if-else Statement

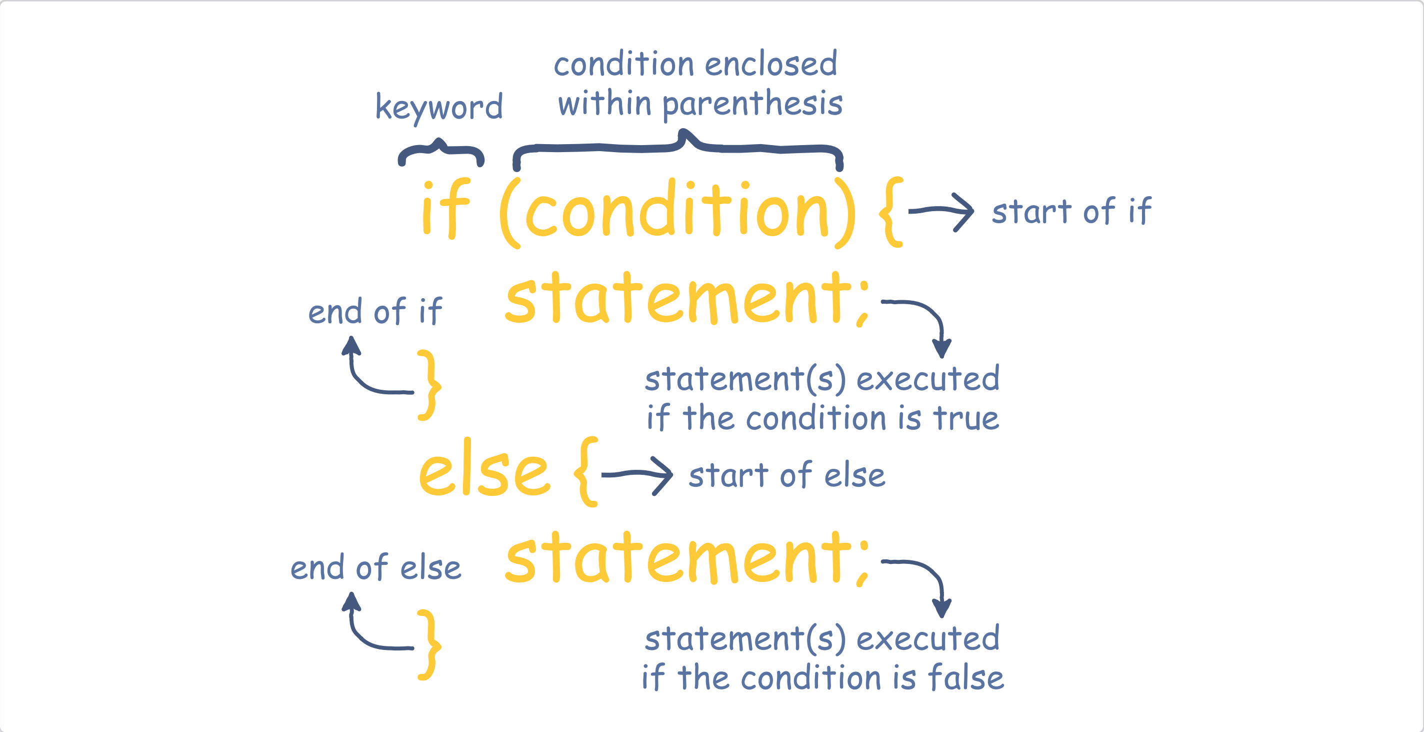
This lesson discusses if-else statements in detail including nested-ifs using examples.

We'll cover the following

Branches in PHP

Programming in general often requires a *decision* or a *branch* within the code to account for how the code operates under different inputs or conditions.

