An Introduction to PHP

Topics

- PHP files
- Comments
- Variables
- Constants
- Operators
- Decisions
- Loop Types
- GET/POST/SESSION/COOKIE

PHP Files

- PHP files are text files that are interpreted, PHP code can be declared inside any text file by using the following tags:
- <?php … ?> - Canonical PHP Open and Close tags
- <? … ?> - Short tags (SGML Style)
- <% … %> - ASP Style tags
- <script language = “PHP”> … </script> - HTML Style tags

PHP Files

- PHP files are text files that are interpreted, PHP code can be declared inside any text file by using the following tags:
- <?php … ?> - Canonical PHP Open and Close tags
- <? … ?> - Short tags (SGML Style)
- <% … %> - ASP Style tags
- <script language = “PHP”> … </script> - HTML Style tags

Print

```
1   <?php
2   $illy = “Dr. Seuss”;
3
4   print “this”;
5   print “that”;
6
7       Print <<END
8       Can you find
9       The $illy
10      Hat?
11      END
12  ?>
13
```

Multi-line printing

```
1   <?php
2
3       //Newlines will be outputted
4       print "You can get help from teachers,
5       but you are going to have to learn a lot by yourself,
6           sitting alone in a room. - Dr. Seuss";
7
8   ?>
```

Statements for printing

- print() – outputs data that is passed in
- echo() does the same
  - The difference between echo and print is echo has no return value and is a bit faster.

Comments

Make good comments, not bad?

Bad comments make me sad.

Your comments should explain what you are trying to do, if you are beginning its ok to use a lot of comments, typically you will not be faulted for putting too many comments. You should add your initials to the end of your comment:

```
//This is my comment –SW May 7, 2018
```

printf() – good for interpolating variables and text.

- $val = “value”;
- printf(“sometext %s”, $val);
- Printf is good because you can use type identifiers to tell the computer how to format your text output to a certain data type (don’t worry data types are coming up soon)

- printf("$%.2f", 43.2); // $43.20

Type Specifiers and printf

Table 3-1. Commonly Used Type Specifiers

Type	Description
%b	Argument considered an integer; presented as a binary number
%c	Argument considered an integer; presented as a character corresponding to that ASCII value
%d	Argument considered an integer; presented as a signed decimal number
%f	Argument considered a floating-point number; presented as a floating-point number
%o	Argument considered an integer; presented as an octal number
%s	Argument considered a string; presented as a string
%u	Argument considered an integer; presented as an unsigned decimal number
%x	Argument considered an integer; presented as a lowercase hexadecimal number
%X	Argument considered an integer; presented as an uppercase hexadecimal number

Comments…

```
//This Comment
#That Comment

/*
    Here is a
    multi-line
    comment. Comment.
    */
```

Case Sensitivity

- PHP functions are case sensitive
- Variable Assignments are case sensitive.

Syntax

- Expressions are terminated by semicolons;
- Expressions are made up of tokens such as numbers (3.194), strings (.text.), variables ($yntax) and constants (true).
- There are also some special words if, then, else, case, switch, foreach etc… These are typically words that make up loops or express conditional statements and Boolean logic.

Identifiers – What to call stuff

- Always pick good names for user defined variables, functions and objects.
  - Must begin with a letter or underscore.
  - Can only consist of letters, numbers and underscores
  - Can be any length
  - Are case sensitive
  - Can’t be identical to any of PHP’s pre-defined keywords.

Variables

- Where data is stored!
- Value is always its most recent assignment
- Always start with a “$”
- Identifiers are case sensitive.
- Assigned with the assignment operator “=“

Variables

- You do not need to instantiate your variables but you should as a matter of good form.
- Weak typed (variables do not need to be instantiated with a type)
- If you use a variable before assigning it, it will have a default value
- PHP does casting for you! (most of the time)…

Value assignment vs Reference Assignment

- Value Assignment:
  - Most common
  - Copies the variable contents on assignment

- Reference Assignment
  - Less common
  - “Alias” or “Pointer” to a variable
  - Equals operator as an ampersand sign appended to it:

```
$value1 = "Hello";
$value2 =& $value1;
```

Variable Scope

- **Local Variable** - Variables defined inside of functions, only visible inside the function.
- **Function Parameters** – arguments accepted in the function declaration. These arguments accept values that come from outside the function, once passed in they are no longer valid after the function exits.
- **Static Variables** – These variables out-live a functions execution and are available the next time a function is run.
- **Global Variables** – accessed by any part of the program, you must explicitly declare the global variable inside the function which it is modified.
- **Super Global Variables** – these are made available as part of the environment, these are pre-defined and usually environment specific.

