WEB 306 - PHP: Databases and Framework

Day 2 – Flow Control

GitHub Resources

1. Git Website
2. Git for Windows/Git Bash
3. GitHub Website
4. Tim's GitHub Account
5. The Seneca Web Programming GitHub Organization
6. GitHub Desktop
7. GitHub Education

1. Intro to GitHub

1.1 About Git

First of all, Git and GitHub are not the same thing but this is a common misconception. Git is a free and open source distributed version control system. Developers use it to manage revisions, compare and review changes and track their code history. Find out more about Git at git-scm.com. Traditionally Git is managed through the command line but many graphic user interfaces have been made to use Git as well.

1.2 About GitHub

GitHub is the most popular hosting service for Git projects and also provides a GUI for using Git. It also acts as a social network and discussion forum for developers where they can find other developers, view and download their code, upload and share their own code, ask questions and provide feedback on projects. To sign up for an account, go to github.com.

1.3 GitHub Desktop App

The easiest way to upload projects to GitHub is to use their Desktop App. To download the app go to desktop.github.com. Log into the app using the account you just created.

GitHub projects are stored in repositories. From the GitHub app you can create a new repository, add an existing repository from a folder on your computer or clone a repository from the GitHub site to your computer. Whenever you make changes to a file in a GitHub repository they are tracked and displayed in the GitHub app. When

GH App Shortcuts

1. **New Repository: ctrl+n**
2. **Add Local Repository: ctrl+o**
3. **Clone Repository: ctrl+shift+o**
4. **Push: ctrl+p**
5. **Pull: ctrl+shift+p**

VS Code Git Shortcuts

1. **Open Git Tab: ctrl+shift+G**

Atom Git Shortcuts

1. **Open Git Tab: ctrl+shift+9**
2. **Open GitHub Tab: ctrl+shift+8**

you are ready to upload your changes to GitHub you make a commit. When commiting changes to GitHub you must include a short summary of the changes you made and can optionally include a longer description of those changes if they are extensive.

1.4 Git/GitHub and VS Code/Atom

Partially because Atom is made by GitHub and VS Code is made by Microsoft, who own GitHub, both editors come with a lot of Git and GitHub integration features right out of the box. You can create and manage repositories directly from VS Code or Atom, but it is not quite as nice as the GitHub app. What's more useful than that is that as you make changes to a file in a GitHub repository in VS Code or Atom it will highlight them in special colours, allowing you to track your changes in real time.

1.5 Branches

When you first create a GitHub project it will be comprised of what is called a "master" branch. The master branch is where the primary version of a project is stored. You can create sub-branches of a project so that developers can work on specific features of a project without altering the code of, or potentially breaking the main branch. You can also create sub-branches of sub-branches. Once the developers working on a given branch have completed development on that branch they will merge their branch back into the master branch.

2. Flow Control

2.1 Control Flow

The ***control flow*** of a program is the order and circumstances in which the code is executed. The control flow of a program can actually be represented using a flow chart.

2.2 Control Structures

Control structures are a set of commands which control and alter the flow of code depending on different conditions and circumstances. These include ***conditionals***, ***loops*** and ***function calls***.

PHP Resources

1. PHP Website
2. XAMPP
3. Documentation
4. W3Schools
5. CSS Tricks
6. Scotch.io
7. Stack Overflow

2.3 Comparison and Logical Operators

2.3.1 Comparison Operators

Comparison operators are used to compare values.

1. **`==` is used as an evaluator of equality**
2. **`===` is used as an evaluator of equality but ALSO checks the data type**
3. **`!=` not equal**
4. **`!==` not equal OR not the same data type**
5. **`<` less than**
6. **`>` greater than**
7. **`<=` less than or equal to**
8. **`>=` greater than or equal to**
9. **`<=>` known as "spaceship," returns 0 if equal, -1 if A is less than B, and 1 if A is greater than B**

2.3.2 Logical Operators

Logical operators are used to combine two conditions.

