CSY2028
Web Programming

Topic 3 - Control Structures and Loops

- Variables and Types in PHP
- If Statements
- Loops
  - for Loop
  - while Loop

Syntax

- In PHP, every statement must be inside PHP tags <?php and ?>
- All expressions must either end in a brace (}) or a semicolon (;)
- Variables must start with a dollar symbol ($)

Variables

- A variable is an identifier for a piece of information
- You can save something under a label that you choose and retrieve it later in the program

```
$num = 123;

echo $num;
```

Comments

- Comments can be included in your code to explain what it is doing
- Comments are not processed by the computer
- The contents of comments don't have a strict structure and can contain any text
- Comments do not have an impact on the program. It will run exactly the same whether they are included or not

Comments

- Comments serve two main purposes:
  - To allow you to document the code
  - To easily “turn off” lines of code without removing them entirely. This is known as “commenting out”

Comments

```
// this is a single line comment

/*
        This is a multi-line comment
*/
```

```
// echo 'test';
```

This code will not be run even though it is valid

Types

- PHP is a loosely typed language
- This means you do not have to declare variables with a type
- In Java, for instance, you have to declare a variable before you can use it: You must declare what type that variable will store, and that variable can then only store a value of that type

```
String myVariable;
myVariable = "hello";
```

Java Code, not PHP

Types

- In PHP, you can create a variable without giving it a type
- That variable can later store other types:

```
//Create a variable to store an integer
$myIntVariable = 123;

//Create a variable to store a string,
$myStringVariable = 'A string';

//However, the type is not fixed: You can store any type in any variable:
$myIntVariable = 'A string';

$myStringVariable = 456;
```

PHP Inbuilt Functions

- The rand() function can be used to generate a random number:
- Each time you run the script it will generate a random number

```php
<?php
echo rand();
?>
```

```
Output:
185902316
```

PHP Inbuilt Functions

- The pi() function calculates mathematical PI to 14 decimal places

```php
<?php
echo pi();
?>
```

```
Output:
3.1415926535898
```

PHP Inbuilt Functions

- Functions can take arguments
- The rand() function will generate a random number from 0 – 2147483647

```php
<?php
echo rand();
?>
```

```
Output:
727321813
```

- You can give functions arguments these are values which change the way the function works

PHP Inbuilt Functions

- Each function has arguments that are specific to that function
- The rand() function optionally takes two arguments:
  - Minimum value
  - Max value
- Both arguments are integers

Arguments

- You can supply a min/max value to the rand function by providing two numbers inside the brackets:

```php
<?php
echo rand(1, 10);
?>
```

```
Output:
6
```

- Each argument is separated by a comma
- Different functions take different numbers of arguments
- Those arguments are specific to that function

Rand values

- As well as printing the result of the rand() function, you can also store the result in a variable

```php
<?php
$num = rand(1, 10);
echo $num;
?>
```

```
Output:
3
```

If Statement

- You can use an if statement to inspect the value of a variable and run some code when that condition is met
- The == operator inspects two values to see if they are equal

```php
$num = 123;


if ($num == 124) {
    echo 'num is equal to 124';
}

if ($num == 123) {
    echo 'num is equal to 123';
}
```

```
Output:
num is equal to 123
```

Else Statement

- You can combine and if statement and an else statement to run one piece of code if the condition is met, and another if it is not:

```php
$num = 123;

if ($num == 124) {
        echo 'num is equal to 124';
}
else {
        echo 'num is not equal to 124';
}
```

```
Output:
num is not equal to 124
```

Checking Multiple Conditions

- You can use boolean operations to compare multiple values at once
- AND

```
$num = 123;
$num2 = 124;

if ($num == 124 && $num2 == 124) {
        echo 'num is equal to 124';
}
else {
        echo 'num is not equal to 124';
}
```

Will only get to here
If num1 is equal to 124
AND
num2 is equal to 124

Checking Multiple Conditions

- You can use boolean operations to compare multiple values at once
- OR

```
$num = 123;
$num2 = 124;

if ($num == 124 || $num2 == 124) {
        echo 'num is equal to 124';
}
else {
        echo 'num is not equal to 124';
}
```