There are also { Braces }

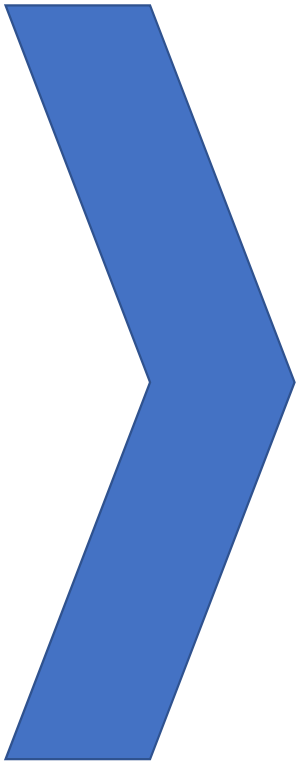
```
1   <?php
2
3   if (2 + 2 == 4) {
4       print "Truth!";
5   }
6
7   if (4 == 2 + 2)
8       print "Concise Truth!";
9
10  ?>
```

Data Types in PHP

- Integers – whole numbers: 1,5, -6, 45565
- Float – floating-point numbers: 4.5676, 4.0, 8.7e4
- Booleans – true or false, nothing else.
- NULL – NULL – yes that's right NULLs are always NULL
- Strings – sequences of characters called 'strings'
- Arrays – named indexes of collections- store data using numeric indexes or associative indexes (more on this next class).
- Objects – Hold instantiated classes

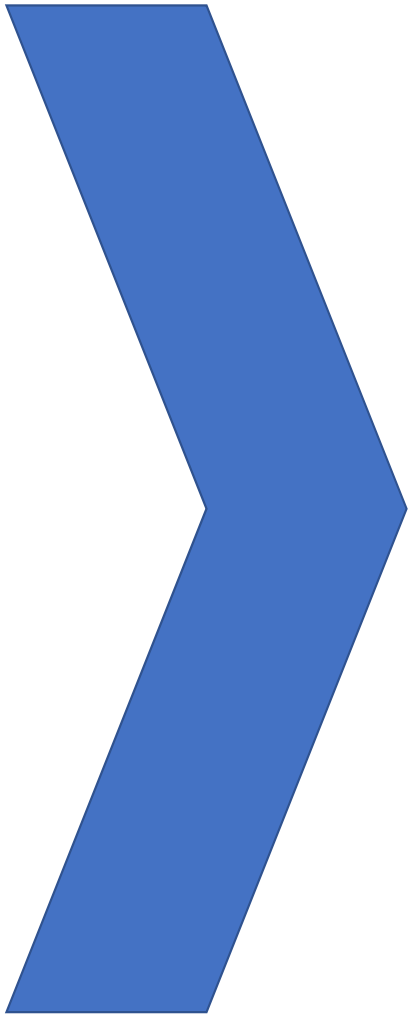
If ever there was a function to remember…

```php
<?php

$var = 2.0002;
var_dump($var);

?>
```



```
float(2.0002)
```

```php
<?php
//variable types
$float = 2.0002;
var_dump($float);
$int = 2;
var_dump($int);
$string = "text";
var_dump($string);
$char = 'c';
var_dump($char);
$null = null;
var_dump($null);
$arr = array(1);
var_dump($arr);

?>
```



```
php C:\temp\test.php

float(2.0002)
int(2)
string(4) "text"
string(1) "c"
NULL
array(1) {
  [0]=>
  int(1)
}
```

Converting data types

- When programming sometimes we need to convert data types, this could be because we need to call a function for example that takes an string value but we want to pass in an integer. We have two options:
  - Cast the variable to a different type using a cast operator
  - Convert using settype()
  - Type juggling – be careful with this one!

Cast Operator

- (int), (integer) - **cast** to integer.
- (bool), (boolean) - **cast** to boolean.
- (float), (double), (real) - **cast** to float.
- (string) - **cast** to string.
- (array) - **cast** to array.
- (object) - **cast** to object.
- (unset) - **cast** to NULL.