1. **`&&` and (returns true if both statements are true)**
2. **`||` or (returns true if one of the statements is true)**
3. **`!` not (returns false if the result is true)**

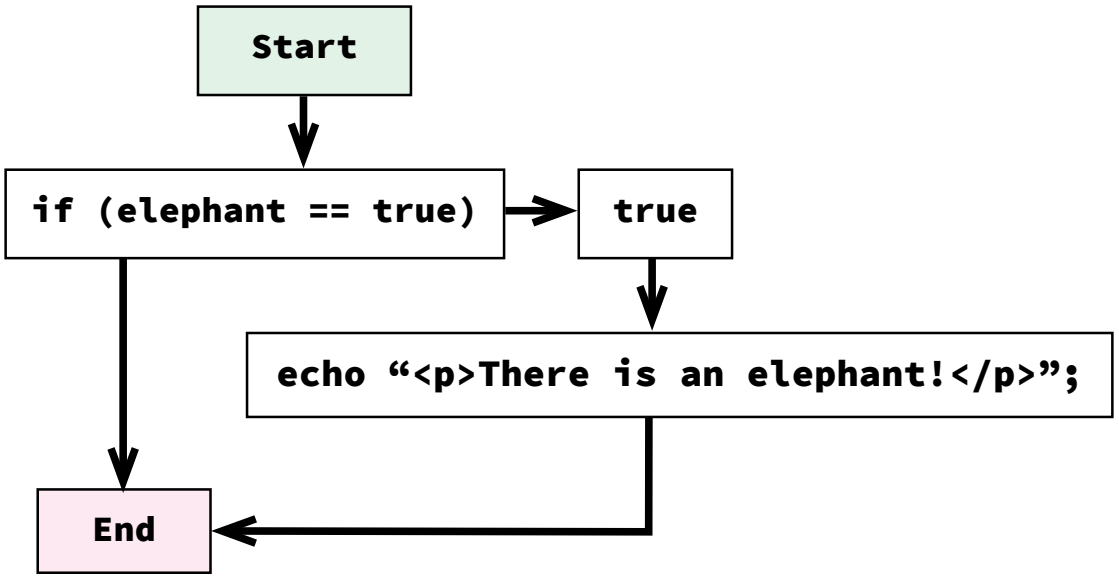
3. Conditionals

Conditionals are used to perform different actions depending on one or more given conditions.

3.1 If Statements

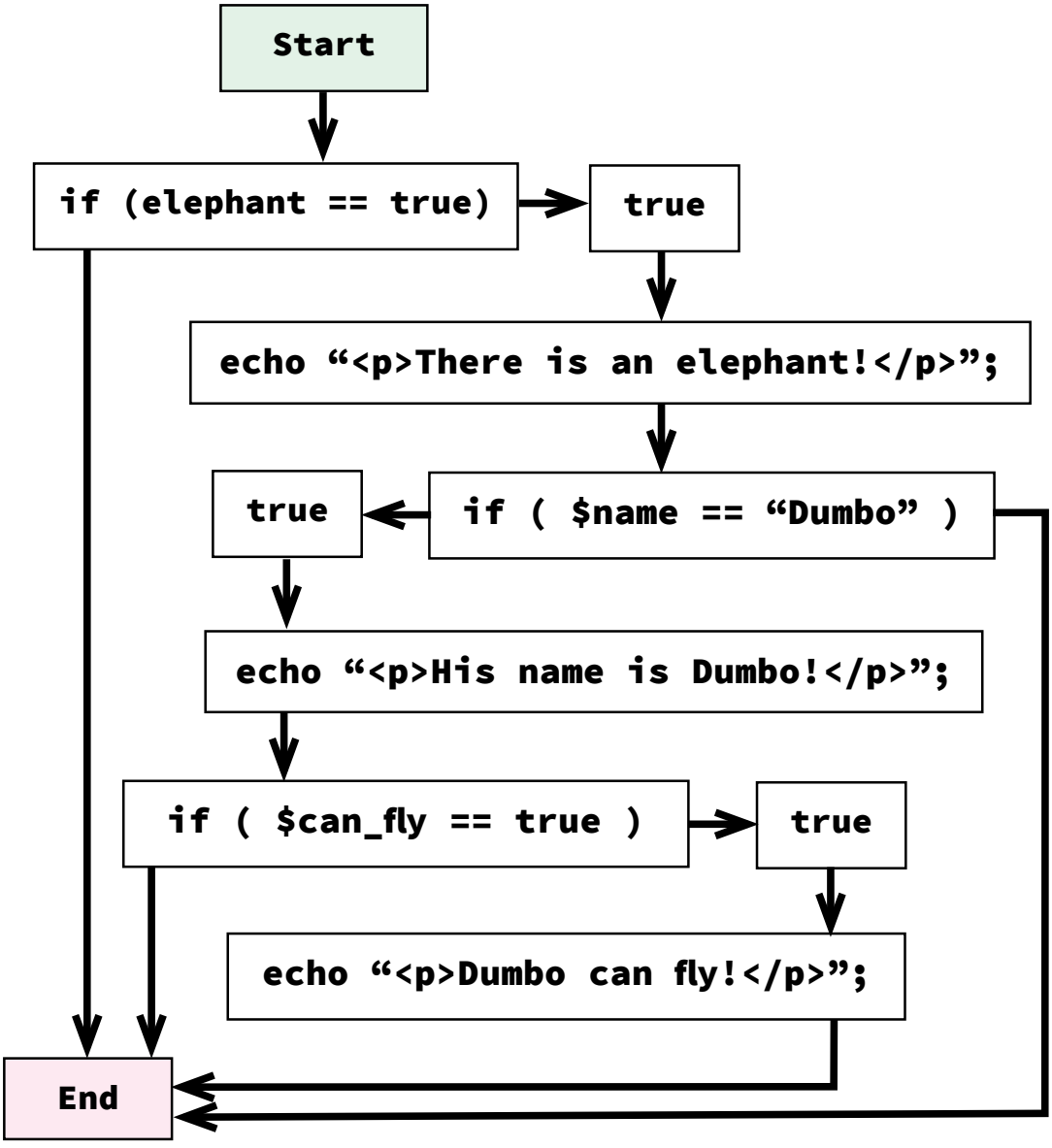
If statements are just a way of checking if a set of conditions is `true` or `false`. If the statement is `true`, the code which is included "inside" of the if statement ***will*** run. If the statement is `false` then the code which is "inside" of the if statement will ***not*** run.

```
if ( $elephant == true ) {
   echo "<p>There is an elephant!</p>";
}
```



