For Loop

This lesson will teach the concept and implementation of for loops and nested for loops in PHP.

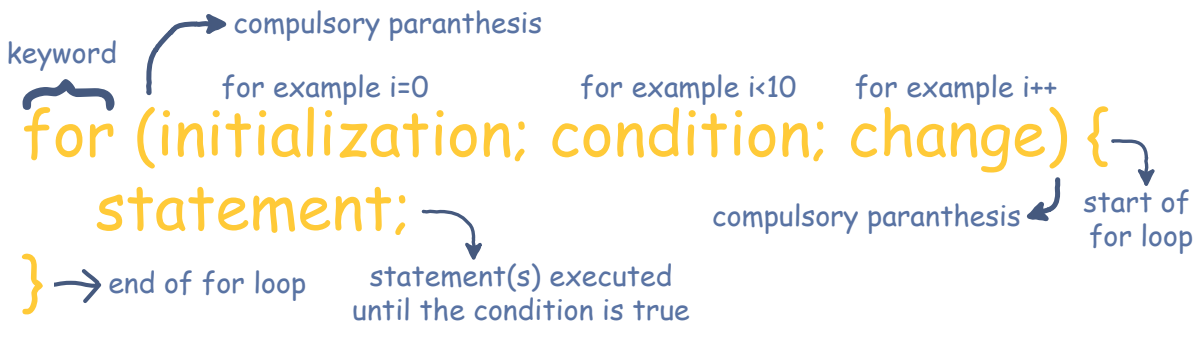
We'll cover the following

What is a for loop?

A for loop can also be used for doing things a certain amount of time. It's like a while loop but the *increment* is included with the *condition*.

Syntax

A `for` loop is set up like this:



Explanation

- **Initialization** - Initialize the variable that will be used as loop control variable.

- **Condition** - The *loop* only runs when the *condition* is **true**.
- **Increment** - How the *variable* changes every time the *loop* runs.

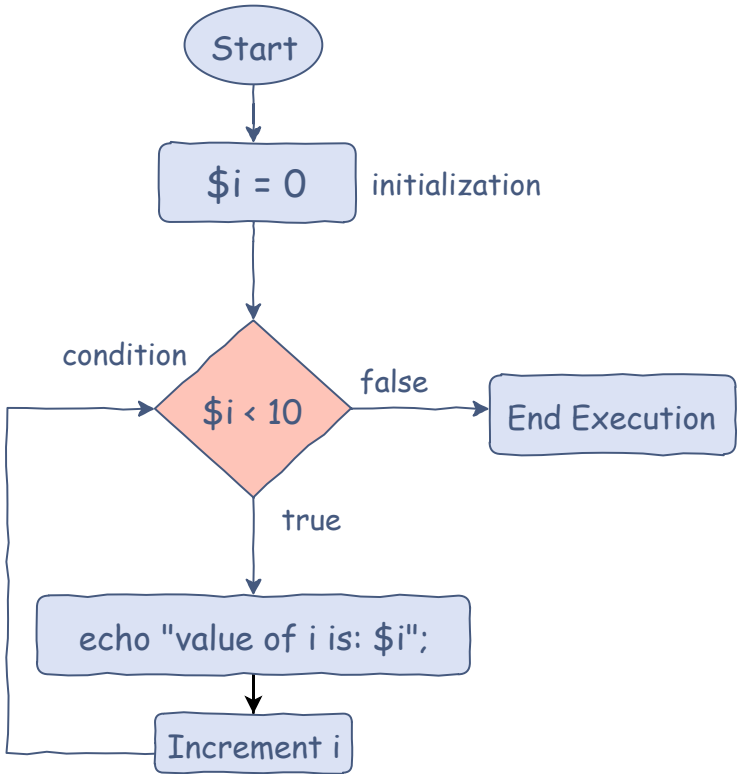
Example

Here's an example of how the `for` *loop* works:

```
<?php
for ($i = 0;$i < 10;$i++)
{
    echo "value of i is: $i\n";
}
?>
```

Explanation

Take a look at the illustration below to understand the code above more clearly.



Flow Chart for For Loop

What does the for loop do?

- Prior to the *first* iteration, it sets the value of `i` to **0**.
- Next, it tests (like a normal `while` loop) if `i` is *less* than **10**

- If the statement returns `true`, the body of the loop is run and the program will *print* the value returned by the simple arithmetic statement `i`.
- Next, the terminal cursor moves down to the next line.
- After the loop is finished, `i` is incremented (by **1**), as specified in the update statement, and the conditional is tested again.

So, this loop will run a total of **10** times, printing the **"i"** value each time. You've just taught your program to count! **Wow!**

The *variable* used in `for` loops is generally an *integer* variable named `i`, `j`, or `k`.

The loop in the above example can also be written as:

```
<?php
$i = 0;

for (;$i < 10;$i++)
{ //initialization can also be done outside loop
    echo "value of i is: $i\n";
}
?>
```

Similarly another method to write it is as follows:

```
<?php
$i = 0;

for (;$i < 10;)
{ //initialization can also be done outside loop. Note the semicolon is compulsory
    echo "value of i is: $i \n";
    $i++; //the increment of loop control variable can also be done separately

}
?>
```

Nested For Loops

It is possible to *nest* `for` loops. *Nesting* means including one `for` loop in another

`for` loop.

