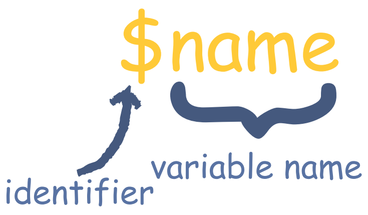
Variables and Data Types

This lesson discusses how you can use variables to store data and the various types of these variables, i.e., data types.

We'll cover the following

Variables

A variable in any programming language is a named piece of computer memory, containing some information inside. Variables are one of the essential parts of a computer program. You can declare a variable in PHP using a `$` sign followed by its name, e.g., `$myVariable`.



Consider the following PHP code where we store data in two variables and print them. The `$str` variable contains a **string**, and the `$num` variable contains an **integer** data type. We will discuss strings and integers later in the lesson.

```
<?php
$str = "I will be back by";
$num = 5;
```

```
echo $str;
echo " "; // Output: I will be back by
echo $num; // Output: 5
?>
```

Data types

Along with the name of variables, their sizes may vary too. In most programming languages (like C++ and Java) you can use different *data types* for your variables. PHP, however, does not have explicit type definitions, but the type of a variable is determined by the type of the value that it is assigned, or by the type that it is cast to.

There are different data types for different purposes, i.e., **null**, **boolean**, **integer**, **float**, **string**, **object**, **resource**, and an **array**. Here is a brief overview of the types. For a detailed documentation and examples, see the PHP documentation.

Null

Null can be assigned to any variable. It represents a variable with no value.

```
<?php
$foo = null;
?>
```

Assigning null to a variable invalidates it and its value is undefined or void if it's used. The variable is cleared from memory and deleted by the garbage collector.

Boolean

This is the simplest data type with only two possible values, i.e., `true` and `false`.

```
<?php
$foo = true;
$bar = false;
?>
```

Booleans can be used to control the flow of code. Take a look at the code snippet below as an example. In case you're not familiar with the given syntax you can learn more about it in a later lesson regarding if-else Statement:

```php
<?php
// ignore the if and else for now. We'll study more about them later.
$foo = true;

if ($foo) {
    echo "true";
} else {
    echo "false";
}
?>
```

Integer

An integer is a positive or negative number. It can be used with any number base (i.e., decimal, hexadecimal, octal, etc.). The size of an integer is platform-dependent.

PHP does not support unsigned integers. This means an integer can be positive or negative as each integer will contain information about its positivity or negativity. Consider the following code widget:

```php
<?php
$negative = -3; // negative
$zero = 0; // zero (can also be null or false (as boolean)
$positive = 123; // positive decimal
$zeroPos = 0123; //0 prefix is used to specify octal - octal value = 83 decimal
$hex = 0xAB; //0x prefix is used to specify hexadecimal - hexadecimal value = 171 decimal
$bin = 0b1010; // 0b prefix is used to specify binary - binary value = 10 decimal
var_dump($negative, $zero, $positive, $zeroPos, $hex, $bin);
?>
```

Float

Floating point numbers, “doubles” or simply called “floats” are decimal numbers.

```php
<?php
$foo1 = 1.23;
$foo2 = 10.0;
$bar1 = -INF; // -INF refers to negative infinity
$bar2 = NAN; // NAN stands for 'Not a Number'
var_dump($foo1,$foo2,$bar1,$bar2);
?>
```

Array

An array is like a list of values. The simplest form of an array is indexed by an integer, and ordered by the index, with the first element lying at index 0.

```
<?php
$foo = array(1, 2, 3); // An array of integers created using array fucntion
$bar = ["A", true, 123 => 5]; // Short array syntax, PHP 5.4+
echo $bar[0]; // Returns "A"
echo "\n";
echo $bar[1]; // Returns 1 for true
echo "\n";
echo $bar[123]; // Returns 5
// echo $bar[1234]; // uncommenting this line will give an error because 1234 is inaccessable
?>
```

Arrays can also associate a key other than an integer index to a value. In PHP, all arrays are *associative arrays* behind the scenes, but when we refer to an *associative array* explicitly, we usually mean one that contains one or more keys that aren't integers.

```
<?php
$array = array();
$array["foo"] = "bar";
$array["bar"] = "quux";
$array[42] = "hello";
echo $array["foo"]; // Outputs "bar"
echo "\n";
echo $array["bar"]; // Outputs "quux"
echo "\n";
echo $array[42]; // Outputs "hello"
?>
```

We will explore arrays in more detail in the chapter, Arrays in PHP.