Will only get to here
If num1 is equal to 124
OR
num2 is equal to 124

Exercise 1

1. Simulate a dice roll game where 6 is a winning number.
2. A number should be chosen at random between 1 and 6.
3. Each time you refresh the page it should print of the following statements depending on the random number that was chosen:
    - “You rolled a 6! You win!”
    - “You didn’t roll a 6, you lose”
4. **Optional** - Change the second message to “You rolled a [number], you lose”
5. **Optional** - Add a link to the page title “Roll again” which reloads the page.
6. Hint: You do not need to use javascript for this
7. **Optional** - Add a second dice. To win you must now roll a 6 on each dice. Add the message “You rolled a [dice1] and a [dice2], you [win/lose]”.
8. **Optional** - Now change the exercise to show the “You win” message if a 6 is rolled on either of the two dice

Exercise 1 - Solutions

```php
$num = rand(1,6);

if ($num == 6) {
     echo 'You rolled a 6, you win!';
}
else {
     echo 'You rolled a ';
     echo $num;
     echo ', you lose';
}

echo '<a href="dice.php">Roll again</a>';
```

```php
$num = rand(1,6);
$num2 = rand(1,6);

echo 'You rolled a ';
echo $num;
echo ' and a ';
echo $num2;
echo '. ';
if ($num == 6 && $num2 == 6) {
     echo 'You win!';
}
else {
   echo 'You lose';
}

echo '<a href="dice.php">Roll again</a>';
```

Concatenation

- String concatenation is a fancy term for joining strings together
- In PHP you can join strings with the . Operator
- This can be useful to reduce the number of echo commands required

```php
$num = rand(1,6);
$num2 = rand(1,6);

echo 'You rolled a ';
echo $num;
echo ' and a ';
echo $num2;
echo '. ';
if ($num == 6 && $num2 == 6) {
     echo 'You win!';
}
else {
   echo 'You lose';
}

echo '<a href="dice.php">Roll again</a>';
```

```php
$num = rand(1,6);
$num2 = rand(1,6);

echo 'You rolled a ' . $num . ' and a ' . $num2 . '. ';

if ($num == 6 && $num2 == 6) {
     echo 'You win!';
}
else {
   echo 'You lose';
}

echo '<a href="dice.php">Roll again</a>';
```

Types

- Types are automatically converted from one to the other
- In a strongly typed language such as Java you must convert between types to do a comparison. For example this will error:

```
String str = "1234";
int num = 1234;

if (num == str) {
        System.out.println("str and num are equal");
}
```

Java Code, not PHP

- And needs to be changed to or the program will not run

```
String str = "1234";
int num = 1234;

if (num == Integer.parseInt(str)) {
        System.out.println("str and num are equal");
}
```

Java Code, not PHP

Types

- In PHP you do not need to convert between types
- When you do any comparison PHP will do the conversion for you:

```
$str = '1234';
$num = 1234;

if ($num == $str) {
        echo 'str and num are equal';
}
```

Comparison Operators in PHP

- There are several comparison operators in PHP
- $a == $b equality, $a and $b are equal
- $a != $b not equals, $a and $b are not equal
- $a > $b true if $a is greater than $b
- $a < $b true if $a is less than $b
- For a complete list see

https://www.php.net/manual/en/language.operators.comparison.php

Comparing Types

- The == operator will try to do a comparison ignoring variable types
- It will try to convert numbers to strings before doing the comparison so this works:

```
$str = '1234';
$num = 1234;

if ($num == $str) {
        echo 'str and num are equal';
}
```

Comparing Types

- This sometimes causes some odd results
- This evaluates to true:

```
if (0 == '0') {
}
```

- As does this:

```
if (0 == 'abc') {
}
```

- But this evaluates to false

```
if ('0' == 'abc') {
}
```

Confused ?

