CSIS 3280: Lecture 2

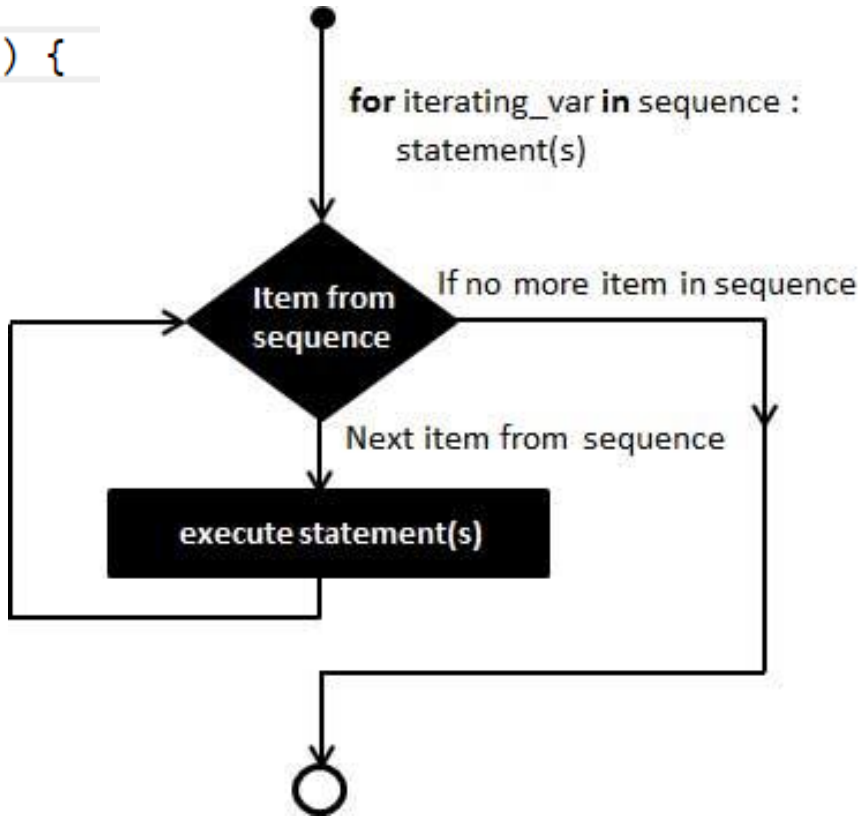
for loop

- The initialization, comparison and update in a single line of code

```php
<?php

    for ($v = 10; $v > 0; $v--) {
        echo $v  . "\n";
    }

?>
```



break and continue

- Use break; to '**break out**' of a block statement, e.g. loop
- Use continue; to **skip** the current iteration (cycle) of the loop **and continue the next iteration**.

```php
<?php

while(true) {

    $s = date("s");
    sleep(1);

    if ( $s %10 == 0 ) {
        break;
    } else{
        echo $s."-";
    }
}

?>
```

```php
<?php

while(true) {

    $s = date("s");
    sleep(1);

    if ( $s %10 == 0 ) {
        break;
    } elseif ( $s %5 == 0 ) {
        continue;
    } else{
        echo $s."-";
    }
}

?>
```

Demo code: for_break.php, for_continue.php

Foreach loop

- Provides an easy way to iterate over arrays
- *foreach* works only on arrays and objects

```
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement
```

Function

Introduction

- Functions allow us to break up computational tasks into smaller reusable pieces.
  - They allow us to give structure to our code.
  - They reduce repetition in our code
- We can combine functions with arrays and have highly re-usable code.
- Our consistency and reliability is increased if we follow the DRY and KISS principle.
  - DRY (Don’t repeat yourself): divide your code and logic into smaller and reusable unit. Less code is good! Saves time and effort, easy to maintain, reduce bugs
  - KISS (Keep it simple stupid): keep code simple and clear; making it easy to understand. Each method should only solve one small problem
- PHP has over 1,000 functions built into the standard library

Reuse: functions

- Great programmer are lazy. Lazy programmer think in terms of reusability!
  - Some functions have no parameter function();
  - Other functions have parameters function($passMeParameters);
  - We can combine function calls and nest them. For example: do(something($now));

Reuse: Structure

- We put functions in files and *require* them in our PHP programs.
- This promotes good programming structure.
- Eventually we will put classes/objects in our include files inside those objects will be functions.
- We will adhere to separation of concern principle:
  - Separate the utility, logic/calculation and html/display

Reuse: HTML templates

- We will be implementing code re-use in our lab.
- Cutting up our HTML files into a header, footer and body as well as various features or UI functions is useful.
  - We will avoid mixing up the HTML and php!
- We may also use different HTML frameworks in this class, we can use functions to keep our code clean.