Retrieving the type of a variable

- You may use the gettype() function to return the type of a provided variable.
- echo gettype(“1.0”);

Constants

- A constant is a value that cannot be modified
- Practically these are used to configure applications and often placed into a configuration file (for example: config.inc.php)
- Constants are defined using the define keyword.

define(CONST_NAME, *value);*

Operands

- These are the input of the expression
- The values or variables you are starting with

```
10    $a++; // $a is the operand
11    $sum = $val1 + val2; // $sum, $val1 and $val2 are operands
```

Operators

- An operator is the symbol that specifies the action in the expression such as adding, or decrementing.
- Depending on the operator you use PHP will convert the type of the operand.

Expressions

- Phrases representing a particular action in a program.
- Expressions all have an operand and one or more operators

```
3   $a = 5;                    // assign integer value 5 to the variable $a
4   $a = "5";                  // assign string value "5" to the variable $a
5   $sum = 50 + $some_int;     // assign sum of 50 + $some_int to $sum
6   $wine = "Zinfandel";       // assign "Zinfandel" to the variable $wine
7   $inventory++;              // increment the variable $inventory by 1
```

Operators

- There are 5 kinds:
- Assignment – these are for modifying variables
- Arithmetic – these do math.
- Comparison – these compare.
- Logical or Relational – these do logic.
- Conditional – these decide.

Operator Precedence

- The order in which operators evaluate the operands around them.
- Think BEDMAS (Brackets, Exponents, Division, Multiplication, Addition, Subtraction)

Operator Associativity

- Associativity is how operations of the same precedence are evaluated.
  - Right to Left
  - Left to Right

```
$value = 3 * 4 * 5 * 7 * 2;
$value = ((((3 * 4) * 5) * 7) * 2);
```

Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Arithmetic – these do math.

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison – these compare.

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Conditional

- Evaluate two Boolean expressions and return the result.

```php
<?php

$a = 1;
$b = 2;

//If 1 is less than two, assign the result variable to 1 or 2.
$result = ($a < $b ) ? "one is less than two" : "two is less than one?";
echo $result;

?>
```

String Interpolation

- Double Quotes – denote a string however these are very cumbersome when dealing with HTML because HTML has a lot of quotes in it regardless
- Single Quotes – denote a string but are a lot easier to deal with as they occur less frequently in HTML and therefore do no t have to be escaped.
- …. So whats “escaped” you ask?

Escape Sequences

- Escape sequences are special characters that can be used to either interpret text literally or for special characters

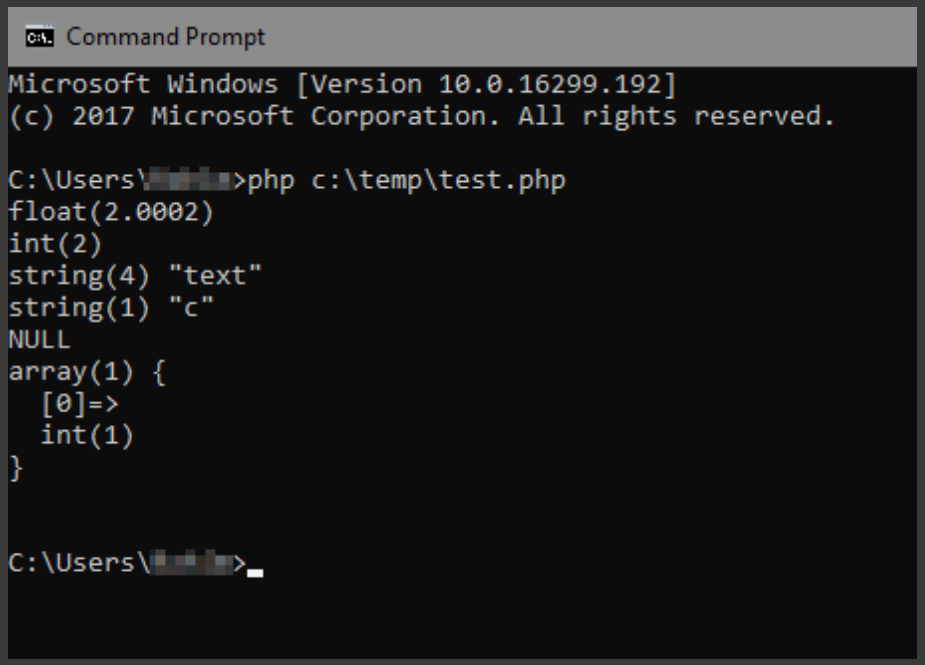
Sequence	Description
\n	Newline character
\r	Carriage return
\t	Horizontal tab
\\	Backslash
\$	Dollar sign
\"	Double quote
\[0-7]{1,3}	Octal notation
\x[0-9A-Fa-f]{1,2}	Hexadecimal notation