- Generally the == operator will work but often has side effects due to the conversion. There are few other problems like this and it can cause difficult to track down bugs

```
0 == '0' //true
0 == 'string' //true
'0' == 'string' //false ??
```

===

- The === operator does an equality and type check. It evaluates to true if
  - The values are equal (e.g. '1' and 1)
  - AND they share the same type

```php
$str = '1234';
$num = 1234;

if ($num == $str) {
     //evaluates to true and this code will get run
}
```

```php
$str = '1234';
$num = 1234;

if ($num === $str) {
     //evaluates to false and this code will not get run
}
```

Type Comparison

- You should use === wherever possible. This is where:
  - You know the types you are dealing with
  - You don't need to rely on the conversion
- This will help reduce bugs and oddities.
- === is also slightly faster You can also use !== in place of !=

PHP Types

- The types in PHP are:
  - Double
  - Boolean
  - String
  - Integer
  - Array
  - Object
  - Resource
  - Null

Boolean Operations on Types

- In PHP boolean operations can be done on any type for example:

```php
$string = 'my string';

if ($string) {
}

$num = 123;

if ($int) {
}

$boolean = true;

if ($boolean) {
}
```

Boolean Operations on Types

- PHP will try to convert any type to a boolean. The following mean false:
  - An empty string e.g. $string = “”;
  - The number zero e.g. $num = 0;
  - Boolean false

- Anything else mean true:
  - A string with more than one character
  - Any number that is not zero

if / else

- You can combine an if statement with an else statement

```
if ($condition) {
        //Any code between these braces will be run
        //When $condition evaluates to true
        echo 'Condition was met'
}
else {
        //Any code between these braces will not be run
        //When $condition evaluates to true
        echo 'Condition was not met'
}
```

if / else if / else

- An if statement can only have one else
- However, you can add an else if statement
- An else if statement will only be run if the original if condition is not met

```
if ($condition1) {
        //Any code between these braces will be run
        //When $condition evaluates to true
        echo 'Condition 1 was met';
}
else if ($condition2) {
        //This code will get run only if condition 2 evaluates to true
        echo 'Condition 2 evaluates to true and condition 1 evaluates to false';
}
else {
        //Any code between these braces will only be run
        //when both $condition1 and $condition2
        //Evaluate to false
        echo 'Condition 1 evaluates to false and condition 2 evaulates to false';
}
```

If / else / else if

- You can add as many else if statements as you like:

```
if ($condition1) {
        //Any code between these braces will be run
        //When $condition evaluates to true
        echo 'Condition 1 was met';
}
else if ($condition2) {
        //This code will get run only if
        echo 'Condition 2 evaluates to true and condition 1 evaluates to false';
}
else if ($condition3) {
        echo 'Condition 3 evaluates to true';
}
else {
        //Any code between these braces will only be run
        //when both $condition1 and $condition2
        //Evaluate to false
        echo 'Condition 1 evaluates to false and condition 2 evaluates to false';
}
```

If / else / else if

- Only one branch of an if-elseif-else statement will ever be run

```
if ($condition1) {
        //Any code between these braces will be run
        //When $condition evaluates to true
        echo 'Condition 1 was met';
}
else if ($condition2) {
        //This code will get run only if
        echo 'Condition 2 evaluates to true and condition 1 evaluates to false';
}
else if ($condition3) {
        echo 'Condition 3 evaluates to true';
}
else {
        //Any code between these braces will only be run
        //when both $condition1 and $condition2
        //Evaluate to false
        echo 'Condition 1 evaluates to false and condition 2 evaluates to false';
}
```

If / else / else if

- If you remove the else from else if, it will be evaluated as two different statements

```
if ($condition1) {
        //Any code between these braces will be run
        //When $condition evaluates to true
        echo 'Condition 1 was met';
}

if ($condition2) {
        //This code will get run only if
        echo 'Condition 2 evaluates to true, condition 1 is either true or false';
}
```

Loops

- There are 2 main types of loop in PHP
  - For
  - While

- For loops for a predefined number of iterations. The number of times that the loop will occur is known upfront
- While loops while a condition is being met

