PHP

control structures - loops
indexed arrays

PHP files are processed top to bottom in sequence

```
<html>
<?php ... ?>
<head>
<?php ... ?>
<title>... <?php ... ?> ...</title>
</head>
<body>
<p>
<?php ... ?>
</p>
</body>
</html>
```

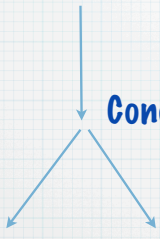
Starting at the top

Working down to the bottom

The control flow

But sometimes we need to have choices / alternatives

Start at the top



Condition here

Outcome 1 Outcome 2

Work down to the bottom

Can be complex flows

Done with an if or a switch statement

```
if (expression)
  statement
```

Perform the statement if the expression is true

```
if (expression)
  statement1
else
  statement2
```

Perform statement1 if the expression is true otherwise statement2

```
if (expression){
   statement;
   statement;
   }
else {
   statement;
   statement;
   };
```

Perform blocks of statements if the expression is true otherwise ...

```
if (expression){
   statement;
   }
else
 if (expression){
       statement;
       }
    else {
         statement;
         };
```

If there are many choices a nested series of if statement may be required

Note how tabs are used to help read the code - do the same with your code

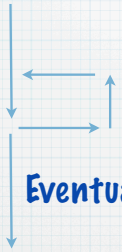
```
switch($name) {
      case 'value 1 of
   name':
        // do something
        break;
      case 'value 2 of
   name':
        // do something
        break;
      case 'value 3 of
   name':
        // do something
        break;
      case 'value 4 of
   name':
        // do something
        break;
    }
```

When the if statement has many sub-if parts, a switch statement may be better

Best to choose this when $name has a limited set of values known in advance

Sometimes we need to do things many times

Start



Round the Loop

Eventually come out of the loop ...

Loops / Iteration / doing things over and over and over and over ....

Three standard loop types

for

while ...

do ... while

Loops / Iteration / doing things over and over and over and over ....

Three standard loop types - for loops today

for
while ...
do ... while

Choose a for-loop if the number of times the loop will run is known 'in advance'

The loop will run 4 times
The loop will run 1000 times
The loop will run 'n' times

Loops / Iteration / doing things over and over and over and over ....

Three standard loop types - for loops today

for
while ...
do ... while

...or you are processing an array (more on this in a minute)

for loop

The structure of a for statement is:

```
for (start; condition; change amount)
   statement
```

or with many statements -

```
for (start; condition; change amount){
   statement;
   statement;
   statement;
   statement;
};
```

for loop

The structure of a for statement is:

```
for (start; condition; change amount)
```

Declare a counter in here, assign it an initial value

for loop

The structure of a for statement is:

```
for ($i=0; condition; change amount)
```

Declare a counter in here, assign it an initial value

for loop

The structure of a for statement is:

```
for ($i=0; condition; change amount)
```

Declare a counter in here, assign it an initial value

Here i'm using $i - but you can use any name

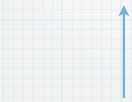
$counter *$y* *$myCount*

$loop *$timesRound*

for loop

The structure of a for statement is:

```
for ($i=0; condition; change amount)
```



Condition goes here - if true the loop will continue

for loop

The structure of a for statement is:

```
for ($i=0; $i<20; change amount)
```


Condition goes here - if true the loop will continue

for loop

The structure of a for statement is:

```
for ($i=0; $i<20; change amount)
```

Change the counter value here

for loop

The structure of a for statement is:

```
for ($i=0; $i<20; $i++)
```

'$i++ is a shorthand for 'add 1 to i'

for loop

```
$myNumber=rand(1,20);
print "<p>This loop will work $myNumber times</p>";
for($i=1;$i<=$myNumber;$i++){
    print "Going round the loop: $i<br />";
};
```

Note how we can use the $i loop counter inside the loop

Don't start changing its value inside the loop unless you know what you are doing

Arrays

- indexed arrays
- associative arrays

Much of this material is explained in PHP programming 2nd Ed. Chap 5

Arrays

- indexed arrays - today
- associative arrays

Much of this material is explained in PHP programming 2nd Ed. Chap 5

Arrays

Sometimes we have a set of values that should have a single name

Can use a structure called an array to store these

A series of boxes with the same name

Arrays

So how do we get at the individual values inside the array?

Use a number - the index

Index is indicated in square brackets

indexed arrays

Uses consecutive integers to index the cells

$colours

0	1	2	3
Red	Green	Blue	Yellow

```
print $colours[1];
```

Green

indexed arrays

Uses consecutive integers to index the cells

$colours

0	1	2	3
Red	Green	Blue	Yellow

```
$colours[2]="Purple";
```

indexed arrays

Uses consecutive integers to index the cells

$colours

0	1	2	3
Red	Green	Purple	Yellow

```
$colours[2]="Purple";
```

indexed arrays

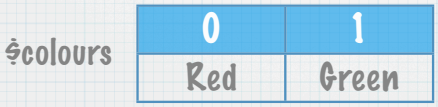
Use simple assignment to create the array

```
$colours[0]="Red";
```

indexed arrays

Use simple assignment to create the array

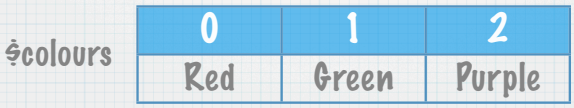
```
$colours[0]="Red";
$colours[1]="Green";
```



indexed arrays

Use simple assignment to create the array

```
$colours[0]="Red";
$colours[1]="Green";
$colours[2]="Purple";
```



indexed arrays

If all the values are known in advance, use the reserved word array

```
$colours = array ("Red","Green","Purple","Yellow");
```

index starts from 0

indexed arrays

To add an element to the end, use []

```
$colours = array ("Red","Green","Purple","Yellow");
$colours[] = "Black";
```

indexed arrays

To add an element to the end, use []

```
$colours = array ("Red","Green","Purple","Yellow");
$colours[] = "Black";
```

useful functions

See appropriate references for more useful array functions

function	explanation
count()	no of array cells

indexed arrays

To process all the elements in an array, use a loop

```
$colours = array ("Red","Green","Purple","Yellow");
for($i=0;$i<count($colours);$i++){
  print $colours[$i]."<br />";
  };
```

Red
Green
Purple
Yellow