If statements can be nested inside of each other as well.

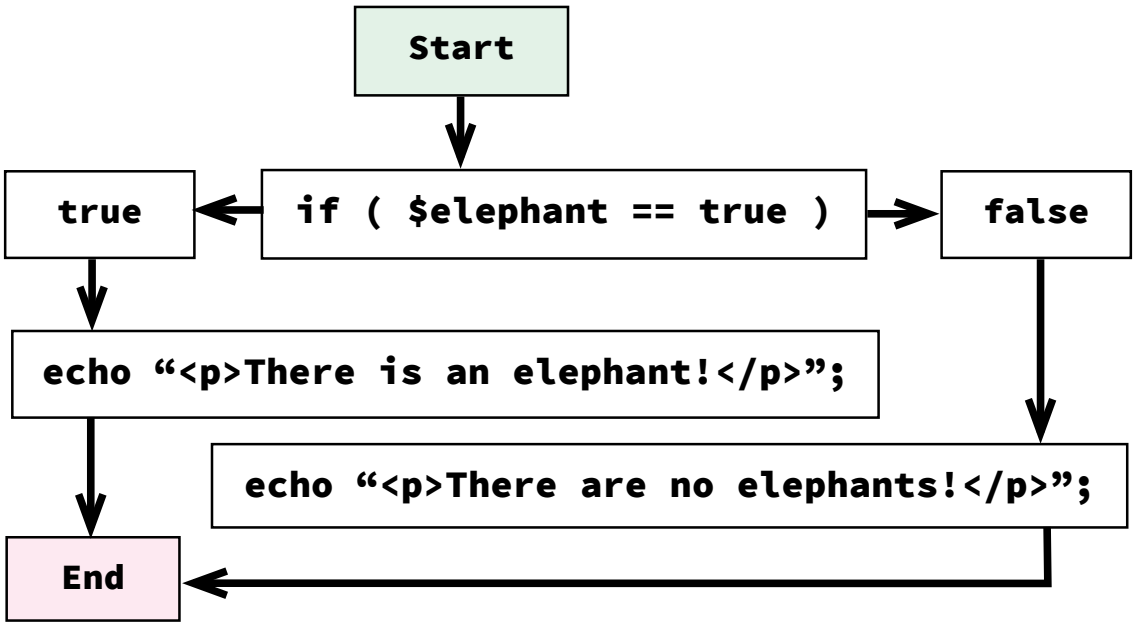
```
if ( $elephant == true ) {
   echo “<p>There is an elephant!</p>”;

   if ( $name == “Dumbo” ) {
        // This code will only run if the first statement and this
        //statement are true
        echo “<p>His name is Dumbo!</p>”;

        if ( $can_fly == true ) {
             echo “<p>Dumbo can fly!</p>”;
        }
   }
}
```



