PHP Basics

Basic Information

All php files must end with the .php extension

Php files should reside in a public_html folder (on your turing account)

Set you home directory to 755 permissions using chmod

```
cd ..     (takes you up one level to the CS home directory)
chmod 755 yourWebID   (changes the permission on your folder)
```

Essential Linux commands

ls	directory listing
ls -l	long directory listing, including permissions
cd *folderName*	change directory to a child directory
cd ..	up one directory level to parent directory
cd \ or cd ~	up to home directory level
mkdir *folderName*	make new directory
rm *folder/file*	remove folder/file

Comments

Single line: //

Multi-line: /*   */

Operators

Unary: ++, --

Binary: +, -, *, /, %

Compound Assignment: +=, -=, *=, /=, *=

Relational:

==, === (identical, including type)

!=, <>, !== (not identical)

>, >=, <, <=

Logical: &&, ||, !, and, or, xor

String:

.   (concatenation)

.= (concatenation append)

Variables – Begin with $ and any combination of letters, numbers, _ (underscore), - (hypen)

```
<?php
        $addOn = "again";
        $addOn2 = 'and again';
        echo "First Attempt:  Hello World $addOn  $addOn2 <br />";
        echo "Second Attempt: Hello World $addOn {$addOn2}";

?>
```

Constants

define(“varName”, ‘value”) , case sensitive

```
<?php
        define("DaysInYear", 365);
        echo DaysInYear."<br /><hr />";

        define("DaysInYear", 235);
        echo DaysInYear."<br /><hr />";
?>
```

String Functions

strtolower($v)
strtoupper($v)
ucfirst($word) - only first word uppercase
ucwords($word) – all words uppercase
trim($word) – trims all leading and trailing white space
str_replace(str1, str2, $var) – replaces all occurrence of ***str1*** with ***str2*** in variable ***$var***
strlen($word) – returns length of string
substr($third, position, lng) – substring of ***$third***, beginning at position for length ***lng***
strstr($var, str) – returns the first occurrence of ***str*** in $***var***
str_repeat($word, num) – repeats $***word num*** times
strpos($word, str) – returns the beginning index of ***str*** in $***word***
strchr($word, str) – returns the remaining fragment of $***word*** beginning at ***str***
strcmp($v, $w) - returns true if strings are equivalent (case sensitive)
strcasecmp($v, $w) - returns true if strings are equivalent (case insensitive)

```
<?php
        $first = "The quick brown fox";
        $second = "jumped over the lazy dog.";
        $third = $first;
        $third .= $second;
        $fifth = "          jumped over the lazy dog.   ";

        echo "Lower: ".strtolower($first)."<br />";
        echo "Upper: ".strtoupper($second) ."<br />";
        echo "First Word: ".ucfirst($second) ."<br />";
        echo "All words:  ".ucwords($second) ."<br />";
        echo "Trim: ".trim($fifth) ."<br />";
        echo "Replace: ". trim(str_replace("dog", "cat", $fifth)) ."<br />";
        echo "Length:  ".strlen($first) ."<br />";
        echo "Substring:  ".substr($second, 16, 4) ."<br />";
        echo "Find: ".strstr($second, "lazy") ."<br />";
        echo "Repeat: {$first}".str_repeat($second, 2) ."<br />";
        echo "The position of brown begins at index: ".strpos($third, "brown") ."<br />";
        echo "String Fragment: ". strchr($third, "z") ."<br />";
        echo "String comparison: ".strcmp($first, $third) ."<br />";
        echo "String comparison ignoring case: ".strcasecmp($first, strtoupper($third)) ."<br />";
?>
```

Numerical Functions

abs(a) absolute value

pow(a, b) exponentiation a^b

sqrt(a)   squareroot of a

fmod(a) modulus

rand(); random number

rand(min, max) – random number between min and max, inclusive

round(a) – rounds

ceil(a) – rounds up

floor(a) – rounds down

is_int(v) – returns true if integer

is_float(v) – returns true if floating point

is_numeric(v) – returns true if numeric

is_nan(v) – returns true if NaN (not a number)

Type Juggling & Type Casting

Note: PHP does type juggling. It will try to convert, picking out what it needs and throwing away the rest

---

```
Type Juggling<br />
<?php $count = "2 cats"; ?>
Type: <?php echo gettype($count); ?><br />

<?php $count += 3; ?>
Type: <?php echo gettype($count); ?><br />

<?php $cats = "I have " . $count . " cats."; ?>
Cats: <?php echo gettype($cats); ?><br />
<br />

Type Casting<br />
<?php settype($count, "integer"); ?>
count: <?php echo gettype($count); ?><br />

<?php $count2 = (string) $count; ?>
count: <?php echo gettype($count); ?><br />
count2: <?php echo gettype($count2); ?><br />
<br />

<?php $test1 = 3; ?>
```

```
<?php $test2 = 3; ?>
<?php settype($test1, "string"); ?>
<?php (string) $test2; ?>
test1: <?php echo gettype($test1); ?><br />
test2: <?php echo gettype($test2); ?><br />
```

User-Defined Functions

With a return value:

```
function name($arg1, $arg2, …)  {
        // function body
        return $rtn;
}
```

With a void return:

```
function name2($arg1, $arg2, …)  {
        // function body
}
```

Using global variables:

```
function name3( $arg2)  {
        global $arg1;
        // function body
        return $rtn;
}
```

Using default values (no argument value passed to function):

```
function name4($arg1=5, $arg2= 'tomorrow', …)  {
        // function body
        return $rtn;
}
```

Multiple return values:

With a return value:

```
function name5($arg1, $arg2, …)  {
        // function body
        return array($rtn1, $rtn2);
}
```

Control Structures (like Java)

Selection

If (condition) {	switch (expression) {

//true
} else
//false
}
	case ***c1***:
	break;
	
	case ***c2***:
	break;
	
	default:
	}

Looping

while (test) {
//loop body
}
	foreach($array as $oneValue) {
	//loop body STANDARD ARRAY
	}

for (initialization; test; increment) {
//loop body
}
	foreach($array as $oneKey => $oneValue) {
	//loop body ASSOCIATIVE ARRAY
	}

Standard Arrays

$numbers = array(); **Empty array**
$numbers2 = array(4, 8, 15, 16, 23, 42); **Populated array where indexing begins at 0**
$numbers2[1] ; **References the element at index 1 in $numbers2**

$numbers3 = array(5, "b", $numbers2); **Can mix datatypes within an array. Note that browser will just print Array for $numbers2**

arrays.php

```
<?php
          $numbers2 = array(4, 8, 15, 16, 23, 42);
          $numbers3 = array(5, "bear", $numbers2, "ram", 3.14);

          echo $numbers2[1];
          echo "<pre>"   DEFINES pre-formatted text

          print_r($numbers3)      Stands for PRINT READABLE  and used for debugging

          echo "</pre>":

?>
```

Now, add to arrays.php

```
…
$numbers3[7] = "mountain lion";
```

```
echo “Added element <br />”;
print_r($numbers3);

//Use for-each loop
foreach($numbers3 as $oneElement)  {
            echo $oneElement.”<br /”;
}
echo "Numbers3[6]: ".$numbers3[6]."<br />";
```

<u>Associative Arrays</u>

- key-value pairs instead of indexing
- indexed by key
- use key: arrayName[“keyName”];
- $assoc = array(“fname” => “Kristi”, “lname” => “Davidson);

```
<?php
            $assoc = array(“fname” => “Kristi”, “lname” => “Davidson);
            echo $assoc[“fname”]. “ “.$assoc[“lname”];
?>
```