

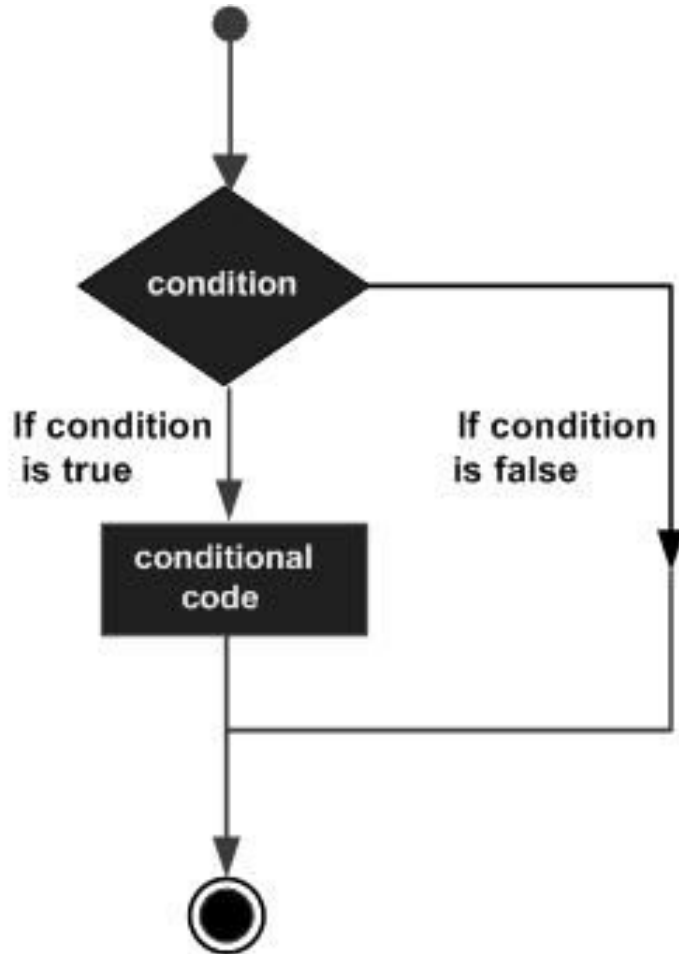
# Programming Basics

Vivify Academy



# Decision Making, Control flow





Control flow statements are used to take decision based on some condition. PHP supports the following three decision making statements:

- if statement
- else statement
- switch statement

# What's a “condition”?

In programming languages, a condition is any expression that can be evaluated as **true** or **false**. In most cases these expressions are boolean values, but in PHP they can be any type of value. A condition is said to be “true” if its value can be converted to the boolean **true**. Examples would be:

- a boolean **true**,
- some number other than zero,
- a non-empty string, etc.



# The **if** statement

Use this statement if you want to execute a piece of code only when some condition is true.

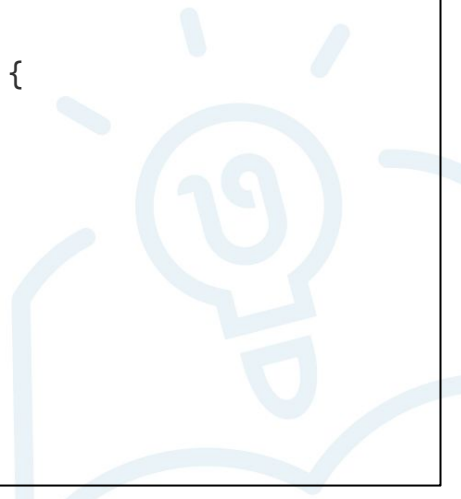


# Pseudocode

```
if (condition)  
    code to be executed if condition is true;
```

# PHP

```
<?php  
  
// this gives us a string with the current day:  
$today = date("l");  
  
if ($today === "Friday") {  
    echo "Finally!";  
}  
  
?>
```



# What is pseudocode?

Alternatively referred to as p-code, pseudocode is a simplified programming language that resembles plain English, that cannot be compiled or executed, but explains a resolution to a problem.