3.2 If Else Statements

If else statements are used to check if one set of conditions is `true` or `false` and then to set a default behaviour if the condition is false. If the statement is true, the code runs. If it is false then it runs the `else` code.

```
if ( $elephant == true ) {
   echo “<p>There is an elephant!</p>”;
} else {
   echo “<p>There are no elephants!</p>”;
}
```



3.3 If Else If Statements

If else if statements are used to check if two or more sets of conditions are `true` or `false`. If the first statement is `true`, the code runs. If it is `false` then it checks the next condition until all conditions have been checked.

```
if ( $elephant == true && $name == “Dumbo” && $can_fly == true ) {
   echo “<p>There is an elephant!</p>”;
} else if ( $elephant == true && $name == “Dumbo” ) {
   echo “<p>There is an elephant named Dumbo!</p>”;
} else if ( $elephant == true ) {
   echo “<p>There is an elephant!</p>”;
} else {
   echo “<p>There are no elephants!</p>”;
}
```

You can alternatively use a bracketless syntax for if statements like languages such as Ruby and Python. The bracketless syntax is supposed to be used for HTML templates. This syntax is very popular in Wordpress themes for HTML templating.

```
if ( $elephant == true && $name == “Dumbo” && $can_fly == true ):
   echo “<p>There is an elephant!</p>”;
elseif ( $elephant == true && $name == “Dumbo” ):
   echo “<p>There is an elephant named Dumbo!</p>”;
elseif ( $elephant == true ):
   echo “<p>There is an elephant!</p>”;
else:
   echo “<p>There are no elephants!</p>”;
endif;
```

3.4 Switch Statements

There is another type of conditional in PHP called a switch statement. Switch statements are used to check one variable which could have many potential values. You can also use bracketless switch statements.

```
switch ($name) {
   case ‘Dumbo’:
        echo ‘<p>Hi Dumbo!</p>’;
   break;
   case ‘Babar’:
        echo ‘<p>Hi Babar!</p>’;
   break;
   case ‘Horton’:
        echo ‘<p>Hi Horton!</p>’;
   break;
   default:
        echo ‘<p>Who is this?</p>’;
}
```

3.5 Try Catch Statements

There is yet another type of conditional in PHP called a try catch statement. Try catch statements are used to customize error messages so that they are phrased in a way that your user will understand the problem. Inside the try statement you would specify certain conditions under which error messages should be thrown. This is done using the keywords `throw new` and the `Exception()` function. The error messages are then “caught” and sent to the catch statement where you specify how you want the error messages to be shown. The preset function `getMessage()` is stored in our `$e` parameter and is used to display the message which is specified in the try statement.

```
try {
   if ($can_fly == false) {
        throw new Exception(‘Dumbo can NOT fly!’);
   }
} catch (Exception $e) {
   echo $e->getMessage();
}
```

4. Arrays

4.1 Basic Arrays

Arrays are just a way of storing a list of multiple pieces of data in one variable. They are written as comma separated lists. In PHP there are two ways to write an array. There is the shorform way which is common in many other languages. They are defined using square braces.

```
$fruits = ['apple', 'banana', 'pear', 'peach', 'grapes'];
```

Or you can use the `array()` function which was the original way of making arrays in PHP.

```
$fruits2 = array('apple', 'banana', 'pear', 'peach', 'grapes');
```

You can not echo out arrays. Instead, you must use `var_dump()` to test them.

```
var_dump($fruits);
```

To select a single array item, write the variable name and then, in square braces, put the index number. In PHP, lists start at 0.

```
echo $fruits[1];
```

4.2 Associative Arrays

An associative array is a way of creating an arry with labels for all of the list items. Associative arrays allow us to store much more complex data and to provide context for it. The "labels" in associative arrays are known as "keys". Associative arrays use `=>` arrows. Technically EVERY array in PHP is an associative one. If you don't specifiy a key then the keys are just index numbers.

```
$elephant = array(
   'name' => 'Dumbo',
   'age' => 3
);
```

To select items from associative arrays, write the variable name and then, in square braces, write the name of the key.

```
echo '<p>' . $elephant['name'] . ' is ' . $elephant['age'] . ' years
old!</p>';
```

It is also common to nest arrays to store data.

```
$elephants = array(
   array(
        'name' => 'Dumbo',
        'age' => 3
   ),
   array(
        'name' => 'Babar',
        'age' => 60
   ),
   array(
        'name' => 'Horton',
```

```
        'age' => 24
    )
);
```

In order to get data from a nested array just keep adding square braces to your variable.

```
echo '<p>' . $elephants[1]['name'] . '</p>';
```

4.3 Counting Array Items

The count() function is used to count array items.