Loops

- Loops can be used to run the same code a number of times.
- A for loop is used when you know the number of iterations (a posh word for the number of times the loop will run!)
- This code will print “Text” ten times
- Any code between the opening and closing brace will be run

```
for ($i = 0; $i < 10; $i++) {
        echo 'Text';
}
```

For Loop

- It's possible to make use of the loop counter inside the loop
- The variable declared in the first part of the for statement (in this example called i) will store the number of the current iteration
- This variable can be used like any variable, calculations, printing it

```
for ($i = 0; $i < 10; $i++) {
        echo $i;
}
```

```
Output:
0123456789
```

Printing HTML

- Because PHP outputs HTML, the output of the last slide was

```
0123456789
```

- If you want to display the count on it's own line you can use

```php
for ($i = 0; $i < 10; $i++) {
        echo '<p>' . $i . '</p>';
}
```

```
Output in browser:
0
1
2
3
4
5
6
7
8
9
```

For Loops

- A for loop has three parts.
- The first initializes the counter with a starting value

```
for ($i = 0; $i < 10; $i++) {
        echo '<p>' . $i . '</p>';
}
```

- Changing this will alter the value at which the counter starts:

```
for ($i = 3; $i < 10; $i++) {
        echo '<p>' . $i . '</p>';
}
```

```
Output in browser:
3
4
5
6
7
8
9
```

For Loops

- The second is the condition
- The condition will be evaluated on each iteration and while it evaluates to true, the loop will continue
- This can be read as “while the counter is less than ten”

```
for ($i = 0; $i < 10; $i++) {
      echo '<p>' . $i . '</p>';
}
```

- Changing this will adjust the stop-point of the loop:

```
for ($i = 0; $i < 4; $i++) {
      echo '<p>' . $i . '</p>';
}
```

```
Output in browser:
0
1
2
3
```

For Loops

- The final part is the modifier, this will be executed at the end of each iteration
- i++ means increment by one but this can be any mathematical expression.

```
for ($i = 0; $i < 10; $i++) {
        echo '<p>' . $i . '</p>';
}
```

- Changing this value can alter the amount that the counter is incremented on each iteration:

```
for ($i = 0; $i < 10; $i=$i+2) {
        echo '<p>' . $i . '</p>';
}
```

```
Output in browser:
0
2
4
6
8
```

Combining Expressions

- You can combine any expressions by nesting them inside the relevant braces

```
for ($i = 0; $i < 10; $i++) {
        if ($i === 2) {
                echo '<p>The counter is 2</p>';
        }
        else if ($i !== 2) {
                echo '<p>The counter is not 2</p>';
        }
}
```

```
Output in browser:
The counter is not 2
The counter is not 2
The counter is 2
The counter is not 2
The counter is not 2
The counter is not 2
The counter is not 2
The counter is not 2
The counter is not 2
The counter is not 2
The counter is not 2
```

While Loops

- While loops will keep looping while a predefined condition is true
- You must affect the condition inside the loop to halt it!

```php
$loop = true;
$counter = 0;

while ($loop === true) {

        $counter++;

        echo '<p>' . $counter . '</p>';

        if ($counter === 5) {
                $loop = false;
        }
}
```

```
Output in browser:
1
2
3
4
5
```

What's wrong with this code?

```
$loop = true;
$counter = 0;

while ($loop === true) {

        $counter++;

        echo '<p>' . $counter . '</p>';
}
```

What's wrong with this code?

- You must always have a way of exiting the loop or you will create an infinite loop

```
$loop = true;
$counter = 0;

while ($loop === true) {

        $counter++;

        echo '<p>' . $counter . '</p>';
}
```

Leaving a loop early

- You can leave a loop early by using the 'break' command.
- This breaks out of the loop and exits it at that point

```php
$loop = true;
$counter = 0;

while ($loop === true) {

        $counter++;

        echo '<p>' . $counter . '</p>';

        if ($counter === 5) {
                break;
        }
}
```

```
Output in browser:
1
2
3
4
5
```