Within the **PHP** programming language, the simplest and sometimes the most useful way of creating a branch within your program is through an `if-else` statement.



keyword
condition enclosed within parenthesis
if (condition) {
start of if
statement;
end of if
}
statement(s) executed if the condition is true
else {
start of else
statement;
end of else
}
statement(s) executed if the condition is false

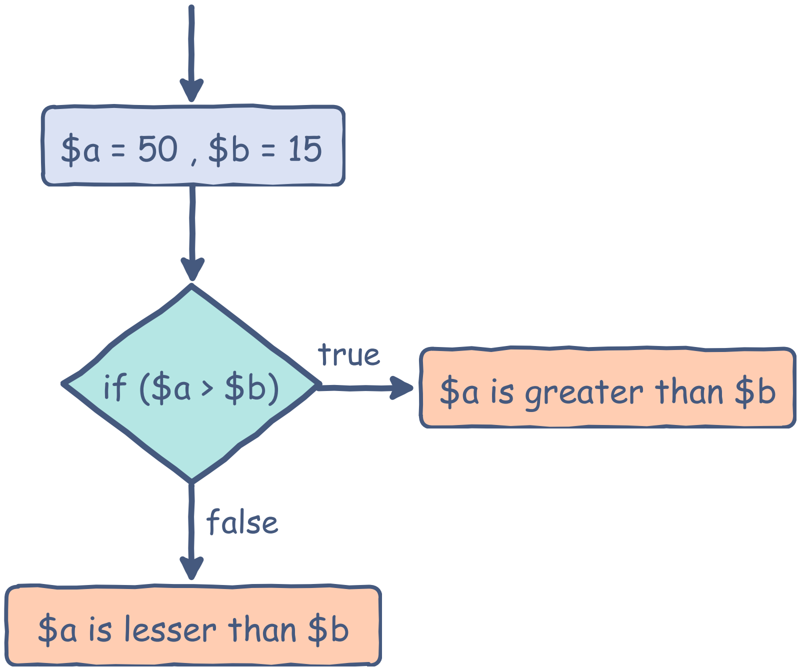
if-else

if-else

Let’s start by taking a look at an example which checks if a variable `a` is greater than or less than `b`:

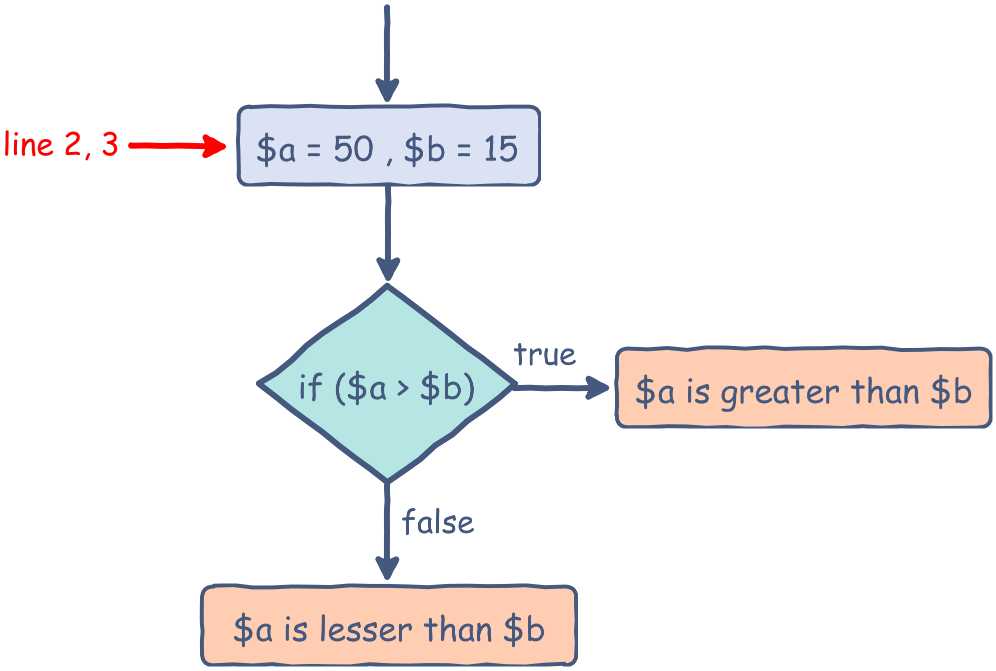
```
<?php
$a = 50; //change the value of "a" so its less than b in order to execute the else statement
$b = 15;
if ($a > $b)
{
    //this code is executed only if $a is greater than $b
    echo "a is greater than b";

}
else
{
    //this code is executed if the preceding "if" condition evaluated to false
    echo "a is less than b";
}
?>
```