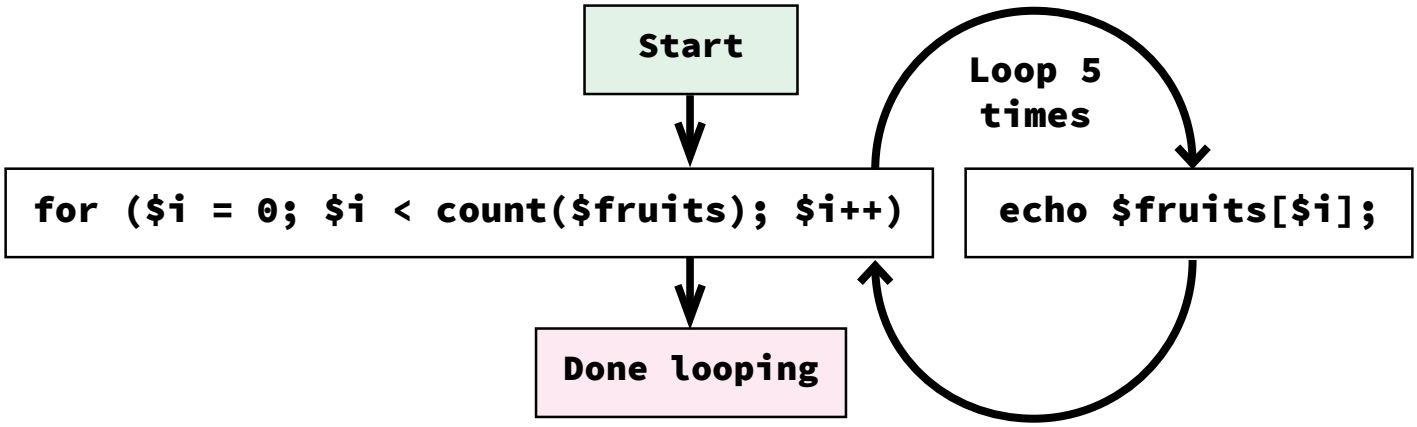
```
echo count($elephants);
```

5. Loops

5.1 For Loops

For loops repeat the same action for x times. To do this we use a variable which only exists inside of the loop which we will call `$iteration`. First we set the `$iteration` to 0. Then we tell the loop to run while `$iteration` is less than the the number of items in the `$fruits` array. Finally we increase the value of `$iteration` by 1 using `$iteration++`.

```
for ($iteration = 0; $iteration < count($fruits); $iteration++) {
  echo '<p>' . $fruits[$iteration] . '</p>';
}
```



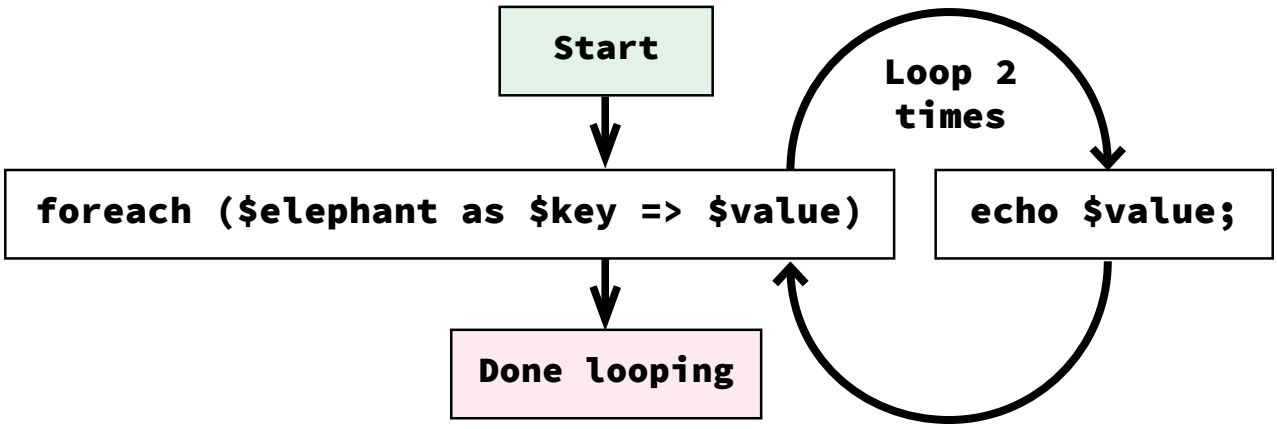
Loops can also be written with the bracketless syntax.

```
for ($iteration = 0; $iteration < count($fruits); $iteration++):
  echo '<p>' . $fruits[$iteration] . '</p>';
endfor;
```

