# PHP – Data Types

The term "data types" refers to the classification of data in distinct categories. PHP has a total of eight data types that we use to construct our variables –

- **Integers** – Whole numbers, without a decimal point, like 4195.
- **Doubles** – Floating-point numbers like 3.14159 or 49.1.
- **Booleans** – Have only two possible values, either true or false.
- **NULL** – Special type that only has one value: NULL.
- **Strings** – Sequences of characters, like 'PHP supports string operations.'
- **Arrays** – Named and indexed collections of other values.
- **Objects** – Instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** – Special variables that hold references to resources external to PHP (such as database connections).

The first five are simple types, and the next two (arrays and objects) are compound types. The compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

In this chapter, let's discuss in detail about these built-in data types of PHP.

## Integer Data Type in PHP

A whole number without a decimal point (like 4195) is of **int** type in PHP. Integer data types are the simplest type. They correspond to simple whole numbers, both positive and negative.

- An **int** is a number of the set Z = {..., -2, -1, 0, 1, 2, ...}.
- An **int** can be represented in a decimal (base 10), hexadecimal (base 16), octal (base 8) or binary (base 2) notation.

To use octal notation, a number is preceded with "0o" or "0O". To use hexadecimal notation, precede the number with "0x". To use binary notation, precede the number with "0b".

Given below are some **examples** –

- **Decimal Integer** – 201, 4195, -15
- **Octal Integer** – 0010, 0O12, -0O21
- **Hexadecimal Integer** – 0x10, -0x100
- **Binary Integer** – 0b10101, -0b100

Integers can be assigned to variables, or they can be used in expressions, like so –

```
$int_var = 12345;
$another_int = -12345 + 12345;
```

## Double Data Type in PHP

Double variables represent floating point numbers (also known as "floats", "doubles", or "real numbers") that are the numbers with a fractional component. The fractional component follows after the integer component separated by the decimal symbol (.)

**Note** – A double variable can be positive, negative, or zero.

```
$var1 = 1.55
$var2 =-123.0
```

Explore our latest online courses and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

## Scientific Float Notation

PHP also allows the use of scientific notation to represent a floating point number with more digits after the decimal point. The symbol "E" or "e" is used to separate the integer and fractional part.

```
- 1.2e3, 2.33e-4, 7E-10, 1.0E5
```

By default, doubles print with the minimum number of decimal places needed. Take a look at the following **example** –

Open Compiler

```
<?php
   $many = 2.2888800;
   $many_2 = 2.2111200;
   $few = $many + $many_2;

   print("$many + $many_2 = $few");
?>
```

It produces the following **output** –

```
2.28888 + 2.21112 = 4.5
```

## Boolean Data Type in PHP

The **bool** type only has only two values; it can either be True or False. The **bool** type is used to express a truth value.

```
$bool1 = true;
$bool2 = false;
```

You can also use the integer values "1" and "0" to represent True and False Boolean values –

```
$bool3 = 1;
$bool4 = 0;
```

Typically, the result of an operator which returns a bool value is passed on to a control structure such as **if, while** or **do-while**. For example,

```
if (TRUE)
   print("This will always print.");

else
   print("This will never print.");
```

### Interpreting Other Data Types as Booleans

Here is a set of rules that you can use to interpret other data types as Booleans –

- If the value is a number, then it is False only if the value is equal to zero, otherwise the value is True.
- If the value is a string, it is False if the string is empty (has zero characters) or is the string "0", and is True otherwise.

- The values of type NULL are always False.
- If the value is an array, it is False if it contains no other values; True otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).

**Note** – Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;
$true_str = "Tried and true"
$true_array[49] = "An array element";
$false_array = array();
$false_null = NULL;
$false_num = 999 - 999;
$false_str = "";
```

## String Data Type in PHP

A string is a sequence of characters, for example, 'PHP supports string operations.'

In PHP, a character is the same as a byte. It means PHP only supports a 256 character set, and hence does not offer native Unicode support.

PHP supports single-quoted as well as double-quoted string formation. Both the following representations are valid in PHP –

```
$string_1 = "This is a string in double quotes";
$string_2 = 'This is a somewhat longer, singly quoted string';
```

Here are some more examples of string type –

```
$string_39 = "This string has thirty-nine characters";
$string_0 = "";      // a string with zero characters
```

Single-quoted strings are treated almost literally, whereas double-quoted strings replace variables with their values as well as specially interpreting certain character sequences.

</> Open Compiler

```
<?php
   $variable = "name";
   $literally = 'My $variable will not print!';

   print($literally);
   print "\n";

   $literally = "My $variable will print!";
   print($literally);
?>
```

When you run this code, it will produce the following **output** –