Break

- Break works for both while and for loops

```
for ($i = 0; $i < 10; $i++) {
        echo '<p>' . $i . '</p>';

         if ($i === 5) {
                 break;
         }
}
```

```
Output in browser:
0
1
2
3
4
```

Ignoring an Iteration

- Sometimes you may want to conditionally skip the rest of the loop.
- This is possible by surrounding all the code with an if statement:

```php
for ($i = 0; $i < 10; $i++) {

        if (!$i !== 3) {
          echo '<p>' . $i . '</p>';
        }

}
```

```
Output in browser:
0
1
2
4
5
6
7
8
9
```

Ignoring an Iteration

- You can also do this with the continue statement
- This is useful as it avoids extra nesting and can make the code clearer

```php
for ($i = 0; $i < 10; $i++) {
        if ($i === 3) continue;
        echo '<p>' . $i . '</p>';
}
```

continue exits the current iteration and moves immediately on to the next.
Any code after the continue statement will be ignored

```
Output in browser:
0
1
2
4
5
6
7
8
9
```

Exercise 2

1. Using a loop, print out a list of all the numbers from 1-10 inside a an unordered list using <ul> and <li> tags
2. Change the exercise to print out the textual representation “One”, “Two”, “Three”, “Four”, etc
3. Write a program that uses a loop to display all the odd numbers from 21 to 99
   - **Grade A: Cam you do this without an if statement?**
4. Write a program that prints the nine times table up to 12 x 9 (9, 18, 27, etc)
   - **Grade A: Can you do this without an if statement and without using the multiplication operator?**
5. For this exercise you may (and probably should!) use the multiplication operator. Print the nine times table from 9 – 900 in the format
   - 1 x 9 = 9
   - 2 x 9 = 18
   - 3 x 9 = 27

Optional Exercises

These are optional and will require some research. You will need to look things up that are not covered in the lecture notes!

6. Using DirectoryIterator (http://php.net/manual/en/class.directoryiterator.php) and a loop, list all the files in the public_html directory inside a
7. Add a link to each file so you can click on it to view it
8. Filter out ‘dot’ files (e.g. `.` and `..` which represent the current directory and directory up
9. Only show files with a PHP extension. Hint: There are multiple ways of doing this but the best way is making use of http://php.net/manual/en/class.splfileinfo.php

Optional Exercises

These are optional and will require some research. You will need to look things up that are not covered in the lecture notes!

10. Using the PHP DateTime object (http://php.net/manual/en/class.datetime.php) list out every day from now until New Year's day in the format:
    - Thursday 20th October 2016
    - Friday 21st October 2016
    - Etc
11. Hint: You can use a while loop and you should use the DataTime's modify method
12. Without using a loop, create a Christmas Countdown clock that displays the number of Weeks/Days/Hours/Minutes until Christmas day?

Solutions

1)

```php
echo '<ul>';

for ($i = 1; $i <= 10; $i++) {
        echo '<li>' . $i . '</li>';
}
echo '</ul>';
```

2)

```php
echo '<ul>';
for ($i = 1; $i <= 10; $i++) {
     echo '<li>';
     if ($i == 1) {
          echo 'One';
     }
     else if ($i == 2) {
          echo 'Two';
     }
     else if ($i == 3) {
          echo 'Three';
     }
     else if ($i == 4) {
          echo 'Four';
     }
     else if ($i == 5) {
          echo 'Five';
     }
     else if ($i == 6) {
          echo 'Six';
     }
     //…...

     echo '</li>';
}
echo '</ul>';
```

Solutions

3)

```
echo '<ul>';

for ($i = 21; $i <= 99; $i += 2) {
        echo '<li>' . $i . '</li>';
}
echo '</ul>';
```

5)

```
echo '<ul>';

for ($i = 1; $i <= 100; $i++) {
        echo '<li>' . $i . ' x 9 = ' . ($i*9). </li>';
}
echo '</ul>';
```

4)

```
echo '<ul>';

for ($i = 9; $i <= 108; $i += 9) {
        echo '<li>' . $i . '</li>';
}
echo '</ul>';
```