E\_PARSE.

**Loosely Typed Language:**

PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

**Web servers Support:**

PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

**Security:**

PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

**Control:**

Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

**A Helpful PHP Community:**

It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits.

**Q6.** PHP is a unique programming language in terms of case sensitivity. **In PHP, variables and constants are case sensitive, because the lowercase variables or constants are totally different from their uppercase. example here $num is to totally different from $NUM and each variable has its own value.**

//you can create two variables like this:

$num=99;

$NUM=20;

echo $num; //99

echo $NUM; //20

**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.

**We use Comments while we writing PHP codes to:**

* Let others understand your code
* 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.

**Types of PHP comments:**

**Single-line comment**: this is one of type of PHP comment that is used to comment only single or one line of code. And it is done by writing // or # followed by comment.

Example:

<!DOCTYPE html>  
<html>  
<body>  
  
<?PHP  
// This is a single-line comment  
  
# This is also a single-line comment  
?>  
  
</body>  
</html>

**Multiple-lines comment:** this is one of type of PHP comment that is used to comment multiple lines of codes. And it is done by writing /\* comment \*/

**Example:**

**<!DOCTYPE html>  
<html>  
<body>  
  
<?PHP  
/\*  
This is a multiple-lines comment block  
that spans over multiple  
lines  
\*/  
?>  
  
</body>  
</html>**

Q8.**a) With PHP, there are two basic ways to get output: echo and print:**

**echo** and **print** are more or less the same. They are both used to output data to the screen. The differences are small:

* **echo** has no return value while **print** has a return value of 1 so it can be used in expressions.
* **echo** can take multiple parameters (although such usage is rare) while **print**can take one argument.
* **echo** is marginally faster than **print**.

**Example:**

**<?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.";  
?>**

**<?php  
print "<h2>PHP is Fun!</h2>";  
print "Hello world!<br>";  
print "I'm about to learn PHP!";  
?>**

b) They are all used to print text on the screen but printf function is way more complicated.

* I believe **print is simply for text, printf for Formatting text with some HTML markup (F = Formatting)**.
* **printf()** outputs a formatted string whereas **print()** outputs one or more strings.

**The print ()** function outputs one or more strings.

**Note:**The **print ()** function is not actually a function, so you are not required to use parentheses with it.

**Tip:** The **print ()** function is slightly slower than [echo()](https://www.w3schools.com/php/func_string_echo.asp)

**Example:**

**Print()**

**<?php  
$str = "Hello world!";  
print $str;  
?>**

**Printf()**

**<?php  
$number = 123;  
printf("With 2 decimals: %1\$.2f  
<br>With no decimals: %1\$u",$number);  
?>**

**c)**The **print\_r** is used to display human-readable information about a variable. 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.

**PHP string printf()** function predefined functions. It is used **to output a formatted string**. We can pass the arg1, arg2, arg++ parameters at percent (%) signs in the main string.

**The printf function** prints a string on the screen using a “format string” that includes the instructions to mix several strings and produce the final string to be printed on the screen.

**Example:**

**<?php  
$number = 9;  
$str = "Beijing";  
printf("There are %u million bicycles in %s.",$number,$str);  
?>**

**<?**

**$a = array("red", "green", "blue");**

**print\_r($a);**

**echo "<br>";**

**?>**

d) The **var\_dump()** function displays structured information about variables/expressions including its type and value. The **var\_dump**() function dumps information about one or more variables. The information holds type and value of the variable(s).

Whereas **The print\_r() displays** information about a variable in a way that's readable by humans.

**Example**

**<?**

**$a = array("red", "green", "blue");**

**print\_r($a);**

**echo "<br>";**

**?>**

**Q9.**

The data type specifies the amount of memory that allocates to the variable associated with it. In addition, the data type determines the operations that you can perform on it.**PHP** data types are used to hold different types of data or values. **PHP** supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types
2. Compound Types
3. Special Types

**PHP** allows eight different types of data types. All of them are discussed below. The first five are called simple data types and the last three are compound data types.

**PHP** Data Types: Scalar Types

In simple words, a variable is called scalar type if it holds singular value only. There are 4 scalar data types in **PHP**.

1. boolean
2. integer
3. float
4. string

**PHP** Data Types: Compound Types

In contrast to Scalar data types, a variable is called compound if it holds multiples values within. There are 2 compound data types in **PHP**.

1. array
2. object

**PHP** Data Types: Special Types

There are 2 special data types in **PHP**.

1. resource
2. NULL

**PHP** Scalar Data Types

* Boolean

Booleans are like a switch which has only two possible values either 1 (true) or 0 (false). Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean.

