introduction to

SCRIPTING, DATABASES, SYSTEM ARCHITECTURE

PHP III: functions, arrays, associative arrays

Claus Brabrand

(((brabrand@itu.dk)))

Associate Professor, Ph.D.

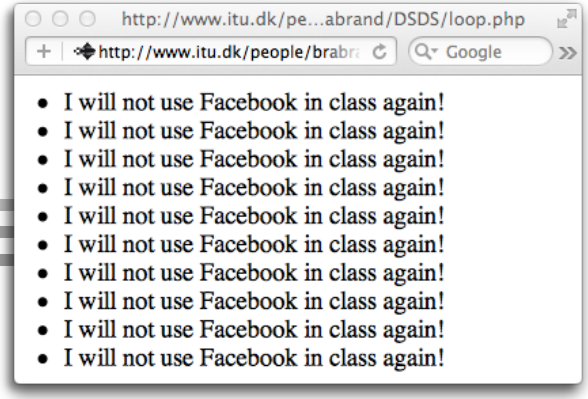
(((Programming, Logic, and Semantics)))

IT University of Copenhagen

Agenda

- RECAP ('while' and 'for' loops)
- FUNCTIONS
- INCLUDING FILES
- ARRAYS
- ASSOCIATIVE ARRAYS ('foreach')

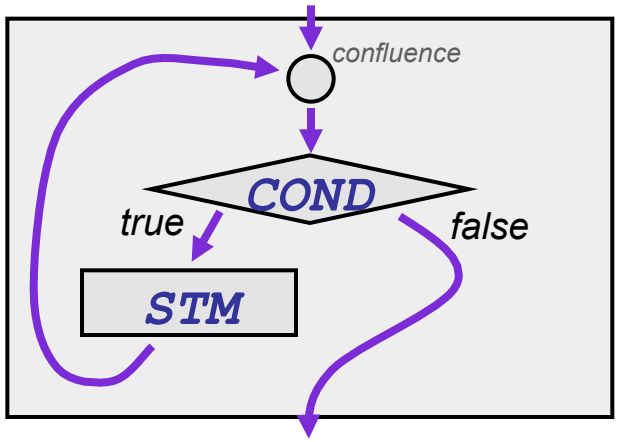
The 'while' loop



- Syntax:

```
while ( COND ) {
      STM
}
```

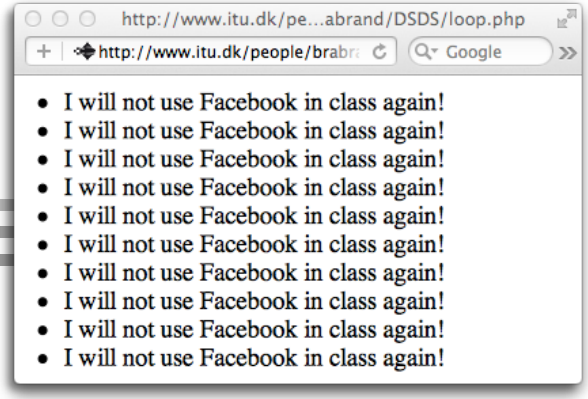
- Semantics:



- Example:

```
$count = 0;                                          // initialization
while ( $count < 10 ) {
  echo "<li> I will not use Facebook in class again! </li>" ;
  $count++ ;                                         // incrementation
}
```

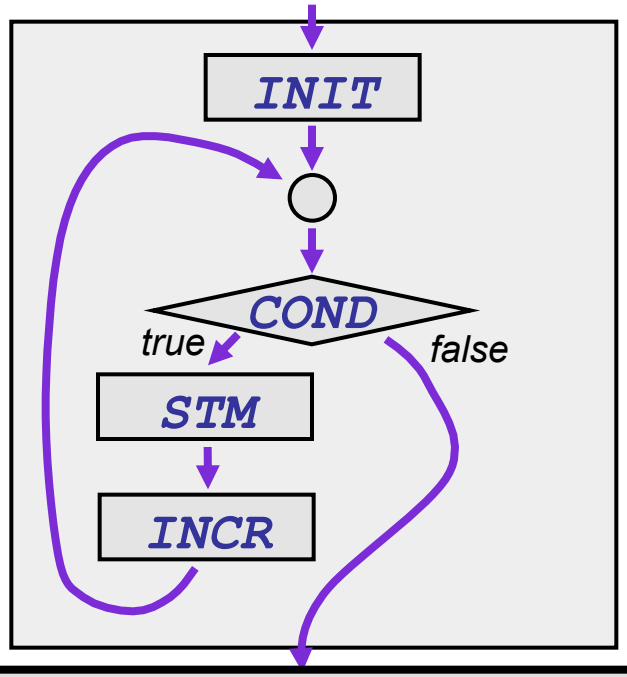
The 'for' loop



- Syntax:

```
for ( INIT ; COND ; INCR ) {
    STM
}
```

- Semantics:

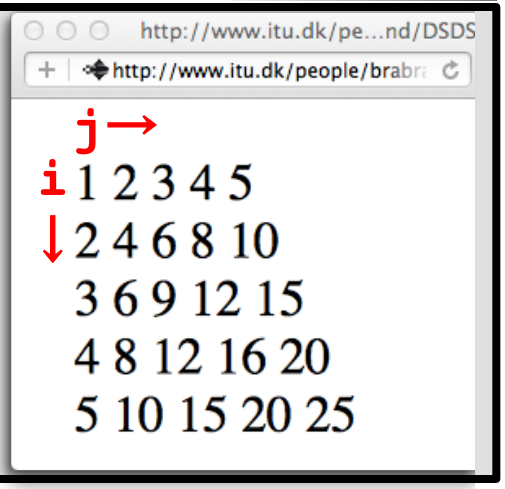


- Example:

```
for ( $count = 0 ; $count < 10 ; $count++ ) {
    echo "<li> I will not use Facebook in class again! </li>" ;
}
```

Nested Loops (5x5 Mult Table)

```
for ( $i=1 ; $i <= 5 ; $i++ ) {
   for ( $j=1 ; $j <= 5 ; $j++ ) {
      $mult = $i * $j ;
      echo "$mult " ;
   }
   echo "<br/>" ;
}
```




```
<table border="1">
<?php
   for ( $i=1 ; $i <= 5 ; $i++ ) {
      echo "<tr>" ;
      for ( $j=1 ; $j <= 5 ; $j++ ) {
         $mult = $i * $j ;
         echo "<td width='30'>$mult</td>" ;
      }
      echo "</tr>" ;
   }  ?>
</table>
```

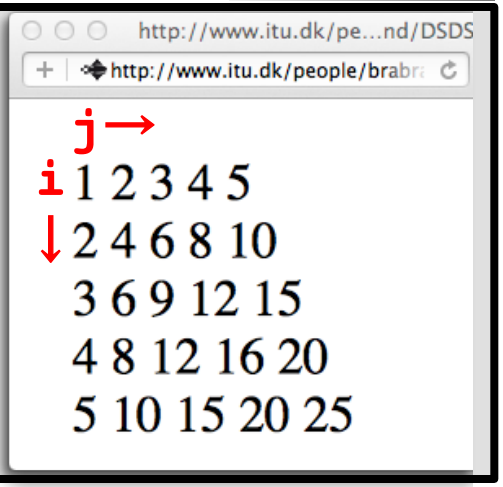
j→

i ↓

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Nested Loops (5x5 Mult Table)

```php
for ( $i=1 ; $i <= 5 ; $i++ ) {
  for ( $j=1 ; $j <= 5 ; $j++ ) {
    $mult = $i * $j ;
    echo "$mult " ;
  }
  echo "<br/>" ;
}
```




```php
<pre>
<?php
  for ( $i=1 ; $i <= 5 ; $i++ ) {
    // start of row $i
    for ( $j=1 ; $j <= 5 ; $j++ ) {
      $mult = $i * $j ;
      echo "$i x $j = $mult <br/>" ;
    }
    echo "<p/>" ;
  }  ?>
</pre>
```

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10

...

5 x 5 = 25
```

Agenda

- RECAP ('while' and 'for' loops)
- FUNCTIONS
- INCLUDING FILES
- ARRAYS
- ASSOCIATIVE ARRAYS ('foreach')

Functions: Declare and Invoke

- A ***function*** is a piece of code we give a name:

```
<?php

   function printMyName() {        // function declaration
      echo "<b>John Doe</b>" ;     // function body
   }

   echo "Hello, my name is " ;
   printMyName();                  // function invocation

?>
```

Hello, my name is **John Doe**

Functions: Code Reuse

- A function allows for **code reuse!**:

```
<?php

   function printMyName() {        // function declaration
      echo "<b>John Doe</b>" ;     // function body
   }

   echo "Hello, my name is " ;
   printMyName();                  // function invoked

   echo "<p/>" ;

   printMyName();                  // function invoked again!
   echo " is indeed my name." ;

?>
```

Hello, my name is **John Doe**

John Doe is indeed my name.

Functions: One Modification

- We **only** have to **modify** the **declaration**:

