Chapter 3

Logic: Making Decisions and Repetition

Boolean Expressions

- **Every expression** in a PHP program can be evaluated to **true or false**.

- All **non-zero integers or floating-point** numbers evaluated to **true**
- **zero (0 or 0.0)** is evaluated to **false**
- **All non-empty strings** are evaluated to **true**
- **Empty string,** or **string containing zero ('0') is** evaluated to **false**
- A variable containing **null** is evaluated to **false**

Evaluated to False	Evaluated to True
$v1 = 0; or $v1 = 0.0;	$v1 = 7; or $v1 = 5.78;
$v2 = '';	$v2 = 'Saeed';
$v3 = '0';	$v3 = true;
$v4 = null;	
$v5 = false;	

If – Statement

Example 3-1. Making a decision with if()

```
if ($logged_in) {
   print "Welcome aboard, trusted user.";
}
```

```
if ($logged_in) {
    print "Welcome aboard, trusted user.";
} else {
    print "Howdy, stranger.";
}
```

```
print "This is always printed.";
if ($logged_in) {
    print "Welcome aboard, trusted user.";
    print 'This is only printed if $logged_in is true.';
}
print "This is also always printed.";
```

elseif – Statement

```php
if ($logged_in) {
    // This runs if $logged_in is true
    print "Welcome aboard, trusted user.";
} elseif ($new_messages) {
    // This runs if $logged_in is false but $new_messages is true
    print "Dear stranger, there are new messages.";
} elseif ($emergency) {
    // This runs if $logged_in and $new_messages are false
    // but $emergency is true
    print "Stranger, there are no new messages, but there is an emergency.";
}
```

```php
if ($logged_in) {
    // This runs if $logged_in is true
    print "Welcome aboard, trusted user.";
} elseif ($new_messages) {
    // This runs if $logged_in is false but $new_messages is true
    print "Dear stranger, there are new messages.";
} elseif ($emergency) {
    // This runs if $logged_in and $new_messages are false
    // but $emergency is true
    print "Stranger, there are no new messages, but there is an emergency.";
} else {
    // This runs if $logged_in, $new_messages, and
    // $emergency are all false
    print "I don't know you, you have no messages, and there's no emergency.";
}
```

Comparison Operators

==, != , <, >, <=, >=

```
if ($new_messages == 12) {
    print "It seems you now have twelve new messages.";
}
```

```
if ($new_messages != 10) {
    print "You don't have ten new messages.";
}
```

```
if ($age > 17) {
    print "You are old enough to download the movie.";
}
if ($age >= 65) {
    print "You are old enough for a discount.";
}
if ($celsius_temp <= 0) {
    print "Uh-oh, your pipes may freeze.";
}
if ($kelvin_temp < 20.3) {
    print "Your hydrogen is a liquid or a solid now.";
}
```

Assignment vs. Comparison

- Be careful not to use = when you mean ==

```
if ($new_messages = 12) {
    print "It seems you now have twelve new messages.";
}
```

- One way to avoid using = instead of == is to put the variable on the right side of the comparison

```
if (12 == $new_messages) {
    print "You have twelve new messages.";
}
```

Floating-point numbers comparison

- Floating-point number may be stored slightly different
  - For example: 50.0 is stored as 50.000000002
- To compare two floating-point numbers, check whether the two numbers differ by less than some acceptably small threshold

```
if(abs($price_1 - $price_2) < 0.00001) {
    print '$price_1 and $price_2 are equal.';
} else {
    print '$price_1 and $price_2 are not equal.';
}
```

Comparison Operators - Strings

- The <, <=, ==, >=, and > operators can be used with **numbers** or **strings**.

```
if ($word < 'baa') {
    print "Your word isn't cookie.";
}
if ($word >= 'zoo') {
    print "Your word could be zoo or zymurgy, but not zone.";
}
```

- When the PHP engine sees strings containing **only numbers**, it **converts them to numbers** for the comparison.

```
// These values are compared using dictionary order
if ("x54321"> "x5678") {
    print 'The string "x54321" is greater than the string "x5678".';
} else {
    print 'The string "x54321" is not greater than the string "x5678".'
}

// These values are compared using numeric order
if ("54321" > "5678") {
    print 'The string "54321" is greater than the string "5678".';
} else {
    print 'The string "54321" is not greater than the string "5678".';
}
```

Comparison Operators - Strings

- the PHP engine converts the **string 6 pack to the number 6**, and then compares it to the number 55 using numeric order. Since 6 is less than 55, the less than test returns true.

```
// These values are compared using numeric order
if ('6 pack' < 55) {
    print 'The string "6 pack" is less than the number 55.';
} else {
    print 'The string "6 pack" is not less than the number 55.';
}
```