**<!DOCTYPE html>**

**<head>**

**<title>try to program-Boolean</title>**

**</head>**

**<body>**

**<?php**

**// Assign the value TRUE to a variable**

**$show\_error = True;**

**var\_dump($show\_error);**

**?>**

**</body>**

**</html>**

**Ouput:**

**bool(true)**

* Integer

 Integers hold only whole numbers including positive and negative numbers, i.e, numbers without fractional part or decimal point. They can be decimal (base 10), octal (base 8) or hexadecimal (base 16). The default base is decimal (base 10).

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>PHP Integers</title>**

**</head>**

**<body>**

**<?php**

**$a = 123; // decimal number**

**var\_dump($a);**

**echo "<br>";**

**$b = -123; // a negative number**

**var\_dump($b);**

**echo "<br>";**

**$c = 0x1A; // hexadecimal number**

**var\_dump($c);**

**echo "<br>";**

**$d = 0123; // octal number**

**var\_dump($d);**

**?>**

**</body>**

**</html>**

**Output:**

**int(123)**

**int(-123)**

**int(26)**

**int(83)**

* **PHP** Floating Point Numbers or Doubles

Floating point numbers (also known as “floats”, “doubles”, or “real numbers”) are decimal or fractional numbers like demonstrated in the example below.

**Example:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>PHP Floats</title>**

**</head>**

**<body>**

<?php

$a = 1.234;

var\_dump($a);

echo "<br>";

 $b = 10.2e3;

var\_dump($b);

echo "<br>";

$c = 4E-10;

var\_dump($c);

?>

</body>

</html>

Output:

float(1.234)

float(10200)

float(4.0E-10)

* String

In a string, letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes but it will be treated differently while printing variables. To clarify this look at the example below.

<!DOCTYPE html>

<html lang="en">

<head>

    <title>PHP Strings</title>

</head>

<body>

<?php

$a = 'Hello world!';

echo $a;

echo "<br>";

$b = "Hello world!";

echo $b;

echo "<br>";

$c = 'Stay here, I\'ll be back.';

echo $c;

?>

</body>

</html>

Output:

Hello world!

Hello world!

Stay here, I'll be back.

**PHP** Compound DataTypes

* Array

An array is a variable that can hold more than one value at a time. It is useful to aggregate a series of related items together, for example, a set of country or city names.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <title>PHP Arrays</title>

</head>

<body>

<?php

$colors = array("Red", "Green", "Blue");

var\_dump($colors);

echo "<br>";

$color\_codes = array(

    "Red" => "#ff0000",

    "Green" => "#00ff00",

    "Blue" => "#0000ff"

);

var\_dump($color\_codes);

?>

</body>

</html>

Output:

array(3) { [0]=> string(3) "Red" [1]=> string(5) "Green" [2]=> string(4) "Blue" }

array(3) { ["Red"]=> string(7) "#ff0000" ["Green"]=> string(7) "#00ff00" ["Blue"]=> string(7) "#0000ff" }

* Object

An object is a data type which stores not only data but also information on how to process that data. Unlike the other data types in PHP, an object must be explicitly declared.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <title>PHP Objects</title>

</head>

<body>

<?php

// Class definition

class greeting{

    // properties

    public $str = "Hello World!";

    // methods

    function show\_greeting(){

        return $this->str;

    }

}

// Create object from class

$message = new greeting;

var\_dump($message);

?>

</body>

</html>

Output:

object(greeting)#1 (1) { ["str"]=> string(12) "Hello World!" }

**PHP** Special Data Types

* Null

The special NULL value is used to represent empty variables in PHP. A variable of type NULL is a variable without any data. NULL is the only possible value of type null.   Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <title>PHP NULL Value</title>

</head>

<body>

<?php

$a = NULL;

var\_dump($a);

echo "<br>";

$b = "Hello World!";

$b = NULL;

var\_dump($b);

?>

</body>

</html>

Output:

NULL

NULL

* Resources

A resource is a special variable, holding a reference to an external resource.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <title>PHP Resources</title>

</head>

<body>

<?php

// Open a file for reading

$handle = fopen("note.txt", "r");

var\_dump($handle);

?>

</body>

</html>

Output:

resource(2) of type (stream)

**A string** is a sequence of characters, like "Hello world!".

**A string** can be any text inside quotes. You can use single or double quotes:

**An integer data type** is a non-decimal number between -2,147,483,648 and 2,147,483,647.

**A float (floating point number)** is a number with a decimal point or a number in exponential form.

**A Boolean** represents two possible states: TRUE or FALSE

**Booleans** are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial

**An array** stores multiple values in one single variable.

**Classes and objects** are the two main aspects of object-oriented programming. When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

**Null** is a special data type which can have only one value: NULL.

The **special resource type** is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

Q10. **Variables** are "containers" for storing information

**A variable** can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

**Rules for PHP variables:**

* A variable starts 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). A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

Rules for PHP variables:

* A variable starts 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)

Q10. Superglobals are built-in variables that are always available in all scopes.

