Embedded PHP

if/else statement

```php
if (condition) {
  statements;
} else if (condition) {
  statements;
} else {
  statements;
}
```

PHP

- can also say `elseif` instead of `else if`

for loop

```
for (initialization; condition; update) {
  statements;
}
```
PHP

```
for ($i = 0; $i < 10; $i++) {
   print "$i squared is " . $i * $i . ".\n";
}
```
PHP

while loop (same as Java)

```
while (condition) {
  statements;
}
```
PHP

```
do {
  statements;
} while (condition);
```
PHP

- break and continue keywords also behave as in Java

String type

```php
$favorite_food = "Ethiopian";
    print $favorite_food[2];              # h
```

- zero-based indexing using bracket notation
- string concatenation operator is . (period), not +
  - 5 + "2 turtle doves" produces 7
  - 5 . "2 turtle doves" produces "52 turtle doves“
- can be specified with "" or ' '

String functions

```
# index  0123456789012345
$name = "Austin Weale";
$length = strlen($name);                # 16
$cmp = strcmp($name, "Linda Guo");      # > 0
$index = strpos($name, "s");            # 2
$first = substr($name, 7, 4);           # "Weal"
$name = strtoupper($name);              # "AUSTIN WEALE"        PHP
```

Name	Java Equivalent
strlen	length
strpos	indexOf
substr	substring
strtolower, strtoupper	toLowerCase, toUpperCase
trim	trim
explode, implode	split, join

Interpreted strings

```
$age = 16;
print "You are " . $age . " years old.\n";
print "You are $age years old.\n";     # You are 16 years old. PHP
```

- strings inside " " are interpreted
  - variables that appear inside them will have their values inserted into the string
- strings inside ' ' are not interpreted:

```
print 'You are $age years old.\n';    # You are $age years old.\n
PHP
```

- if necessary to avoid ambiguity, can enclose variable in {}:

```
print "Today is your $ageth birthday.\n";     # $ageth not found
print "Today is your {$age}th birthday.\n";
PHP
```

Arrays

```php
$name = array();                              # create
$name = array(value0, value1, ..., valueN);

$name[index]                                  # get element value
$name[index] = value;                         # set element value
$name[] = value;                              # append
```

```php
$a = array();      # empty array (length 0)
$a[0] = 23;        # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";    # add string to end (at index 5)
```

- to append, use bracket notation without specifying an index
- element type is not specified; can mix types

Array functions

function name(s)	description
count	number of elements in the array
print_r	print array's contents
array_pop, array_push, array_shift, array_unshift	using array as a stack/queue
in_array, array_search, array_reverse, sort, rsort, shuffle	searching and reordering
array_fill, array_merge, array_intersect, array_diff, array_slice, range	creating, filling, filtering
array_sum, array_product, array_unique, array_filter, array_reduce	processing elements

Array function example

```
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
  $tas[$i] = strtolower($tas[$i]);
}                                    # ("md", "bh", "kk", "hm", "jp")
$morgan = array_shift($tas);         # ("bh", "kk", "hm", "jp")
array_pop($tas);                     # ("bh", "kk", "hm")
array_push($tas, "ms");              # ("bh", "kk", "hm", "ms")
array_reverse($tas);                 # ("ms", "hm", "kk", "bh")
sort($tas);                          # ("bh", "hm", "kk", "ms")
$best = array_slice($tas, 1, 2); # ("hm", "kk")
```

- the array in PHP replaces many other collections in Java
  - list, stack, queue, set, map, ...

The foreach loop

```
foreach ($array as $variableName) {
  ...
}
```
PHP

```
$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($stooges); $i++) {
  print "Moe slaps {$stooges[$i]}\n";
}
foreach ($stooges as $stooge) {
  print "Moe slaps $stooge\n";  # even himself!
}
```

- a convenient way to loop over each element of an array without indexes

Expression block example

```
<!DOCTYPE html>
<html>
  <head><title>CSE 154: Embedded PHP</title></head>
  <body>
    <?php for ($i = 99; $i >= 1; $i--) { ?>
      <p> <?= $i ?> bottles of beer on the wall, <br />
          <?= $i ?> bottles of beer. <br />
          Take one down, pass it around, <br />
          <?= $i - 1 ?> bottles of beer on the wall. </p>
    <?php } ?>
  </body>
</html>
```

PHP

Common errors: unclosed braces, missing = sign

```
<body>
    <p>Watch how high I can count:
      <?php for ($i = 1; $i <= 10; $i++) { ?>
        <? $i ?>
    </p>
  </body>
</html>
```
PHP

- </body> and </html> above are inside the for loop, which is never closed
- if you forget to close your braces, you'll see an error about 'unexpected $end‘
- if you forget = in <?=, the expression does not produce any output

Complex expression blocks

```php
  <body>
    <?php for ($i = 1; $i <= 3; $i++) { ?>
      <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
    <?php } ?>
  </body>
```

PHP

output

- expression blocks can even go inside HTML tags and attributes