Troubleshooting Functions

- Are they spelled correctly?
- Do they exist in your current runtime?
- Did you include the file?
- Do they require installation of some dependencies (pear, GNU readline support etc..)

functionConventions()

- Functions are **not case sensitive**,
  - Its best to use camelCase naming convention
  - its best to name them according to the object or include file.

```
htmlHeader($title); //html is the include file, header is the
function hence the name htmlHeader();
```

- Use descriptive names in order to encourage code readability
- You cannot have two functions with the same name, you can have only letters, numbers and underscores.
  - You cannot begin a function name with a digit.
- Always declare them at the top of your file.

Parameters

```
htmlHeader($title); //html is the include file,
header is the function hence the name htmlHeader();
```

- We use these to pass data to the function, this can either be single values or arrays.
- We can pass any kind of array we wish – This is awesome.
- We can **pass by value** or **pass by reference**
- Pass by reference parameters are prepended with an ampersand (&)

Scope

- Misunderstanding about scope will lead to error or bug
- Variables declared in a function are called local variables, inside the function they are in the function scope.
- Variables outside functions are in the global scope.
- There are super global variables and these are visible regardless of local, global or functional scope.
- You can delete variables using *unset()* this is important for scalability. **There is no garbage collector pick up your own trash for large operations!**

Calling and Creating a Function

- We have used print() and printf()
- A function definition looks like below

```
function functionName(parameters)
{
    function body
}
```

Function declaration example

```
//Simple function
function getInput() {
    $input = stream_get_line(STDIN, 1024,PHP_EOL);
    return $input;
}
```

Function call example

```
$input1 = getInput();
```

Passing arguments by Value

- You can pass data into a function by value

```
function calcSalesTax($price, $tax){
    echo "Total cost: $" . $price * (1 + $tax);
}
```

  - Argument $price and $tax were passed by value.
  - $price and $tax may be called something different outside of this function
  - The scope of $price and $tax is within calcSalesTax() function only

```
$produtPrice = 100;
$tax_percentage = 0.12;

calcSalesTax($produtPrice, $tax_percentage);
```

Passing by reference

- You can pass a variable by reference to a function
  - The variable's value can be changed within the function block

```php
$produtPrice = 100;
$tax_percentage = 0.12;
$cost = "";

function calcSalesTaxReference(&$cost, $price, $tax){
    $cost = "\nTotal cost: $" . $price * (1 + $tax);
    // do we know $productPrice here?
}

calcSalesTaxReference($cost,$produtPrice,$tax_percentage);
echo $cost;
```

Demo code: functions.php, passbyreference.php

Default arguments

- You can specify that an input argument has a default value in the function declaration.
  - Note the default value of tax is changed to 0.1

```php
function calcSalesTaxDefault($price, $tax=0.1){
    echo "\nTotal cost: $" . $price * (1 + $tax);
}

calcSalesTaxDefault($produtPrice);
```

  - If we want to use the default value, omit the $tax argument in the function call

Using Type Hinting

- You can use Type Hinting – This allows us to force either arrays or objects, scalar data type hinting is not supported in PHP 5.
- Type hinting for scalar data types IS supported in PHP 7.

```
function processPayPalPayment(Customer $customer) {
    // Process the customer's payment
}
```

Returning values

- Your functions most of the time will have return values particularly if they are data processing.
- These values are returned using the **`return`** keyword.
- After the `return` keyword is reached nothing else is processed and the function returns.
- You may specify a return value when your function returns such as return 0;
- You can also return arrays, this is also very powerful.

The return statement

- Return statement returns values back to the caller
  - What good is calculating the total if you don't pass it back to the caller?
  - The value will sit there until the function is complete and then be destroyed.

```
24  function calcSalesTax($price, $tax=.0675)   {
25      $total = $price + ($price * $tax);
26      return $total;
27  }
```

Demo code:
type_hinting.php,
require_return.php

Recursive

- Function that call themselves
- Used to divide the problem into smaller chunks and solve it little by little, e.g., factorial, palindrome test, sorting
- Good for problem in which we don't know the depth of iteration, e.g., tree traversing, sorting, etc.
- More cool stuff in the data structure course!
- Textbook page 105 – 108

```php
<?php

function reverse_r($str) {
  if (strlen($str)>0) {
    reverse_r(substr($str, 1));
  }
  echo substr($str, 0, 1);
  return;
}

function reverse_i($str) {
  for ($i=1; $i<=strlen($str); $i++) {
    echo substr($str, -$i, 1);
  }
  return;
}

reverse_r('Hello');
echo '<br />';
reverse_i('Hello');

?>
```