1. **$GLOBALS**
2. **$\_SERVER**
3. **$\_REQUEST**
4. **$\_POST**
5. **$\_GET**
6. **$\_FILES**
7. **$\_ENV**
8. **$\_COOKIE**
9. **$\_SESSION**
10. **$Http\_Response\_Header**

**$GLOBALS** is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS

**<?php  
$x = 75;  
$y = 25;  
   
function addition() {  
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];  
}  
   
addition();  
echo $z;  
?>**

**$\_SERVER** is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $\_SERVER:

**<?php  
echo $\_SERVER['PHP\_SELF'];  
echo "<br>";  
echo $\_SERVER['SERVER\_NAME'];  
echo "<br>";  
echo $\_SERVER['HTTP\_HOST'];  
echo "<br>";  
echo $\_SERVER['HTTP\_REFERER'];  
echo "<br>";  
echo $\_SERVER['HTTP\_USER\_AGENT'];  
echo "<br>";  
echo $\_SERVER['SCRIPT\_NAME'];  
?>**

**PHP $\_REQUEST** is a PHP super global variable which is used to collect data after submitting an HTML form.

**<html>  
<body>  
  
<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">  
  Name: <input type="text" name="fname">  
  <input type="submit">  
</form>  
  
<?php  
if ($\_SERVER["REQUEST\_METHOD"] == "POST") {  
  // collect value of input field  
  $name = $\_REQUEST['fname'];  
  if (empty($name)) {  
    echo "Name is empty";  
  } else {  
    echo $name;  
  }  
}  
?>  
  
</body>  
</html>**

## PHP $\_POST

**PHP $\_POST** is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $\_POST is also widely used to pass variables.

**<html>  
<body>  
  
<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">  
  Name: <input type="text" name="fname">  
  <input type="submit">  
</form>  
  
<?php  
if ($\_SERVER["REQUEST\_METHOD"] == "POST") {  
  // collect value of input field  
  $name = $\_POST['fname'];  
  if (empty($name)) {  
    echo "Name is empty";  
  } else {  
    echo $name;  
  }  
}  
?>  
  
</body>  
</html>**

## PHP $\_GET

**PHP $\_GET** is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$\_GET can also collect data sent in the URL.

**<html>  
<body>  
  
<a href="test\_get.php?subject=PHP&web=W3schools.com">Test $GET</a>  
  
</body>  
</html>**

The **$http\_response\_header** array is similar to the [get\_headers()](https://www.php.net/manual/en/function.get-headers.php) function. When using the [HTTP wrapper](https://www.php.net/manual/en/wrappers.http.php), $http\_response\_header will be populated with the HTTP response headers. $http\_response\_header will be created in the [local scope](https://www.php.net/manual/en/language.variables.scope.php).

**PHP $\_SESSION** is **an associative array that contains all session variables**. It is used to set and get session variable values. Example: Store information.

**$\_ENV** is **another superglobal associative array in PHP**. It stores environment variables available to current script. $HTTP\_ENV\_VARS also contains the same information, but is not a superglobal, and now been deprecated. Environment variables are imported into global namespace.

What does $\_ files do in PHP?

**$\_FILES** is a super global variable which can be used to **upload files**. Here we will see an example in which our php script checks if the form to upload the file is being submitted and generates a message if true.

**$\_cookie** **another superglobal associative array in PHP that create**  **a small file that the server embeds on the user's computer**. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

# References

<https://interviewbazaar.net/interview-question/what-is-the-difference-between-printf-and-print-in-php/>

<https://www.w3schools.com/php/php_regex.asp>

<https://www.php.net/manual/en/language.variables.superglobals.php>

<https://www.w3schools.com/php/php_intro.asp>

<https://www.w3schools.com/php/php_comments.asp>

<https://www.w3schools.com/php/php_variables.asp>

<https://www.w3schools.com/php/php_echo_print.asp>

<https://www.w3schools.com/php/php_datatypes.asp>

<https://www.w3schools.com/php/php_superglobals.asp>

<https://php.watch/versions>