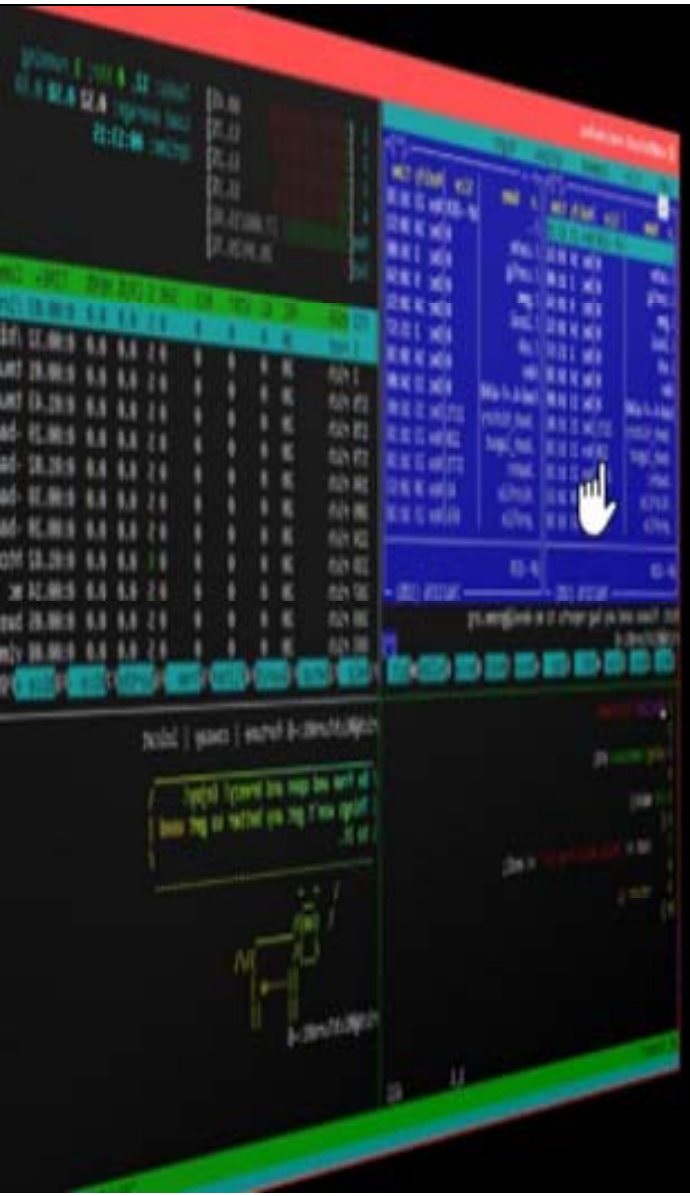
Using the Console

- PHP scripts can be invoked via the console.
- PHP should be set in your $PATH to do this.
- PHP is a very useful language to know and script for as it runs on every major platform.
- It was earlier known as a web language and then unix administrators started shell scripting with it.
- You can get to the console in Code with: Ctrl + ~