```
<?php

   function printMyName() {
      echo "<b>Jane Doe</b>" ;    // modification
   }

   echo "Hello, my name is " ;
   printMyName();                 // changes here

   echo "<p/>" ;

   printMyName();                 // ...and here!
   echo " is indeed my name." ;

?>
```

Hello, my name is **Jane Doe**

Jane Doe is indeed my name.

Functions (with arguments)

- ***Functions*** can take arguments:

```
<?php

   function printName($name) {
      echo "Hello, my name is <b>$name</b>" ;
   }

   printName("John Doe");

   echo "<p/>" ;

   printName("Barack Obama");
?>
```

Hello, my name is **John Doe**

Hello, my name is **Barack Obama**

- Gives us a chance to ***parameterize*** the code
  (to use the function differently in different contexts)

Functions (multiple arguments)

- ***Functions*** can take multiple arguments:

```
<?php

   function printName2($first, $last) {
      echo "Hello, my name is $last, <em>$first $last</em>." ;
   }

   printName2("James", "Bond");

   echo "<p/>" ;

   printName2("John", "Doe");

?>
```

Hello, my name is Bond, *James Bond*.

Hello, my name is Doe, *John Doe*.

Functions (can return a value)

- Functions are allowed to *"return"* a value:

```
function mult($x, $y) {
   $result = $x * $y ;
   return $result ;
}

$val = mult(5,7) ;                          // function invocation
echo "The result of 5*7 is: <b>$val</b>" ;
```

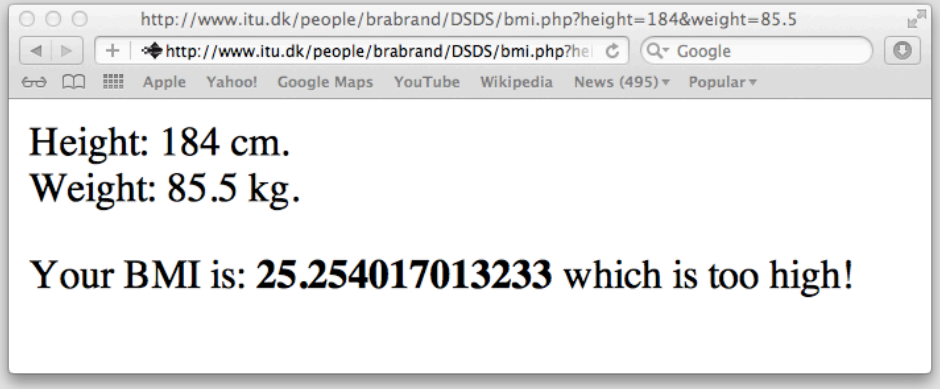
The result of 5*7 is: **35**

```
function emphasize($text) {
   $emph = "<em><b><font color='red'>$text</font></b></em>" ;
   return $emph ;
}

$x = emphasize("never") ;                   // function invocation
echo "I will $x use Facebook in class again!" ;
```

I will ***never*** use Facebook in class again!

EXERCISE: Extract "BMI Computation" as a Function!

```
<!-- This file is located at: "http://www.itu.dk/people/brabrand/DSDS/bmi.php" -->
<html>
    <body>
        <?php
      /* Body Mass Index (BMI) calculator */

      $h = $_REQUEST['height'] ;
      $w = $_REQUEST['weight'] ;

      echo "Height: $h cm.<br/>" ;
      echo "Weight: $w kg.<p/>" ;

      $bmi = $w / (($h / 100) * ($h / 100)) ;
      echo "Your BMI is: <b>$bmi</b> " ;

      if ( $bmi < 20.0 ) {
          echo "which is too low!" ;
      } elseif ( $bmi > 25.0 ) {
          echo "which is too high!" ;
      } else {
          echo "which is normal." ;
      }
      ?>
    </body>
</html>
```



Q: Which of these is easier to reuse and understand (and why)?

Return Result vs Print Result?

- Function returns result:

Easy(ier) to reuse + understand!

```
function emphasize($text) {
    $emph = "<em><b><font color='red'>$text</font></b></em>" ;
    return $emph ;
}

$x = emphasize("never") ;        // function computes sth for us
echo "I will $x use Facebook in class again!" ;
```

I will *never* use Facebook in class again!

- Function prints result (side-effect):

Hard(er) to reuse + understand!