Anonymous function

- Function that doesn't have name
- Function whose scope is only when it is declared and used

```
$a = 15;
$example = function() {
    $a += 100;
    echo $a . "\n"; // $a is not known here
};
$example();
```

- Variable can be passed using **use** language construct

```
$a = 15;
$example = function() use ($a){ // use
    $a += 100;
    echo $a . "\n";
    };
$example();
```

- See https://www.php.net/manual/en/functions.anonymous.php

Returning multiple values

- We can only return one variable. How can we return multiple values?
  - Use array and list construct (see page textbook page 101)
  - More in the next part of the lecture!

Be lazy, not sloppy

- Re-use your code, put it in include files, you can use require() or require_once()
- From now on, your functions should always be in include files.

Array

What is an Array?

- Its a value, a scalar variable, we put stuff in it!
- Arrays provide a structure to store multiple pieces of Data
- Arrays store a set or a sequence.
- Arrays can also store other Arrays!
- Any data type (mixed) can be stored in an array -- Remember PHP is weak-typed

- How to visualize an array?
  - For 1 or 2 dimensional array: a table or spreadsheet
  - How about for n>2 ? How about mixed dimensional array?

Defining array

- Using *array* construct (legacy, but is still supported)
- Using [] (JSON notation)
- Each item in an array consists of pairs of key and value
  - Key can be numerical (default) or associative

```php
$user = ['Joe Douglas', 'joe@douglas.com'];
$user[] = 20;
$user[] = 'English';
var_dump($user);

echo "\nUser email ". $user[1];
```

```
array(4) {
  [0] =>
  string(11) "Joe Douglas"
  [1] =>
  string(15) "joe@douglas.com"
  [2] =>
  int(20)
  [3] =>
  string(7) "English"
}

User email joe@douglas.com
```

Associative Arrays

- Associative arrays have keys that are non-numerical, like indexed arrays, elements in an associative array can be indexed by their key.

```
3    $state["Delaware"] = "December 7, 1787";
4    $state["Pennsylvania"] = "December 12, 1787";
5    $state["New Jersey"] = "December 18, 1787";
6    ...
7    $state["Hawaii"] = "August 21, 1959";
```

- Always remember, **key must be unique** whether it is a numeric index or a text-based index.

Demo code: create_array.php

Populating arrays

- You can populate arrays with a pre-defined range by using the range() function for example:

```
$dice = range(1,6);
```

The range() function takes a number of parameters and they can be used to populate arrays for odd numbers, even numbers, the alphabet, parts of the alphabet etc.

```
$letters = range("A","F");
$letters = range('a', 'z');
```

Array functions

- Element management
  - array_unshift – add elements to the front of the array
  - array_shift – remove value from the front of an array
  - array_pop – remove the value from the end of an array and return it.
  - array_push – add values to the end of an array
- Locating element
  - in_array() – Search for a specific value
  - array_key_exists() – returns true if the specified key exists
  - array_search – arguably the most useful because it will return the key of the specified value if it is found.
- Counting element
  - count() – returns the number of elements in an array
  - array_count_values – returns the frequency of a given value in an array.
- Printing: print_r()

Loops to access arrays

- You can use different kinds of loops that have a counter to access array elements
- You can also use loops that don't have an explicit counter like a foreach loop.

```
foreach($products as $fruit) {
    echo $fruit." ";
}
```

- You can access the keys and values while looping with foreach

```
foreach ($products as $key=>$value) {
    echo $key."\t"."=>\t".$value."\n";
}

0       =>      apples
1       =>      oranges
2       =>      bannanas
```

Demo code: associative_array.php

Multidimensional Arrays

- Multidimensional array is an array within an array..

```
$products = array(
    array('TIR', 'Tires', 100),
    array('OIL','Oil',10),
    array('SPK','Spark Plugs', 4)
);
```

- What do you think about this array?

```
$products = array(
    "label"=>"vanparts",
    array('TIR', 'Tires', 100),
    array('OIL','Oil',10),
    array('SPK','Spark Plugs', 4)
);
```

Demo code:
multidimensional.php

Lab 2

- Download Lab2 file from the Blackboard
- You need to submit it next week in the morning of the day of next class (9 am)
- **Make sure to follow the submission guideline!**