switch Statements

```
switch(value) {
    case value1:
        ...
        break;
    ...
    case valueN:
        ...
        break;
    default:
        ...
        break;
}
```

or

```
switch(value):
    case value1:
        ...
        break;
    ...
    case valueN:
        ...
        break;
    default:
        ...
        break;
endswitch;
```
for Loops

```php
for (initialValue; condition; operation)
{
    ...
}
```

or

```php
for (initialValue; condition; operation):
    ...
endfor;
```

while Loops

```php
while (condition)
{
    ...
}
```

or

```php
do
{
    ...
} while (condition);
```

or

```php
while (condition):
    ...
endwhile;
```
Functions

Function name($parameter1, ..., $parameterN)
{
    return value; // Optional
}

- Need not necessarily follow the list of parameters
  - Too many parameters?
    - Extras are ignored
  - Too few parameters?
    - Missing parameters have NULL values
    - Warning message displayed (if so configured on server)

- Can provide default values for parameters
  - function name(..., $param1 = val1, $paramN = valN)

Value vs. Reference

- Pass by value
  - Parameter values are copied
  - Use a variable as a parameter?
    - Variable's value is unchanged
    - This is the default in PHP

- Pass by reference
  - Same data in memory is used
  - Use a variable as a parameter?
    - Variable's value is changed
  - Add & to parameter name
    - e.g., &$paramI

```php
<?php
function dec1($z)
{
    $z--;
}
function dec2(&$z)
{
    $z--;
}
$x = 12; $y = 12;
decl($x); dec2($y);
print "$x and $y";
?>
```

12 and 11
Variable Scope

- Local variables
  - Declared inside of functions
  - Can only be used within that function

- Global variables
  - Declared outside a function
  - Can be used in any subsequent statement
  - Can be used in a function ...
    - if you "re-define" it
    - `global $globalVarName;`

Advanced Embedding

- Control structures can be split across blocks

```php
<?php for($i=0; $i <10; $i++): ?>
    <p>Am I repeating myself?</p>
<?php endfor; ?>

<p>Am I repeating myself?</p>
<p>Am I repeating myself?</p>
<p>Am I repeating myself?</p>
<p>Am I repeating myself?</p>
<p>Am I repeating myself?</p>
```

- Variables can be referenced in different blocks

```php
<?php $d = 25; $m = "October"; ?>
<p>25th of October</p>
```

- For good embedding style ...
  - Avoid printing or echoing HTML tags where possible
  - Use PHP to create, not classify, content
Reducing Embedding

```php
<?php include_once filePath.php ?>
```

- Inserts the text from the given file
  - Any HTML is left alone
  - Any PHP code is processed
    - Output is inserted instead
    - Same file cannot be included again

- Result is like any other block
  - Can use variables and functions in other blocks

---

Reducing Embedding

```php
<?php include_once filePath.php ?>
```

- Other options
  - `require_once`: generates an error if no file at `filePath`
  - `include` and `require`
    - No limit on number of times same file can be used
    - Otherwise the same as their `once` counterparts

- When to embed and when to `include`?
  - Use embedding for small tasks
  - Use `include` for significant sections
    - Large amounts of code
    - Common source between pages
### Arrays

- There are several ways to create an array

```php
$anArray = array();
$anArray[] = "one";
$anArray[] = "two";
```

- `print_r($anArray)`
  - Outputs the values in `anArray`

- `count($anArray)`
  - Returns the number of elements in `anArray`

- `array_sum($anArray)`, `array_product($anArray)`
  - Calculates the sum/product of the entries in `anArray`

- `array_push($anArray, $val)`, `array_pop($anArray)`
  - Adds/removes a value to/from the end of `anArray`

- `array_shift($anArray)`, `array_unshift($anArray, $val)`
  - Removes/adds a value from/to the beginning of `anArray`

- Array size is dynamic
  - Increases to accommodate largest index
  - Unassigned entries will be `NULL`
  - Append values using `$arrayName[] = value`

---

### Some Array Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print_r($anArray)</code></td>
<td>Outputs the values in <code>anArray</code></td>
</tr>
<tr>
<td><code>count($anArray)</code></td>
<td>Returns the number of elements in <code>anArray</code></td>
</tr>
<tr>
<td><code>array_sum($anArray)</code>,</td>
<td>Calculates the sum/product of the entries in <code>anArray</code></td>
</tr>
<tr>
<td><code>array_product($anArray)</code></td>
<td></td>
</tr>
<tr>
<td><code>array_push($anArray, $val)</code>, <code>array_pop($anArray)</code></td>
<td>Adds/removes a value to/from the end of <code>anArray</code></td>
</tr>
<tr>
<td><code>array_shift($anArray)</code>,</td>
<td></td>
</tr>
<tr>
<td><code>array_unshift($anArray, $val)</code></td>
<td>Removes/adds a value from/to the beginning of <code>anArray</code></td>
</tr>
</tbody>
</table>
foreach loops

- Examines each element in an array in order
  - Contained in $entry
- Ignores NULL entries
- Can not alter values in the array
  - Can not assign a value to $entry

```php
foreach($arrayName as $entry)
{
  ...
}
```

or

```php
foreach($arrayName as $entry):
  ...
endforeach;
```