The figure below illustrates how the working of the above code snippet using a flow chart:



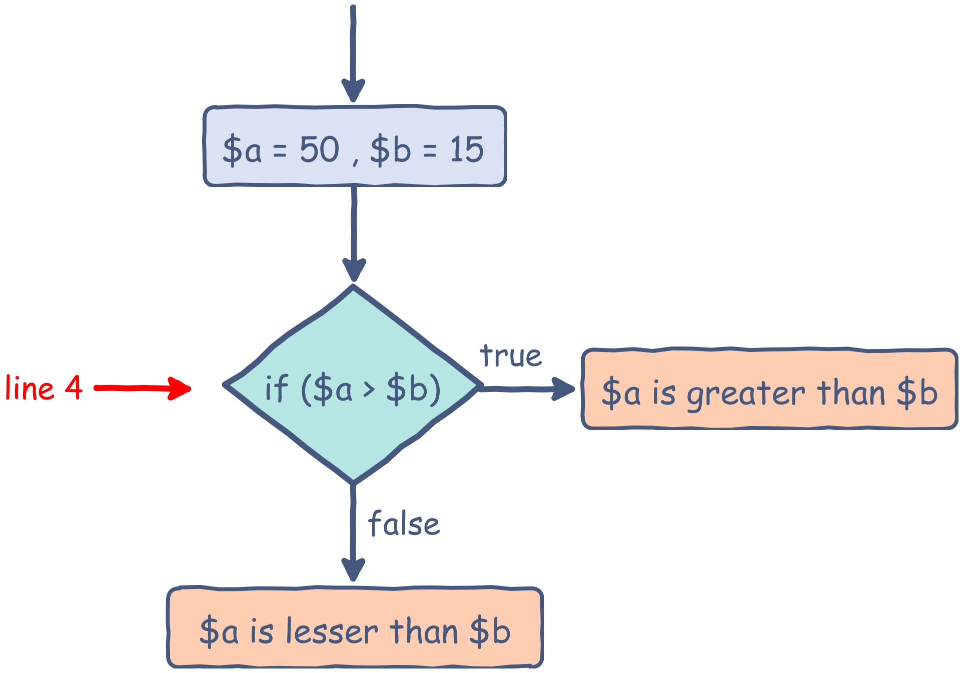
Flowchart of if-else variation

1 of 7



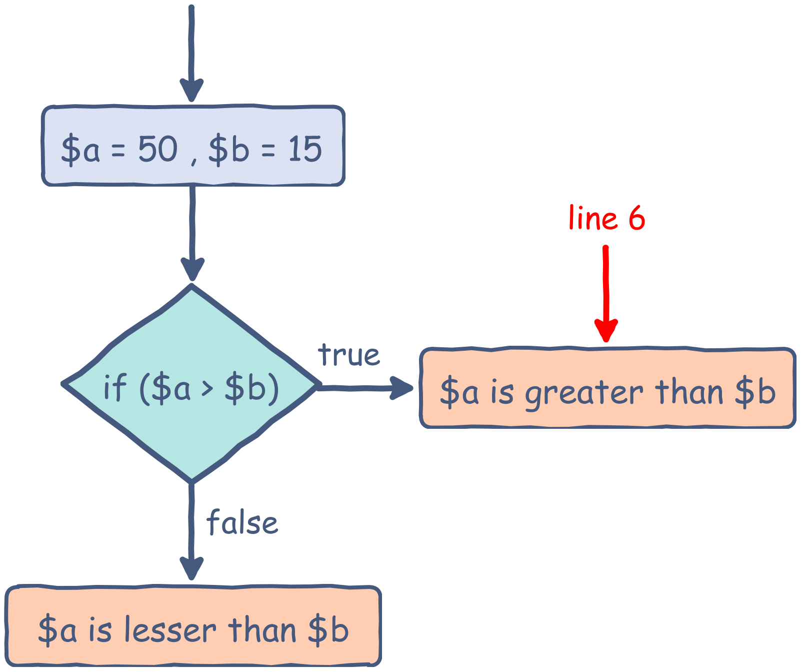
Flowchart of if-else variation

2 of 7



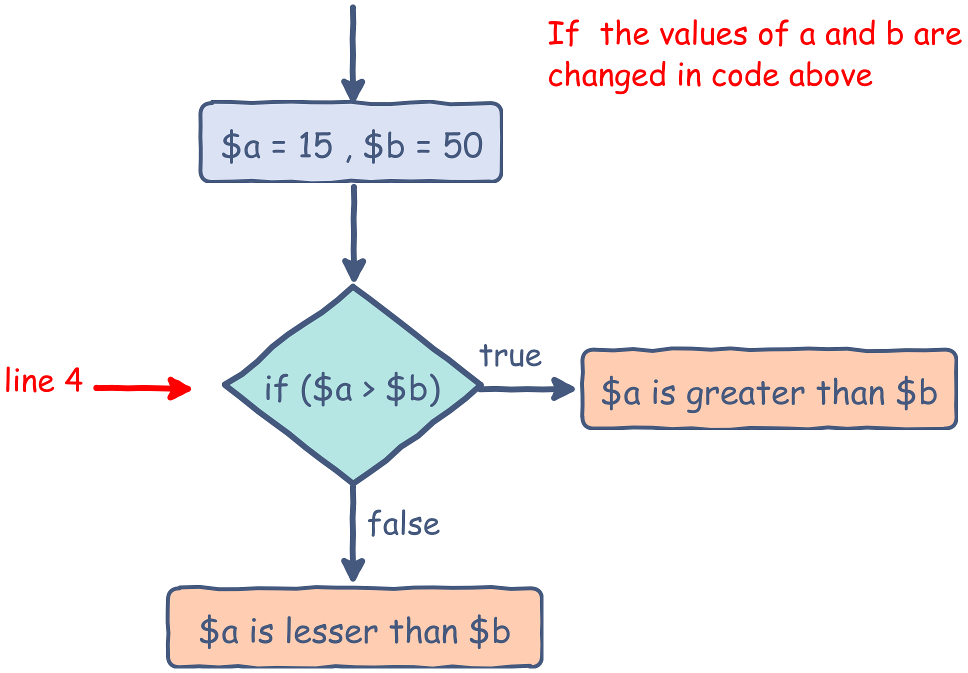
Flowchart of if-else variation

3 of 7



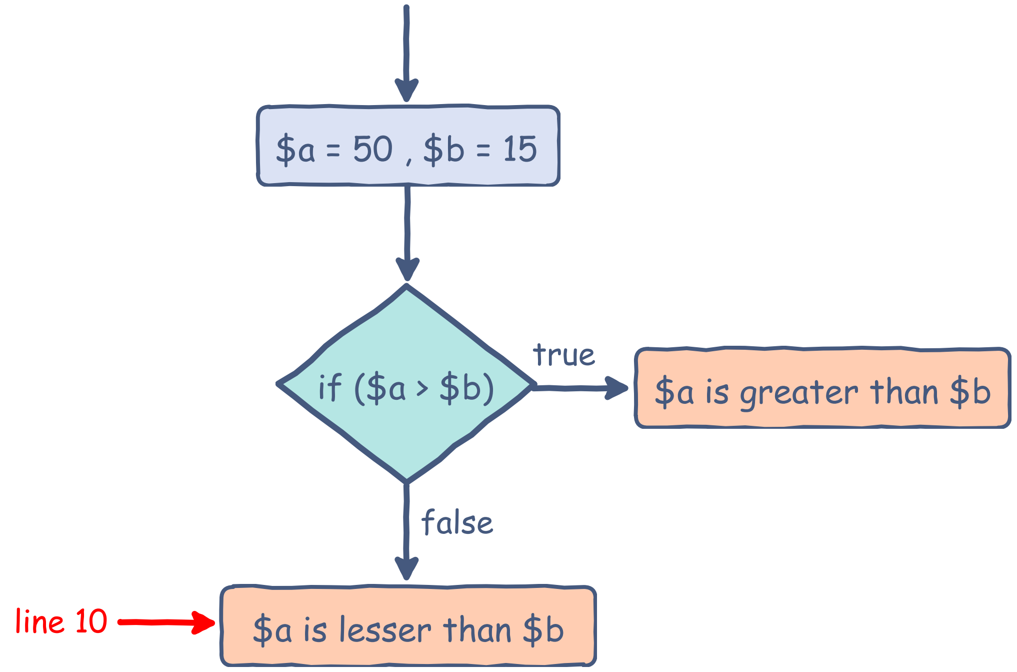
Flowchart of if-else variation

4 of 7



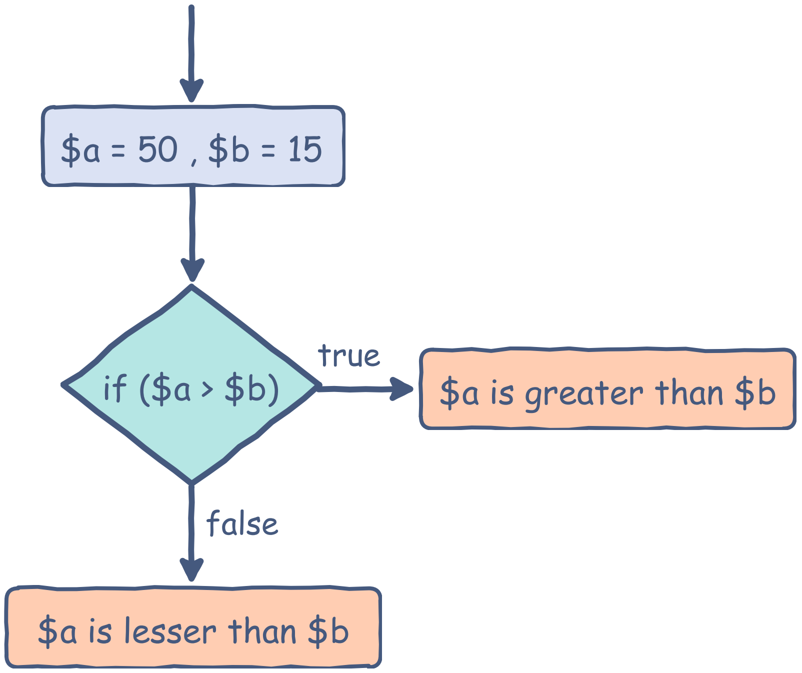
Flowchart of if-else variation

5 of 7



Flowchart of if-else variation

6 of 7



Flowchart of if-else variation

7 of 7

Explanation

Line 4:

- In this line note that there is space between the keyword `if` and the *opening parenthesis*.
- Inside the parentheses is the **condition**; in this case, it is a comparison using **greater** than operator between two values.
- It is a good practice to use the **same** type of *arguments* (not comparing *floating-point* values to *characters*).