Windows Console

- Much improved in Windows 10



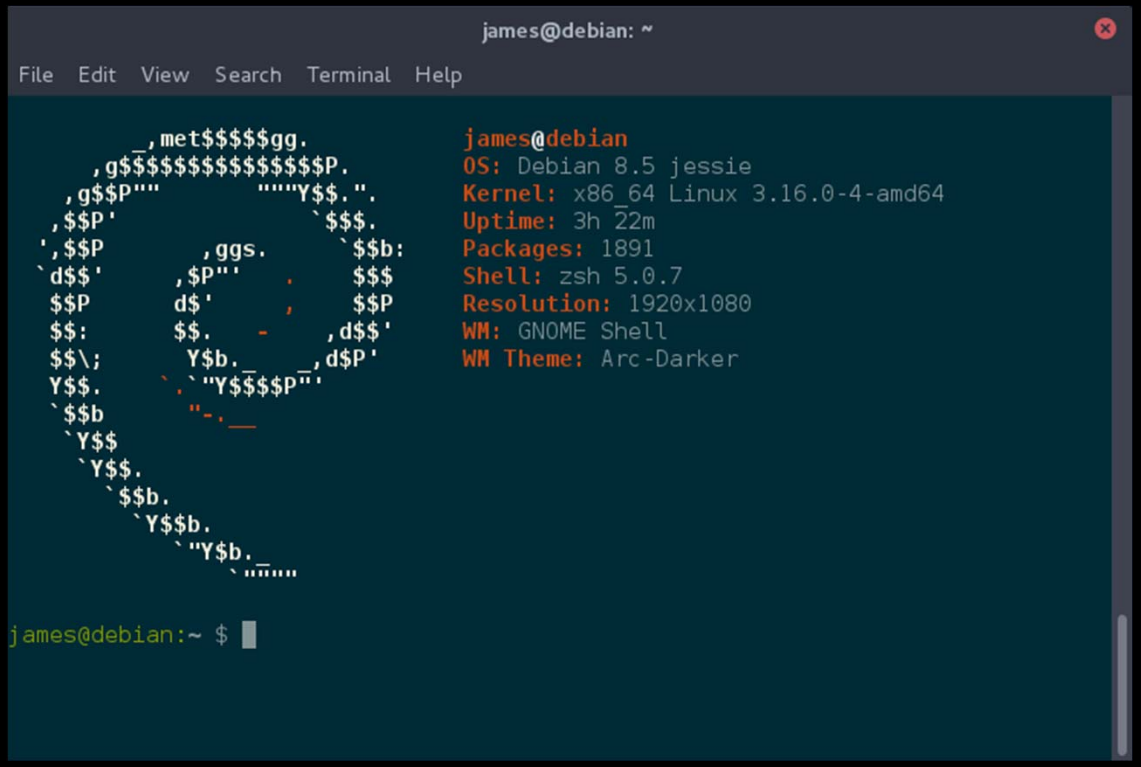


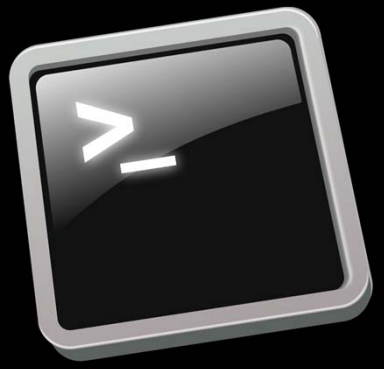
Windows Subsytem for Linux

- Even better than the Command Console…
- Do almost all the GNU/Linux things!

- https://blogs.msdn.microsoft.com/commandline/learn-about-windows-console-and-windows-subsystem-for-linux-wsl/

But it’s the best in Linux ☺



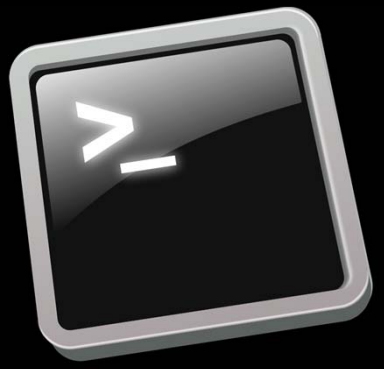


Reading in from the Console

- stream_get_line() will allow you to read the console for input:

```php
<?php

echo "Please enter your name:";
$name = stream_get_line(STDIN, 1024, PHP_EOL);
echo "Hello $name!";

?>
```

```
PS C:\Users\[redacted]> php C:\temp\test.php
Please enter your name:Sam
Hello Sam!
```



If/else statements

```php
<?php

$a = 3;

if ($a > 5) {
    //If the condition above is true...
    echo "$a is greater than 5";
} else {
    //Otherwise ....
    echo "$a is less than 5";
}

?>
```

Elseif statements

- Elseif can be used to chain If statements – don't over use this.

```php
<?php

$a = 5;

if ($a > 5) {
    //If the condition above is true...
    echo "$a is greater than 5";
} elseif ($a < 5)   {
    //If the condition above is true
    echo "$a is less than 5";
} else {
    //If its not greather than or less than... it must be equal!
    echo "$a is equal to 5";
}
```

