Functions

Functions

A **function** is a block of statements that can be used repeatedly in a program.
A function will not execute immediately when a page loads. It will be executed by a call to the function.
A user defined function declaration starts with the word **function**:

```
function functionName() {
  //code to be executed
}
```

A function name can start with a letter or an underscore, but not with a number or a special symbol.

> Function names are NOT case-sensitive.

Functions

In the example below, we create the function **sayHello()**. The opening curly brace ({) indicates that this is the beginning of the function code, while the closing curly brace (}) indicates that this is the end.
To call the function, just write its name:

```
function sayHello() {
 echo "Hello!";
}

sayHello(); //call the function

//Outputs "Hello!"
```

Try It Yourself

> Tap **Try It Yourself** to play around with the code!

Function Parameters

Information can be passed to functions through **arguments**, which are like variables.
Arguments are specified after the function name, and within the parentheses.
Here, our function takes a number, multiplies it by two, and prints the result:

```
function multiplyByTwo($number) {
  $answer = $number * 2;
  echo $answer;
}
multiplyByTwo(3);
//Outputs 6
```

Try It Yourself

You can add as many arguments as you want, as long as they are separated with **commas**.

```
function multiply($num1, $num2) {
  echo $num1 * $num2;
}
multiply(3, 6);
//Outputs 18
```

Try It Yourself

> When you define a function, the variables that represent the values that will be passed to it for processing are called **parameters**. However, when you use a function, the value you pass to it is called an **argument**.

Default Arguments

Default arguments can be defined for the function arguments.
In the example below, we're calling the function **setCounter()**. There are no arguments, so it will take on the default values that have been defined.

```
function setCounter($num=10) {
   echo "Counter is ".$num;
}
setCounter(42);  //Counter is 42
setCounter();  //Counter is 10
```

Try It Yourself

> When using default arguments, any defaults should be on the right side of any non-default arguments; otherwise, things will not work as expected.

Return

A function can return a value using the **return** statement.
Return stops the function's execution, and sends the value back to the calling code.
For example:

```
function mult($num1, $num2) {
  $res = $num1 * $num2;
  return $res;
}

echo mult(8, 3);
// Outputs 24
```

Try It Yourself

> Leaving out the return results in a **NULL** value being returned.
> A function cannot return multiple values, but returning an **array** will produce similar results.

End.