# The **if** - **else** statement

Use this statement if you want to execute a set of code when a condition is true and another if the condition is not true. You can't use **else** without **if**.





# Pseudocode

```
if (condition)
    code to be executed if condition is true;

else
    code to be executed if condition is false;
```


# PHP

```
<?php

$today = date('l');

if ($today === "Friday") {
    echo "Have a nice weekend!";
} else {
    echo "Have a nice day!";
}

?>
```



# The **else if** statement

The **else if** statement is used with the if - else statement to execute a set of code if one of the several condition is true.



# Pseudocode

```
if (condition)
    code to be executed if condition is true;

else if (condition)
    code to be executed if condition is true;

else
    code to be executed if condition is false;
```


# PHP

```
<?php

$today = date('l');

if ($today === "Friday") {
    echo "Have a nice weekend!";
} else if ($today === "Sunday") {
    echo "Have a nice Sunday!";
} else {
    echo "Have a nice day!";
}

?>
```



# The **switch** Statement

The **switch** statement is used if you want to select one of many blocks of code (marked with **case**) to be executed. The switch statement is used to avoid long blocks of **if..elseif..else** code.



# Pseudocode

```
switch (expression) {  
    case label1:  
        code to be executed if expression = label1;  
        break;  
  
    case label2:  
        code to be executed if expression = label2;  
        break;  
  
    default:  
        code to be executed  
        if expression is different  
        from both label1 and label2;  
}  

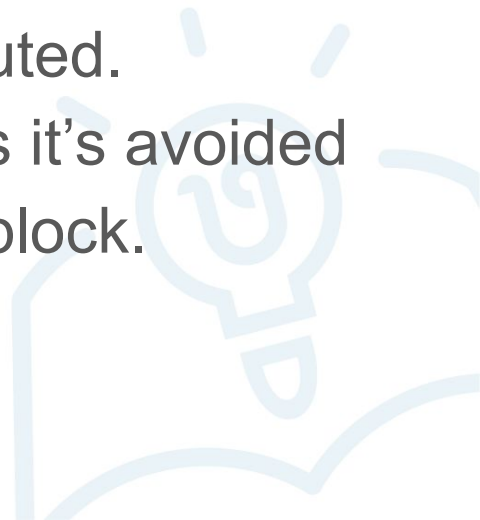
```

# PHP

```
<?php  
  
$today = date('l');  
  
switch ($today) {  
    case "Monday":  
        echo "Today is Monday";  
        break;  
  
    case "Tuesday":  
        echo "Today is Tuesday";  
        break;  
  
    default:  
        echo "Which day is this ?";  
}  
  
?>
```

# The **break** Statement

The **break** statement is used to exit the surrounding **switch** statement or loop (as we'll see a bit later). In the case of our **switch** example, without the **break** statement after each block, more than one **case** block could be executed. Sometimes this can be useful, but in most cases it's avoided and there's a **break** statement after each **case** block.



# Loops



# What is a loop?

A loop is code that describes repetition of the same instructions or processes, over and over until receiving the order to stop.

If not handled properly a loop can cause the computer to become slower as it becomes overwhelmed with having to repeat the same steps over and over causing it to get stuck in an endless loop.