The syntax for a **nested** `for` loop is as follows:

```
for (expression for initialization ; expression for testing ; expression for updating ) {
   for (expression for initialization ; expression for testing ; expression for updating) {
      //body
   }
   //body
}
```

Example

Let's take a look at an example code to understand *nesting* of `for` loops better.

```
<?php
$input = 5;
echo "How many missiles will you fire?\n";
echo "I will fire: $input missiles\n";

for ($i = 0;$i < $input;$i++)
{ // outer for loop
    for ($j = 3;$j > 0;$j--)
    { // inner for loop
        echo "$j  ";
    }
    $temp = $i + 1;
    echo "Missile $temp has launched.\n";
}
echo "All missiles have been launched.\n";
?>
```

Explanation

In a nested `for` loop, for a single value of the **outer** loop, in this case, `i`, the inner (*nested*) `for` loop will iterate over all its values, that is, for example for `i=0` the inner (*nested*) loop will run from `j = 3` to `j=1`. After this is done, `i` will be incremented to **1** and the inner loop will again iterate over all its values against this value of `i`. The process continues till all values of `i` are iterated over.

Look at the illustration below which will help you visualize this and help you understand this concept more clearly.

input = 5

```
input = 5

How many missiles will you fire?
```

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles
```


```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = ?
```

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0

j = ?
```


```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0

j = 3
```

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0

j = 3
```

3

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0

j = 2
```

3

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0

j = 2
```

3 2

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0

j = 1
```

3 2

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0

j = 1
```

3 2 1

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0
```

3 2 1

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0
```

3 2 1

temp = 1


```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 0
```

3 2 1

temp = 1

Missile 1 has launced.

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1
```


```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1

j = 3
```

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1

j = 3
```

3

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1

j = 2
```

3

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1

j = 2
```

3 2

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1

j = 1
```

3 2

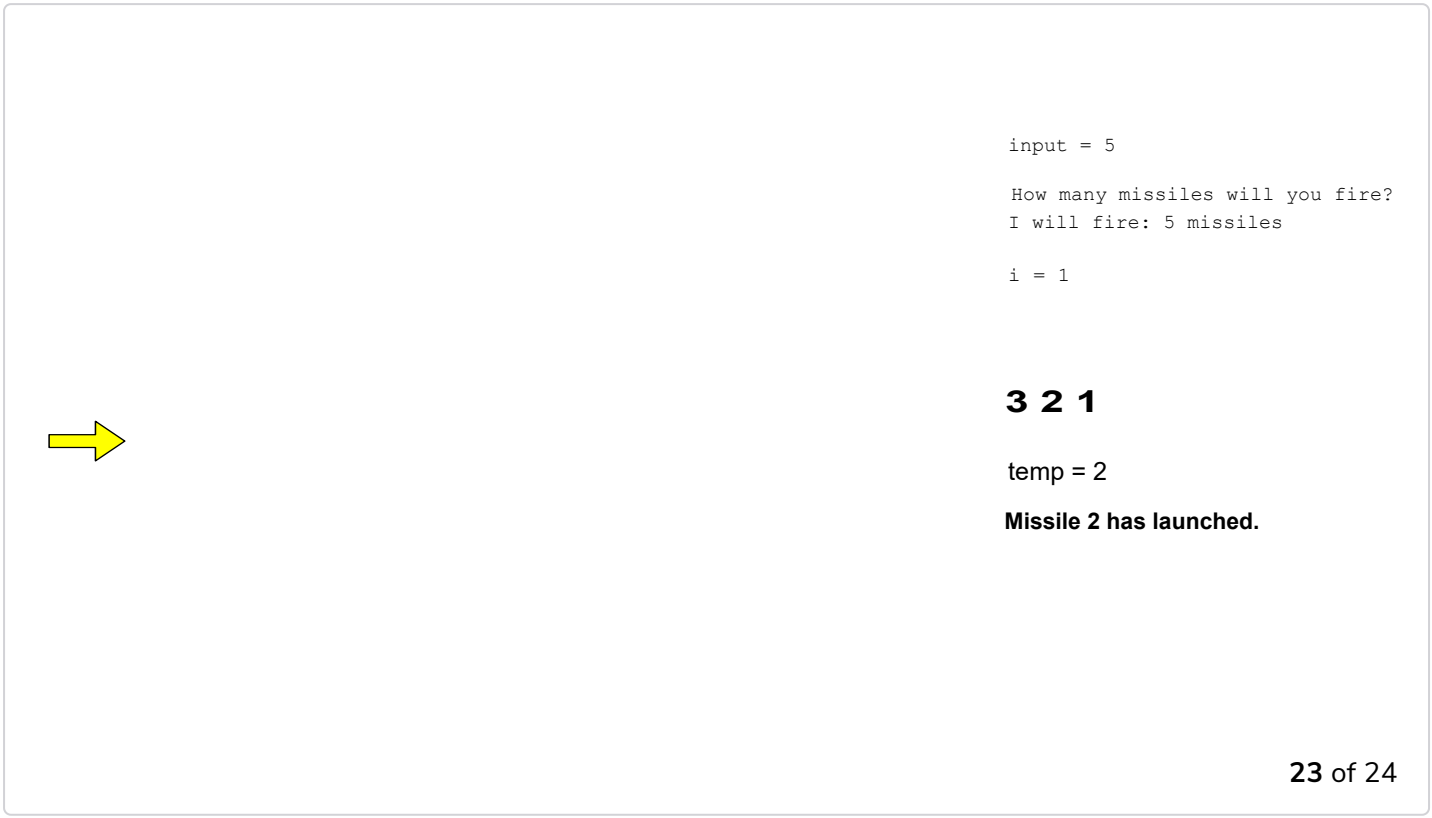
```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1

j = 1
```