5.2 For Each Loops

For each loops repeat the same action for each item in a set of items. They are great for looping through associative arrays. You can also use bracketless for each loops.

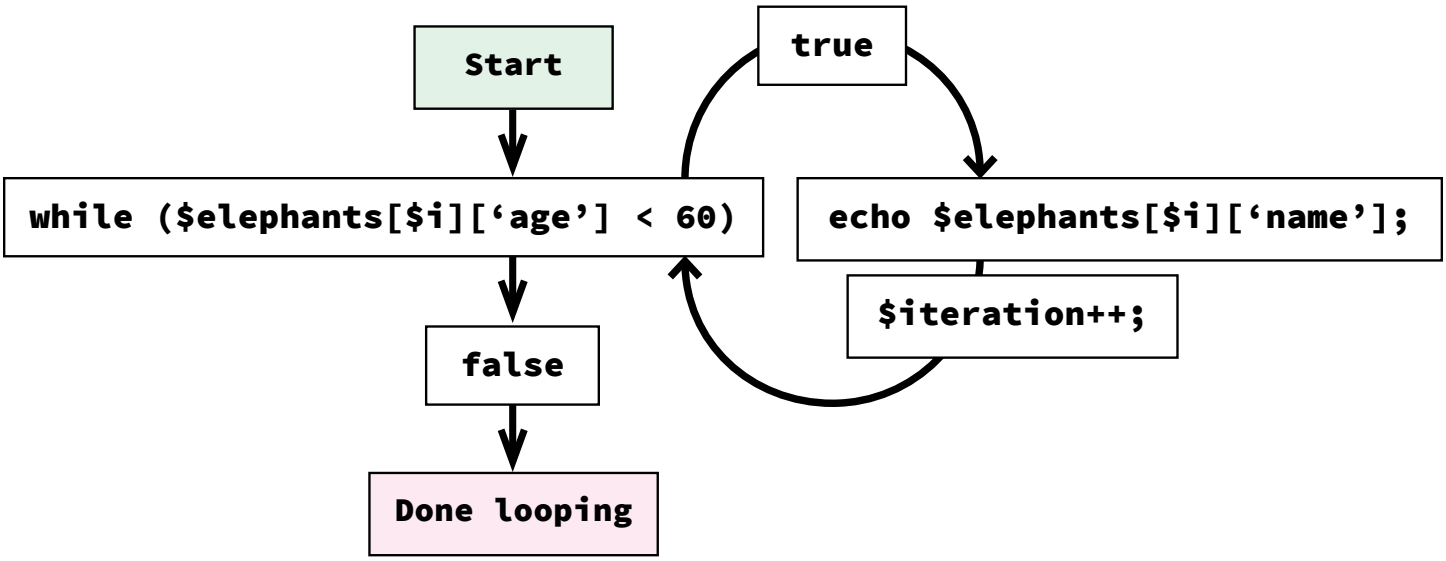
```php
foreach ($elephant as $key => $value) {
   echo '<p>' . $value . '</p>';
}
```



5.3 While Loops

While loops are like a combination of a loop and a conditional. A while loop will check for a condition and continue to loop through the code while the condition remains true.

```php
$iteration = 0;

while ($elephants[$iteration]['age'] < 60) {
   echo '<p>' . $elephants[$iteration]['name'] . ' is  young.</p>';
   $iteration++;
}
```