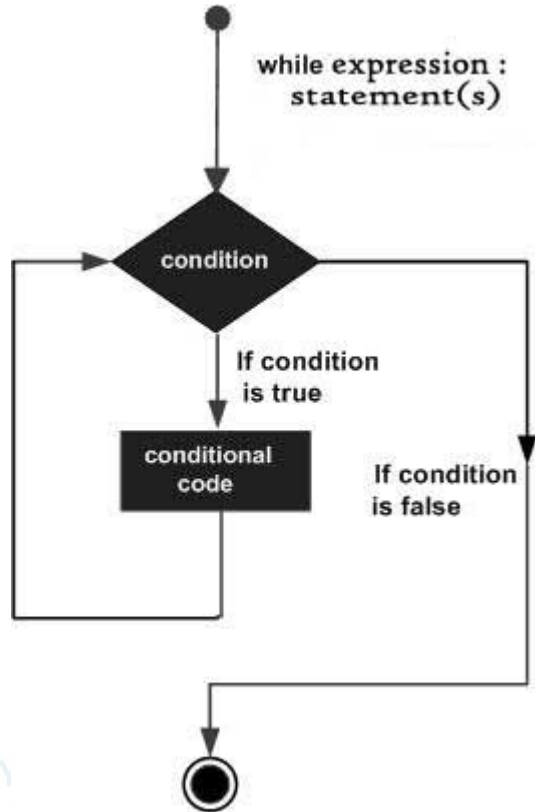
# Loops in PHP

Loops in PHP are used to execute the same block of code a specified number of times. This block of code is called the *body* of the loop. PHP supports following four loop types:

- **while** – loops through a block of code if and as long as a specified condition is true,
- **do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true,
- **for** – loops through a block of code a specified number of times,
- **foreach** – loops through a block of code for each element in an array.

We will discuss about **continue** and **break** keywords used to control the loops execution.

# The **while** loop statement



The **while** statement will execute a block of code if, and as long as a test expression is true.

If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

# Pseudocode

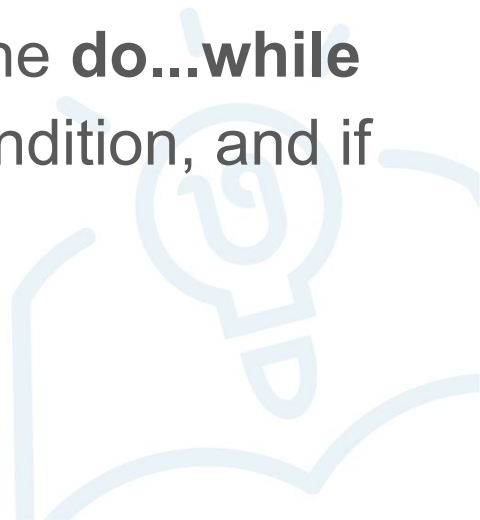
```
while (condition) {  
    code to be executed;  
}
```

# PHP

```
<?php  
  
$i = 0;  
  
// as long as $i is less than 10...  
while ($i < 10) {  
    // increase $i by 1  
    $i++;  
}  
  
echo "Loop stopped, value of variable i is $i";  
  
?>
```

# The **do...while** loop statement

The **do...while** statement will execute a block of code at least once, and then it will repeat the loop as long as a condition is true. Unlike the **while** loop, which checks the condition and then executes the code if the condition is true, the **do...while** loop first executes the code, then checks the condition, and if it's true it goes back to the beginning.



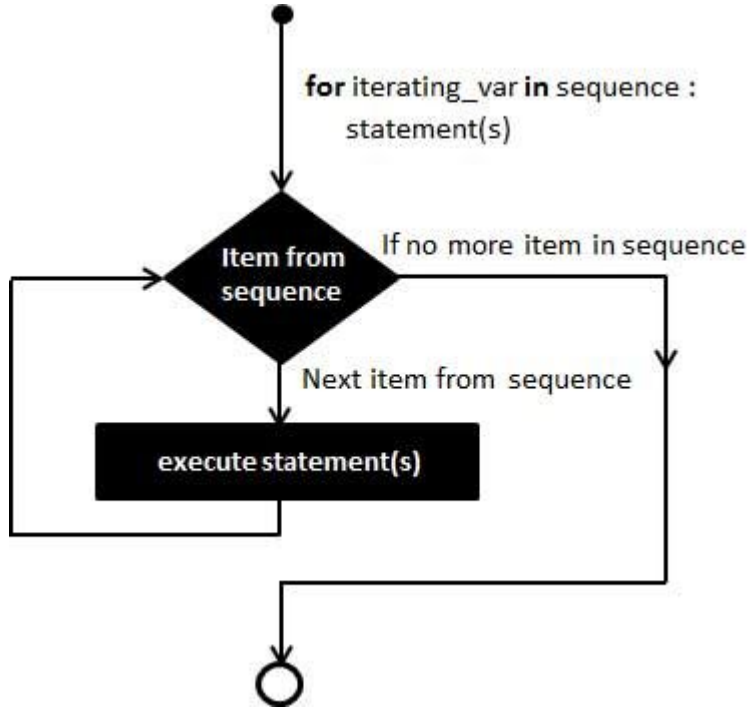
# Pseudocode

```
do {  
    code to be executed;  
}  
while (condition);
```