Switch statement

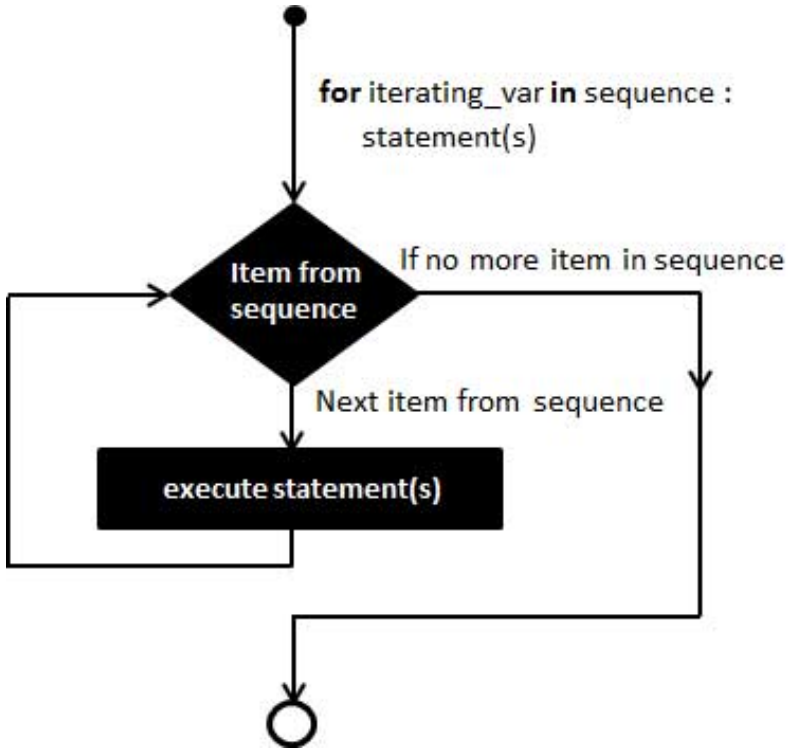
```php
<?php

    $today = date("D");

    switch ($today){

        case "Fri":
            echo "Thank goodness its Friday!";
            break;
        case "Sat": //A little trick.
        case "Sun":
            echo "Have a good weekend!";
            break;

        default:
            echo "Happy $today";
    }

?>
```

Loops

- for
- while
- do while

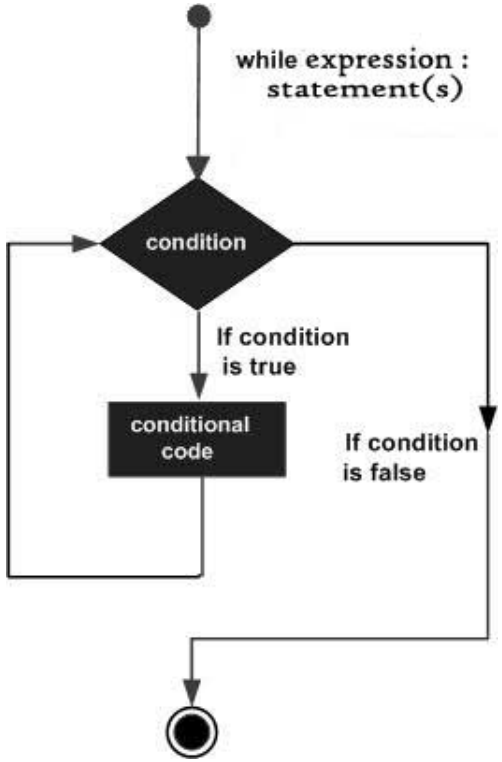
for

```php
<?php

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

?>
```



while

```php
<?php

$v = 0;

while ($v <= 10) {

    echo $v . "\n";
    $v++;
}

?>
```



do…while

```php
<?php
$start = date("s");
do {

    //do stuff
    echo ".";
    sleep(1);

} while (date("s") % 10 != 0)

?>
```

break; and continue;

- Use break; to ‘break out’ of a loop.
- Use continue; to stop the current iteration (cycle) of the loop and start again.

break;

```php
<?php

while(true) {

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

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

?>
```

```
PS C:\Users\[redacted]> php C:\temp\test.php
43-44-45-46-47-48-49-
```

continue;

```php
<?php

while(true) {

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

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

?>
```

```
PS C:\Users\[redacted]> php C:\temp\test.php
23-24-26-27-28-29-
```

Oh and one more quick thing!

- We haven't covered strings today but you can use the "." operator to concatenate two strings
- You can use the "\n" for a newline character!

```php
<?php

$one = "orange";
$two = "apple";
$three = "lemon";


echo $one.$two.$three;
echo "\n";

echo $one."\n".$two."\n".$three."\n";

?>
```

```
orangeapplelemon
orange
apple
lemon
```

References

https://www.tutorialspoint.com/php/php_syntax_overview.htm
https://sayingimages.com/best-dr-seuss-quotes-pictures/
https://www.w3schools.com/php/php_echo_print.asp