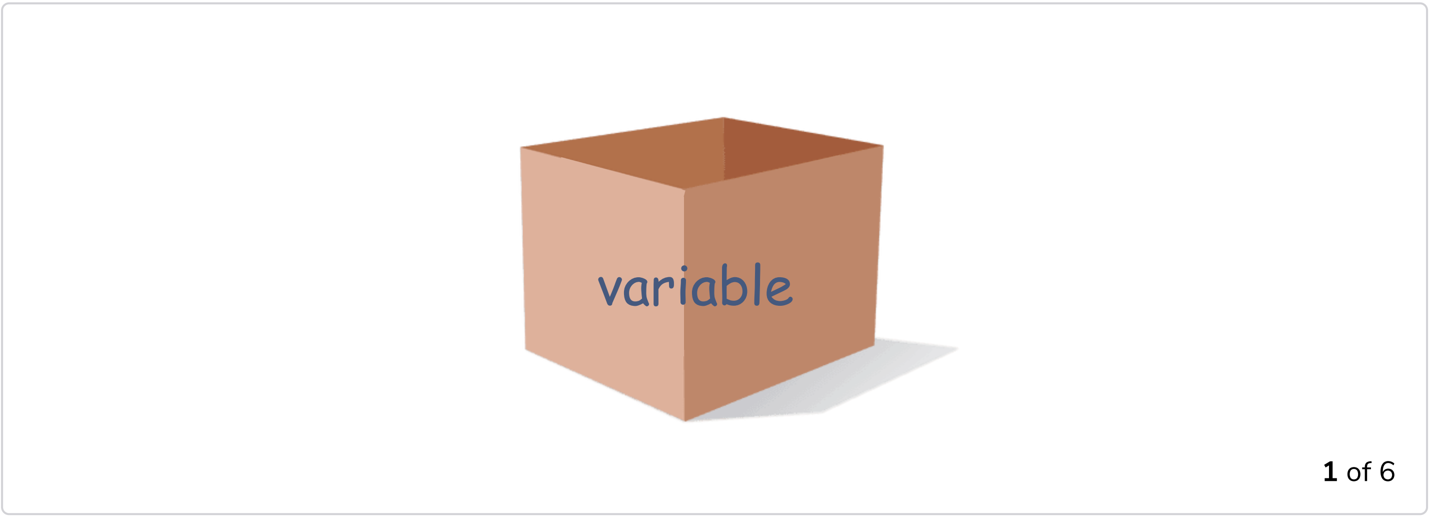
String

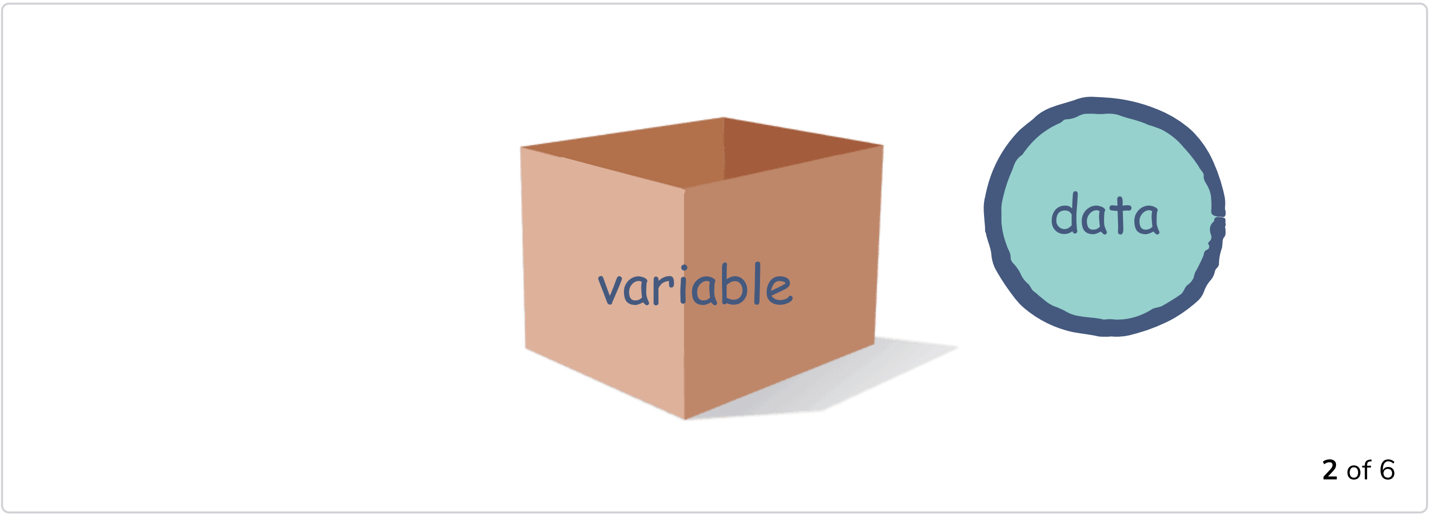
A string is like an array of characters. Like an array, a string can be indexed to return its individual characters:

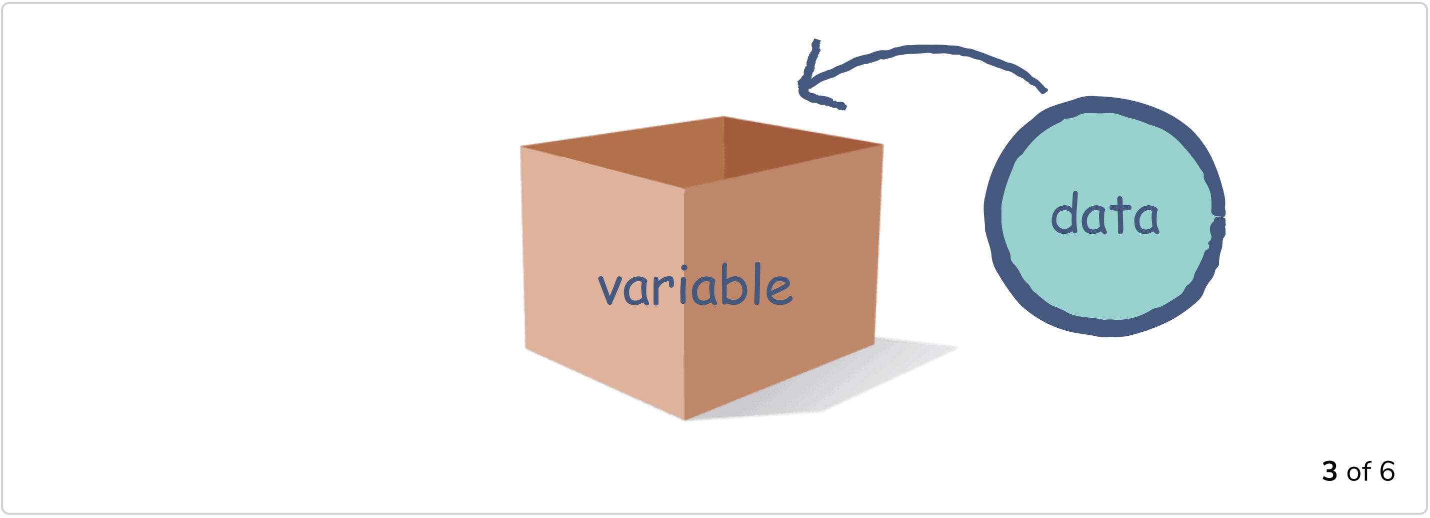
```
<?php
$foo = "I am a string";
echo $foo; // outputs "I am a string"
echo "\n";
echo $foo[3]; // Prints 'm', the third character of the string in $foo.
?>
```

We will explore strings in more detail in the chapter, Strings

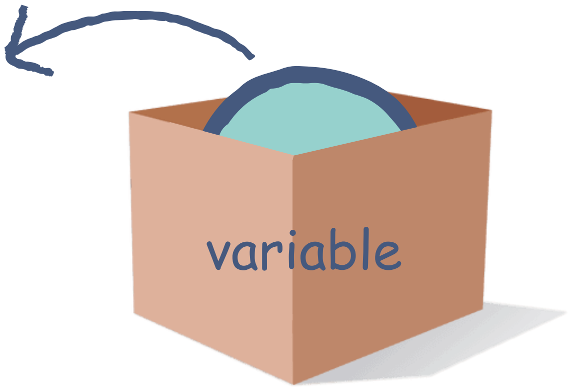
This whole process of declaring variables and storing data in these variables is summed up in the following figure:



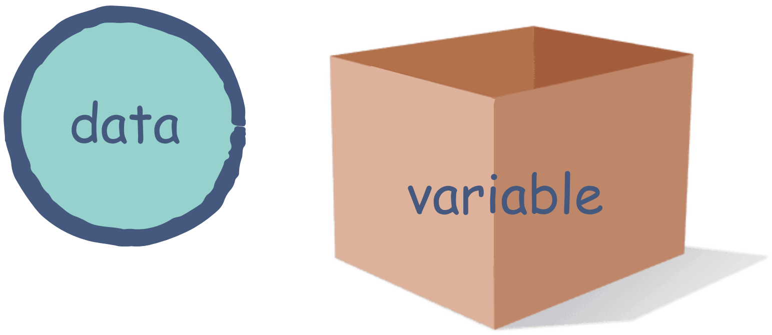




4 of 6



5 of 6



6 of 6

Now that you know all about variables and data types, let’s take a quiz in the next lesson.