- **strcmp()** compares string using dictionary order without any converting to numbers

```
$x = strcmp("x54321","x5678");
if ($x > 0) {
    print 'The string "x54321" is greater than the string "x5678".';
} elseif ($x < 0) {
    print 'The string "x54321" is less than the string "x5678".';
}
```

spaceship operator (<=>)

- The **spaceship operator (<=>)** does comparison **similar to strcmp(),** but for **any data type**.
  - It evaluates to a **negative** number when its left-hand operand is **less than** the righthand operand,
  - a **positive** number when the righthand operand is **bigger**,
  - and **0** when they are **equal**.

```
// $a is a negative number since 1 is less than 12.7
$a = 1 <=> 12.7;

// $b is a positive number since "c" comes after "b"
$b = "charlie" <=> "bob";

// Comparing numeric strings works like < and >, not like strcmp()
$x = '6 pack' <=> '55 card stud';
if ($x > 0) {
    print 'The string "6 pack" is greater than the string "55 card stud".';
} elseif ($x < 0) {
    print 'The string "6 pack" is less than the string "55 card stud".';
}

// Comparing numeric strings works like < and >, not like strcmp()
$x ='6 pack' <=> 55;
if ($x > 0) {
    print 'The string "6 pack" is greater than the number 55.';
} elseif ($x < 0) {
    print 'The string "6 pack" is less than the number 55.';
}
```

Negation (!)

- To negate a truth value, use !
- Putting ! before an expression is like testing to see whether the expression equals false.

```
// The entire test expression ($finished == false)
// is true if $finished is false
if ($finished == false) {
    print 'Not done yet!';
}

// The entire test expression (! $finished)
// is true if $finished is false
if (! $finished) {
    print 'Not done yet!';
}
```

Combining multiple expressions with && and ||

- The logical AND operator (&&)
- The logical OR operator (||)

```
if (($age >= 13) && ($age < 65)) {
   print "You are too old for a kid's discount and too young for the senior's
discount.";
}

if (($meal == 'breakfast') || ($dessert == 'souffle')) {
   print "Time to eat some eggs.";
}
```

Repetition (Loops)

For-loop, while-loop, do-while

- for()
- while()
- do-while()

```
print '<select name="people">';
for ($i = 1; $i <= 10; $i++) {
    print "<option>$i</option>\n";
}
print '</select>';
```

```
$i = 1;
print '<select name="people">';
while ($i <= 10) {
    print "<option>$i</option>\n";
    $i++;
}
print '</select>';
```

Example 3-17 prints:

```
<select name="people"><option>1</option>
<option>2</option>
<option>3</option>
<option>4</option>
<option>5</option>
<option>6</option>
<option>7</option>
<option>8</option>
<option>9</option>
<option>10</option>
</select>
```

For-loop

- You can combine multiple expressions in the initialization expression and the iteration expression of a for() loop by separating each of the individual expressions with a comma.

```
print '<select name="doughnuts">';
for ($min = 1, $max = 10; $min < 50; $min += 10, $max += 10) {
    print "<option>$min - $max</option>\n";
}
print '</select>';
```

Each time through the loop, `$min` and `$max` are each incremented by 10. Example 3-19 prints:

```
<select name="doughnuts"><option>1 - 10</option>
<option>11 - 20</option>
<option>21 - 30</option>
<option>31 - 40</option>
<option>41 - 50</option>
</select>
```

Exercises

1. Without using a PHP program to evaluate them, determine whether each of these expressions is `true` or `false`:
   a. `100.00 - 100`
   b. `"zero"`
   c. `"false"`
   d. `0 + "true"`
   e. `0.000`
   f. `"0.0"`
   g. `strcmp("false","False")`
   h. `0 <=> "0"`
2. Without running it through the PHP engine, figure out what this program prints:

```php
$age = 12;
$shoe_size = 13;
if ($age > $shoe_size) {
    print "Message 1.";
} elseif (($shoe_size++) && ($age > 20)) {
    print "Message 2.";
} else {
    print "Message 3.";
}
print "Age: $age. Shoe Size: $shoe_size";
```

3. Use `while()` to print a table of Fahrenheit and Celsius temperature equivalents from –50 degrees F to 50 degrees F in 5-degree increments. On the Fahrenheit temperature scale, water freezes at 32 degrees and boils at 212 degrees. On the Celsius scale, water freezes at 0 degrees and boils at 100 degrees. So, to convert from Fahrenheit to Celsius, you subtract 32 from the temperature, multiply by 5, and divide by 9. To convert from Celsius to Fahrenheit, you multiply by 9, divide by 5, and then add 32.
4. Modify your answer to Exercise 3 to use `for()` instead of `while()`.