```
function emphasize2($text) {
    echo "<em><b><font color='red'>$text</font></b></em>" ;
}

echo "I will " ;
emphasize2("never") ;  // function has "side-effect" (echo)
echo " use Facebook in class again!" ;
```

I will *never* use Facebook in class again!

We can use built-in functions

We can use ***built-in functions:***

- ***date(format)***

```
$today = date("d-m-Y") ;
echo "Today is: " ;
echo $today ;
```

Today is: 30-9-2011

- ***round(number)***

```
$pi = 3.1415926 ;
$pi_ish = round($pi) ; // function call
echo "Pi (&pi;) is approximately: $pi_ish" ;
```

Pi (π) is approximately: 3

- ***strlen(string)***

```
$str = "Hello World" ;
$len = strlen($str) ;
echo "The length of '$str' is: <b>$len</b>" ;
```

The length of 'Hello World' is: **11**

Invaluable SPAM Functions

- *strtoupper(string)*

```
$txt = "Dear friend, I have a business opportunity ..." ;
$upper = strtoupper($txt) ;
echo $upper ;
```

DEAR FRIEND, I HAVE A BUSINESS OPPORTUNITY ...

- *mail(to,subject,message,header)*

```
$sender    = "Master Kofu George" ;
$email     = "kofu.george@yahoo.com" ;
$recipient = "alle@itu.dk" ;
$mail_body = $upper ;
$subject   = "Business Opportunity" ;
$header    = "From: " . $sender . " <" . $email . ">\r\n" ;

mail($recipient, $subject, $mail_body, $header); // send email
```

- You should now be able to make your very own **419**!

Function optional arguments

You don't need to supply all arguments:

- ***number_format*(num)**

```
$number = 1234567.89 ;
echo "Here's a number: " ;
echo number_format($number) ;
```

```
Here's a number: 1,234,567
```

- ***number_format*(num,dec)**

```
$number = 1234567.89 ;
echo "Here's a number: " ;
echo number_format($number,2) ;
```

```
Here's a number: 1,234,567.89
```

- ***number_format*(num,dec,dot,mil)**

```
$number = 1234567.89 ;
echo "Here's a number: " ;
echo number_format($number,2,",",".") ;
```

```
Here's a number: 1.234.567,89
```

Agenda

- RECAP ('while' and 'for' loops)
- FUNCTIONS
- INCLUDING FILES
- ARRAYS
- ASSOCIATIVE ARRAYS ('foreach')

Including functions

- You can define functions in a file...:

*** my_function.php ***

```
<?php
function writeEntirePage($title, $body) {
    echo "<html>
            <head>
              <title>$title</title>
            </head>
            <body bgcolor='yellow'>
              $body
            </body>
          </html>" ;
} ?>    // Note: a string 'in quotes' may span multiple lines!
```

- ...and ***include*** them in your other PHP scripts:

```
<?php
    include("my_function.php") ;                    // include!
    writeEntirePage("Hello", "<h1>Hello World!</h1>") ;
?>
```

Another 'include' example

- You can define functions in a file...:

*** header_footer.php ***

```
<?php
function writeHeader() {
    echo "<html><head><title>Welcome to ITU</title></head>
          <body><h1>Welcome to the digital world!</h1>" ;
}

function writerFooter() {
    echo "<p>ITU: Rued Langgaards Vej 7; 2300 Copenhagen S</p>
          </body></html>";
} ?>
```

- ...and ***include*** them in your other PHP scripts:

```
<?php
    include("header_footer.php") ;
    writeHeader();
    echo "Blah Blah Blah..." ;
    writeFooter() ;
?>
```

Welcome to the digital world!

Blah Blah Blah...

ITU: Rued Langgaards Vej 7; 2300 Copenhagen S

Variable Visibility (aka Scope)

- Variables are visible after their definition:

```
<?php    $x = 42 ;
         /* ...whole bunch of code... */
         echo "Value of 'x' is: $x" ;    ?>
```

Value of 'x' is: 42

- Even var's included are visible after inclusion:

```
*** my_numbers.php ***
<?php    $x = 42 ;
         $pi = 3.1415926 ;    ?>
```

```
<?php    include "my_numbers.php" ;
         echo "x = $x, &pi; = $pi" ;    ?>
```

x = 42, π = 3.1415926

- (...as if the included text had been pasted in)

Functions & Variable Visibility

- A function has its **own private variables** (that it shares with no one):

```
$x = 42 ;

function myFunction() {
   $y = 87 ; // local variable (only visible in 'myFunction')!
   echo "x = ($x), y = ($y)" ; // x not visible here!
}

myFunction() ; // invoke function!