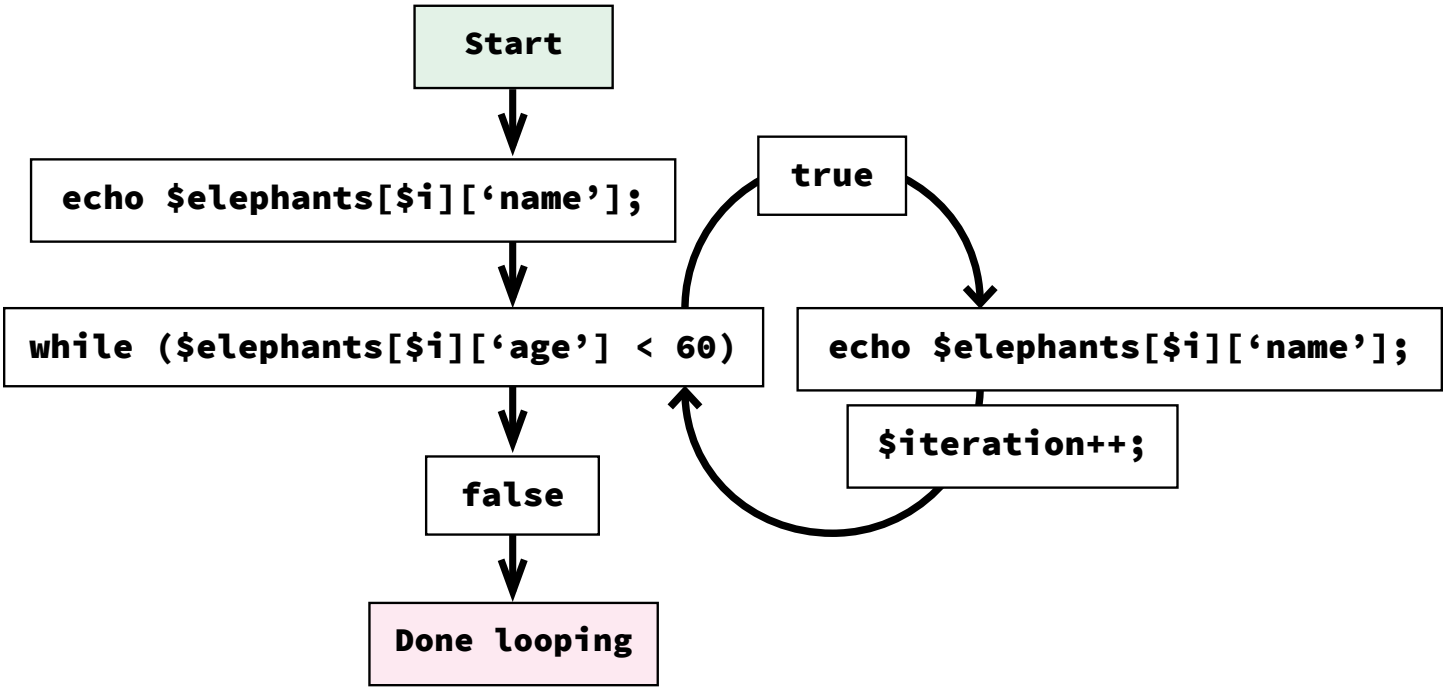
It is important to increment the condition of your while loop manually or it will just loop endlessly and could potentially crash your computer. You can also use bracketless while loops.

5.2 Do While Loops

Do while loops are very similar to while loops but they check the condition at the end of each iteration instead of the beginning. This means that the code will always run once, even if the condition is false.

```
$iteration = 0;

do {
   echo '<p>' . $elephants[$iteration]['name'] . ' is  young.</p>';
   $iteration++;
} while ($elephants[$iteration]['age'] < 60);
```

To keep things inconsistent, there is no brackless version of the do while loop.