# PHP

```
<?php  
  
$i = 0;  
  
do {  
    // increase $i by 1...  
    $i++;  
  
    // as long as $i is less than 10  
} while ($i < 10);  
  
echo "Loop stopped, value of variable i is $i";  
  
?>
```

# The **for** loop statement



The **for** statement is used when you know how many times you want to execute a statement or a block of statements. Any **for** loop can be written as **while** and vice-versa, but the **for** loop is better for working with numbers.

# Pseudocode

```
for (initialization; condition; step) {  
    code to be executed;  
}
```

# PHP

```
<?php  
  
$a = 0;  
  
for ($i = 0; $i < 10; $i++) {  
    $a += 10;  
}  
  
echo "Loop stopped, value of variable i is  
$i<br>";  
echo "Value of variable a is $a";  
  
?>
```

# Equivalence of **for** and **while**

## for

```
for ($i = 0; $i < 5; $i++) {  
    // do something...  
}
```

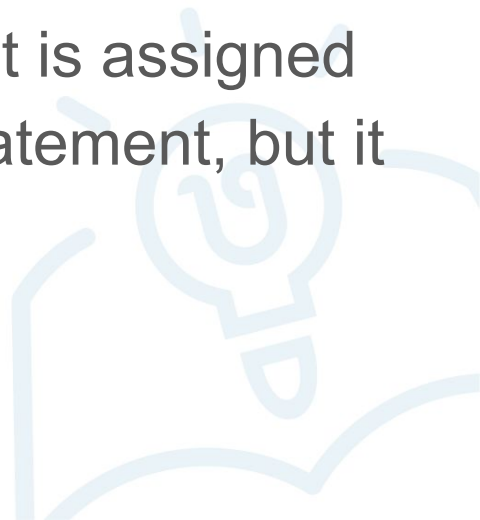
## while

```
$i = 0;  
while ($i < 5) {  
    // do something...  
    $i++;  
}
```



# The **foreach** loop statement

The **foreach** statement is used to loop through *arrays*, which are special values in PHP consisting of an ordered group of *elements*. They will be covered in more detail tomorrow. For each step, the value of the current array element is assigned to \$value. It's a simpler alternative for the **for** statement, but it can't always be used.



# Pseudocode

```
foreach (array as value) {  
    code to be executed;  
}
```

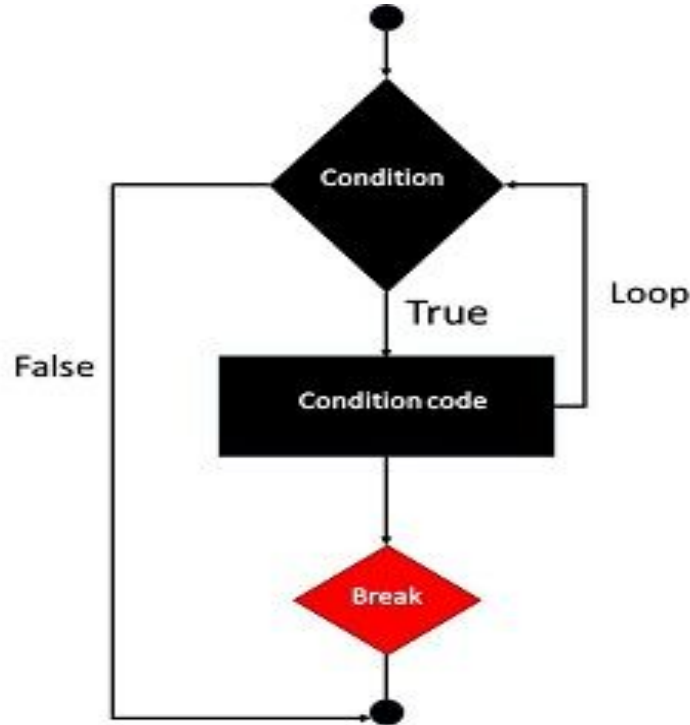
# PHP

```
<?php  
  
// this is how we create a simple array:  
$array = [1, 2, 3, 4, 5];  
  
// and here we go through all the elements:  
foreach ($array as $value) {  
    echo "Value is $value <br>";  
}  
  
?>
```

# Special statements in loops



# The **break** statement



The PHP **break** keyword is used to terminate the execution of a loop prematurely (or break out of a **switch**).