```
My $variable will not print!
My name will print
```

There are no artificial limits on string length. Within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP –

- Certain character sequences beginning with backslash (\) are replaced with special characters.
- Variable names (starting with **$**) are replaced with string representations of their values.

The escape-sequence replacements are –

- **\n** is replaced by the newline character
- **\r** is replaced by the carriage-return character
- **\t** is replaced by the tab character
- **\$** is replaced by the dollar sign itself ($)
- **\"** is replaced by a single double-quote (")
- **\\** is replaced by a single backslash (\)

PHP also has **Heredoc** and **Nowdoc** representations of string data type.

## Heredoc Representation of String Data Type

You can assign multiple lines to a single string variable using heredoc −

```
<?php
   $channel =<<<_XML_

   <channel>
      <title>What's For Dinner</title>
      <link>http://menu.example.com/ </link>
      <description>Choose what to eat tonight.</description>
   </channel>
   _XML_;

   echo <<< END
      This uses the "here document" syntax to output multiple lines with
      variable interpolation. Note that the here document terminator must
      appear on a line with just a semicolon. no extra whitespace!
   END;

   print $channel;
?>
```

When you run this code, it will produce the following output −

```
This uses the "here document" syntax to output
multiple lines with variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon. no extra whitespace!

<channel>
  <title>What's For Dinner</title>
  <link>http://menu.example.com/ </link>
  <description>Choose what to eat tonight.</description>
</channel>
```

## Nowdoc Representation of String Data Type

All the rules for heredoc identifiers also apply to nowdoc identifiers. A **nowdoc** is specified just like a **heredoc**, but there is no parsing inside a nowdoc. You can use the nowdoc construct for embedding large blocks of text without having to use any escape characters.

A nowdoc is identified with the same <<< sequence used for heredocs, but the identifier is enclosed in single quotes, e.g. <<<'EOT'. Nowdocs apply to single-quoted strings just the way heredocs apply to double-quoted strings.

Take a look at the following example −

</> Open Compiler

```
<?php
   echo <<<'IDENTIFIER'
   As the cat cleared its throat with a refined "Meow",
   the squirrel chirped excitedly about its latest
   discovery of a hidden stash of peanut treasure!
   IDENTIFIER;
?>
```

Run the code and check its output −

```
As the cat cleared its throat with a refined "Meow",
the squirrel chirped excitedly about its latest
discovery of a hidden stash of peanut treasure!
```

## Null Data Type in PHP

In PHP, null represents a special type that only has one value: NULL. Undefined and unset() variables will resolve to the value "null".

Programmers normally use the Null data type in PHP to initialize variables or to indicate that a value is missing.

To give a variable the NULL value, simply assign it like this −

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is **case insensitive;** you could just as well have typed −

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties −

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with IsSet() function.

**Note** – The data types of variables in PHP are determined at runtime based on the values that are assigned to them.

## Array Data Type in PHP

An array in PHP is an ordered map, a key is associated with one or more values. A PHP array is defined using the array() function, or with the use of a short notation where the data is put in square brackets.

Take a look at the following examples of **associative arrays** –

### Using the array() Function

```
$arr = array(
   "foo" => "bar",
   "bar" => "foo",
);
```

### Using the Short Notation

```
$arr = [
   "foo" => "bar",
   "bar" => "foo",
];
```

An array in PHP can also be defined with the "key-value pair" syntax. It is called an **indexed array**.

```
$arr = array("foo", "bar", "hello", "world");
```

In a **multi-dimensional array**, each element in the main array can also be an array. And, each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

**Note** – In PHP, compound data types are used to store collections of data, including arrays and objects.

## Object Data Type in PHP

An object type is an instance of a programmer-defined class, which can package up both other kinds of values and functions that are specific to the class.

To create a new object, use the **new statement** to instantiate a class –

```
class foo {
   function bar() {
      echo "Hello World.";
   }
}
$obj = new foo;
$obj->bar();
```

## Resource Data Type in PHP

Resources are special variables that hold references to resources external to PHP (such as a file stream or database connections).

Here is an example of **file resource** –

```
$fp = fopen("foo.txt", "w");
```

Data belonging to any of the above types is stored in a variable. However, since PHP is a dynamically typed language, there is no need to specify the type of a variable, as this will be determined at runtime.

## Example: The gettype() Function

The gettype() function is helpful to find out the type of data stored in a variable –

Open Compiler

```
<?php
   $x = 10;
   echo gettype($x) . "\n";

   $y = 10.55;
   echo gettype($y) . "\n";

   $z = [1,2,3,4,5];
   echo gettype($z);
?>
```

When you run this code, it will produce the following **output** –

```
integer
double
array
```