- Note also the **left** *curly brace* `{`. This symbol denotes a block of *multiple* lines of code. Without it, the conditional would only refer to the statement immediately following it.
- It is a good practice to always use the braces.

Line 6:

- This comment represents the *body* of the conditional statement.

Line 8:

- The *right* curly brace is **essential**; it matches the *opening brace* on line **4** and signals the *end* of the `if` body.
- This line is optional. If there is a sort of **"default"** behavior that should be carried out, it would be placed here.
- The `else` clause does not belong by itself, only directly following an `if` clause.

Line 10:

- This is the body of the `else` clause.

Line 11:

- This *curly* brace is also **essential**; it matches the *opening* brace on line **8** and signals the end of the `else` body.

The body of the `else` block can be another `if` statement. This is known as **"nested conditionals"** because the conditionals are indeed nested; that is, placed inside of one another.

Nested-If

Here one `if-else` block is nested within another. One has to be careful in closing the inner `if` before closing the outer `if`.

```
if(condition1){
  //execution statement(s)
  if(condition2){
    //execution statement(s)
  }//end of inner if
  else{
    //execution statement(s)
 }//end  of inner else
}//end ofouter if
else{
  //execution statement(s)
}//end ofouter else
```

Explanation

In the code block above if

- `condition1` is **true** then
  - after executing the **execution statement(s)**, code flows to the `condition2` check.
  - Based on the result either **execution statement(s)** of the nested `if` or `else` are executed.
- `condition1` is **false** then
  - code flows to `else` statement in **line 13** and the **execution statements** of this `else` get executed.

Now let’s take a look at the `if-elseif-else` statements in the upcoming lesson.