The break statement is placed inside the statement block, and when encountered the program flow jumps out of the loop. It gives you full control and allows you to exit from the loop even when the loop condition is still true.

# The break statement - example

```
<?php

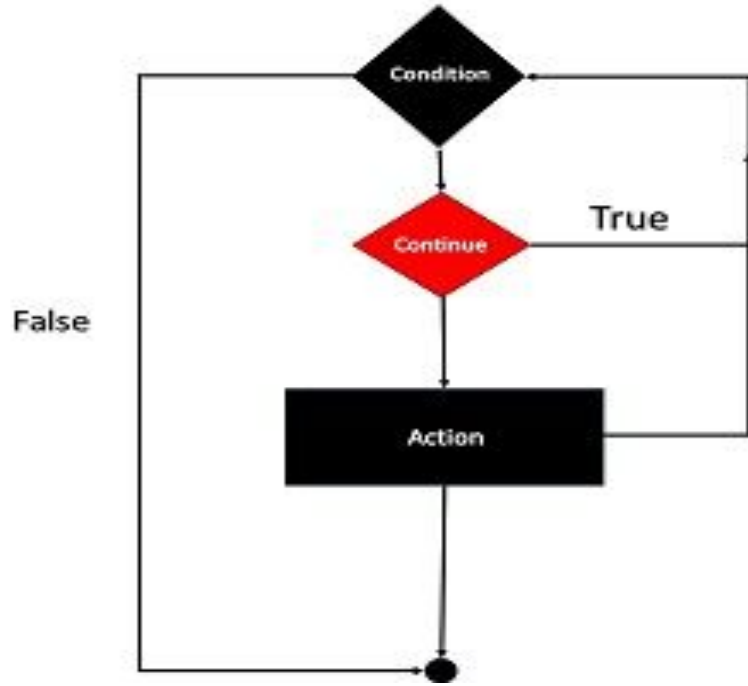
$number = 71;
$prime = true;

$i = 2;
while ($i < $number) {
    if (($number % $i) === 0) {    // when $number is divided by $i remainder is 0
        $prime = false;
        break;
    }
    $i++;
}

if ($prime) {
    echo "$number is a prime number!";
} else {
    echo "$number is divisible by $i!";
}
```



# The continue statement



The PHP continue keyword is used to stop the current iteration of a loop and jump immediately to the next one, but it does not terminate the entire loop.

Just like the break statement the continue statement is placed inside the statement block containing the code that the loop executes. When the program flow encounters the continue statement, the rest of the code in the loop body is skipped and next pass starts.

# The continue statement - example

```
<?php
```

```
$start = 3;
```

```
$end = 19;
```

```
$evenNumbers = 0;
```

```
for ($i = $start; $i <= $end; $i++) {
```

```
    if ($i % 2 === 1) {
```

```
        continue;
```

```
    }
```

```
    $evenNumbers++;
```

```
}
```

```
echo "There are $evenNumbers even numbers between $start and $end.";
```



# Useful links

- <http://php.net/manual/en/control-structures.if.php>
- <http://php.net/manual/en/control-structures.else.php>
- <http://php.net/manual/en/control-structures.elseif.php>
- <http://php.net/manual/en/control-structures.switch.php>
- <http://php.net/manual/en/control-structures.while.php>
- <http://php.net/manual/en/control-structures.do.while.php>
- <http://php.net/manual/en/control-structures.for.php>
- <http://php.net/manual/en/control-structures.foreach.php>
- <http://php.net/manual/en/control-structures.break.php>
- <http://php.net/manual/en/control-structures.continue.php>





# Exercise!