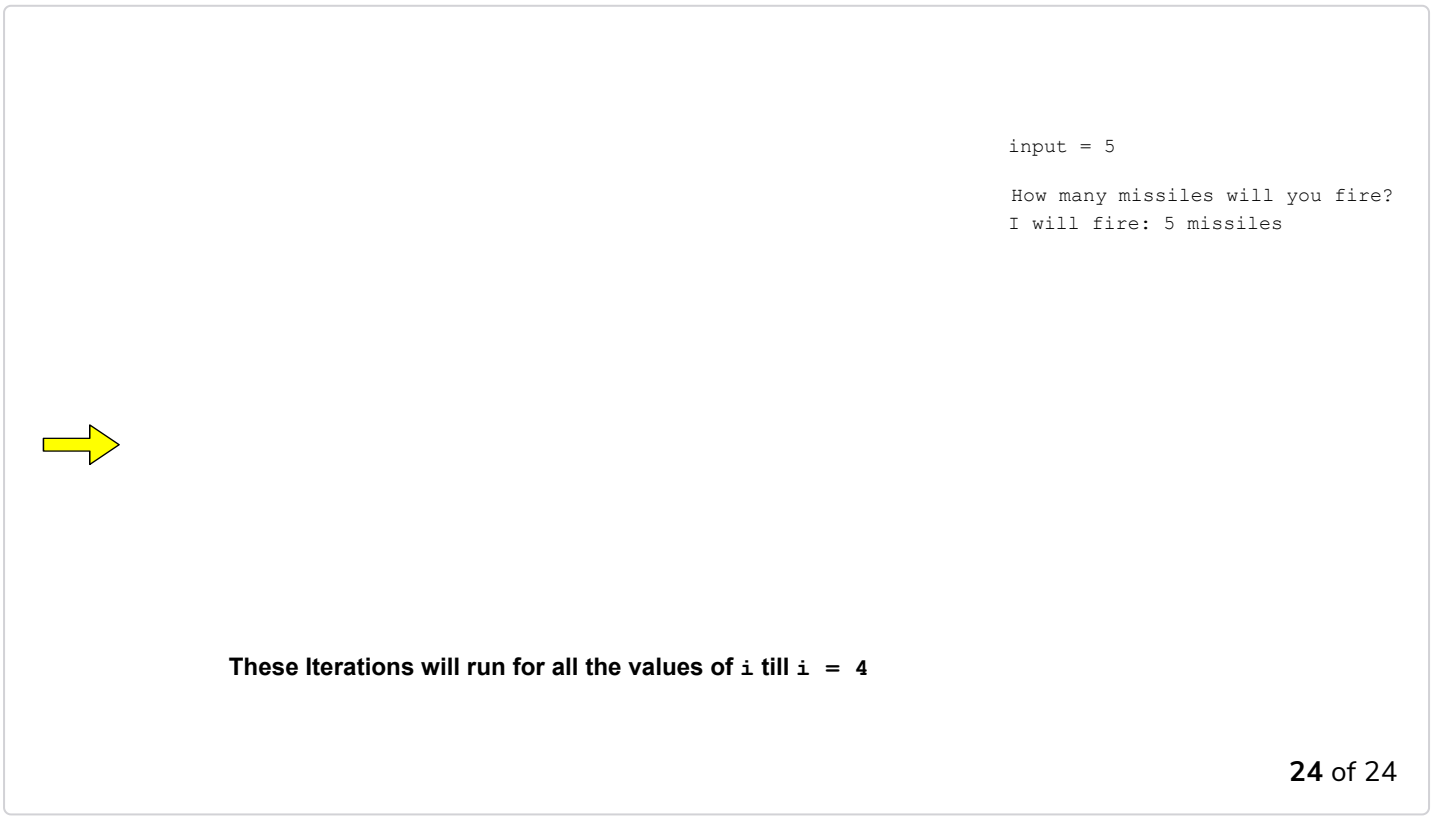
3 2 1

```
input = 5

How many missiles will you fire?
I will fire: 5 missiles

i = 1
```

3 2 1

temp = 2





Very interesting right? Now that the concept of `for` loops and *nested* `for` loops is clear let's look at *foreach* loop in the next lesson.