6. Functions

6.1 Defining Functions

We can create custom functions which will perform actions which we have programmed in ourselves. Functions are kind of like mini programs. They allow us to save code in chunks which can be used and reused later. To define a function, use the `function` keyword followed by the name of the function and the parentheses.

```
function flying_elephant() {
   echo "<p>Dumbo is flying!<p>";
}
```

6.2 Parameters

Parameters allow us to change the way functions work by acting like variables which only exist inside of the function. The parameter acts as a placeholder for a real value which will be added when the function is called.

```
function flying_elephant($elephant_name) {
   echo “<p>{$elephant_name} is flying!<p>”;
}
```

6.2.1 Default Parameters

Parameters can have default values set as a fallback in case there is no value set when the function is called.

```
function flying_elephant($elephant_name = “Dumbo”) {
   echo “<p>{$elephant_name} is flying!<p>”;
}
```

6.3 The Return Statement

The `return` statement is used to return single values and calculations when a function is called. When functions return a single value they can be saved in variables.

```
function flying_elephant() {
   return “<p>Dumbo is flying!<p>”;
}
```

6.4 Calling Functions

Functions do not actually do anything until we “call” them. To call the function write the name of the function, followed by the parentheses to indicate that it is a function.

```
flying_elephant(“Babar”);
```

6.5 Premade Functions

6.5.1 The Date Function

There are many premade functions in PHP. The `date()` function gets the date.

```
echo date(‘Y’);
```

6.5.2 Value Checking Functions

`isset()` checks if a value is set or null. If a value is set it will return true.

```
var_dump(isset(‘This value is set!’)); // returns true
var_dump(isset(null)); // returns false
```

If we add an `!` in front of it then it will return true if the value is NOT set.

```
var_dump(!isset(null)); // returns true
var_dump(!isset(‘’)); // returns false
```

`empty()` checks if a value is empty but it will ***ALSO*** return true if a string is empty.

```
var_dump(empty(null)); // returns true
var_dump(empty(‘’)); // returns true
var_dump(empty(‘This value is full!’)); // returns false
```

If we add an `!` in front of it then it will return true if the value is ***NOT*** empty.

```
var_dump(!empty(‘This value is full!’)); // returns true
var_dump(!empty(‘’)); // returns false
```

6.5.3 String Manipulation Functions

Sometimes users enter data which is improperly formatted. We can use functions to format the data. The `strtolower()` function changes ALL characters to lowercase.

```
echo ‘<p>’ . strtolower(‘AN UPPERCASE STRING’) . ‘</p>’;
```

The `strtoupper()` function changes ALL characters to UPPERCASE.

```
echo ‘<p>’ . strtoupper(‘a lowercase string’) . ‘</p>’;
```

The `ucwords()` function capitalizes the first letter of every word in the string.

```
echo ‘<p>’ . ucwords(‘a lowercase string’) . ‘</p>’;
```

The `ucfirst()` function capitalizes the first letter of the string.

```
echo ‘<p>’ . ucfirst(‘a lowercase string’) . ‘</p>’;
```

6.6 Combining Functions

Functions can also be called inside of other functions. You can even call a function inside of itself.

6.6.1 Sanitizing User Data

Hackers can insert HTML, XML or Javascript code into a form and hijack it. To prevent this, it is important to sanitize form data before displaying or using it. Let’s make a function to sanitize any form data we use.

```
function sanitize_string($string, $special_characters) {
   $string_strip_tags = strip_tags($string);
   $string_remove_chars = preg_replace($special_characters, ‘’,
$string_strip_tags);
   $string_sanitized = htmlspecialchars($string_remove_chars, ENT_
QUOTES, ‘UTF-8’);
   return $string_sanitized;
}
```

First we will TRY to strip HTML tags from user input to avoid hacking using `strip_tags()`. This function tries its best but might miss some stuff. Then we will remove special characters and replace them with an empty string using the `preg_replace()` function. Then we will convert any remaining

HTML or special characters into plain text using the `htmlspecialchars()` function. Finally we will export the sanitized string using the `return` statement (see ***6.3 The Return Statement***).

6.6.2 Formatting User Data

Let's make a function to properly format a name. First we will use the `strtolower()` function to change ALL characters to lowercase. Then we will use the `ucwords()` function to capitalize the first letter of every word in the name. Finally we will export the formatted name using the `return` statement (see ***6.3 The Return Statement***).

```
function format_name($name) {
  $name_lowercase = strtolower($name);
  $name_capitalized = ucwords($name_lowercase);
  return $string_sanitized;
}
```

6.6.3 Calling Multiple Functions

We can call functions as parameters of other functions. We will call `sanitize_string()` first and then call `format_name()` to sanitize the user's name and then to format the sanitized string.

```
$processed_name = format_name(sanitize_string('<p>dUmBo@#$%^&</p>', '/
([<>!@#$%^&])/'));

echo $processed_name;
```

We will use these custom functions in the next lesson.