echo "x = ($x), y = ($y)" ; // y not visible here!
```

x = (), y = (87)

x = (42), y = ()

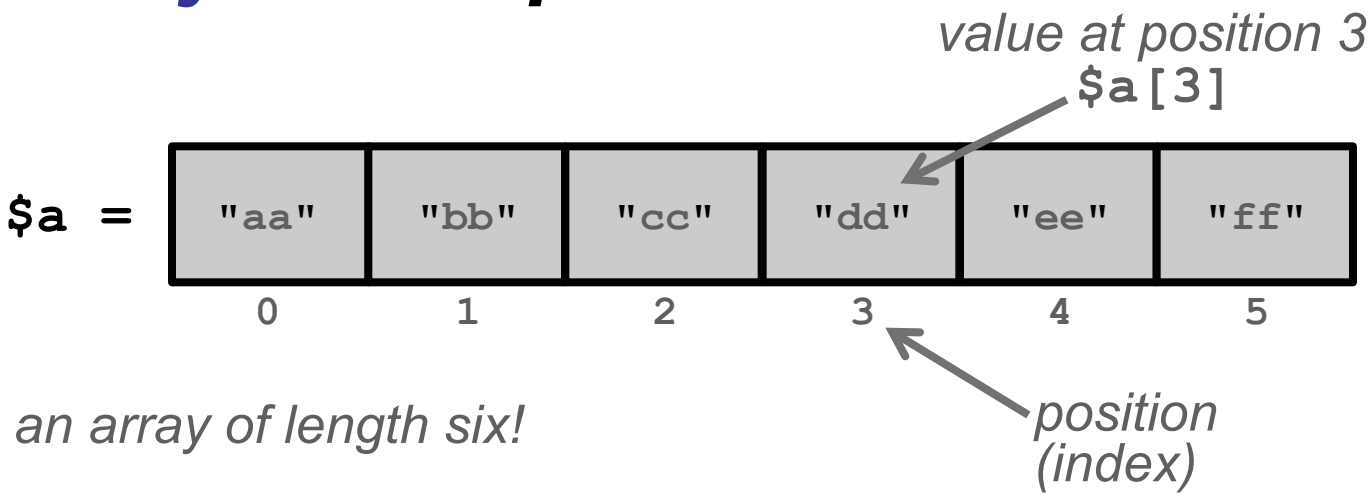
Agenda

- RECAP ('while' and 'for' loops)
- FUNCTIONS
- INCLUDING FILES
- ARRAYS
- ASSOCIATIVE ARRAYS ('foreach')

Arrays

- So far, you have seen (these kinds of values):
  - ***Numbers:*** `0`, `7`, `-3`, `2.34`, `3.1415926`
  - ***Strings:*** `"Barack Obama"`, `"Blah *!%&"`
  - ***Booleans:*** `true`, `false`

- An ***array*** is a ***sequence of values:***



'$a' is an array of length six!

Array Example

- Here's how to create this array in PHP:

```
$a[0] = "aa" ;
$a[1] = "bb" ;
$a[2] = "cc" ;
$a[3] = "dd" ;
$a[4] = "ee" ;
$a[5] = "ff" ;

echo "At index '3', we have: " ;
echo $a[3] ;
```

$a =

At index '3',we have: dd

- Alternatively, ***array()*** constructs an array (starting at index zero):

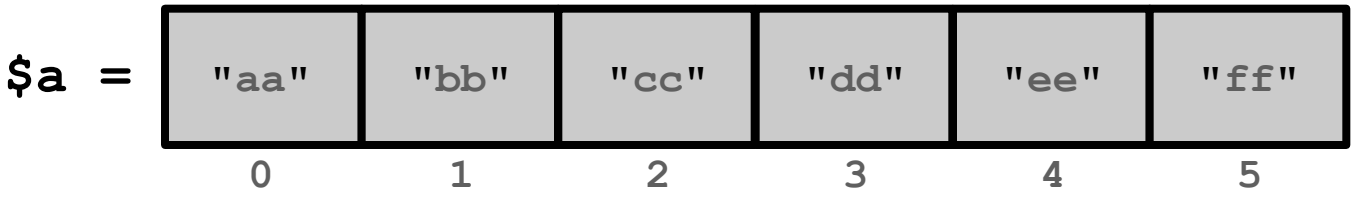
```
$a = array( "aa", "bb", "cc", "dd", "ee", "ff") ;

echo "At index '3', we have: " ;
echo $a[3] ;
```

At index '3', we have: dd

Traversing arrays with 'for' :-)

- Given the array from before:

```
$a = array( "aa", "bb", "cc", "dd", "ee", "ff") ;
```



- We can easily traverse an array using '**for**':

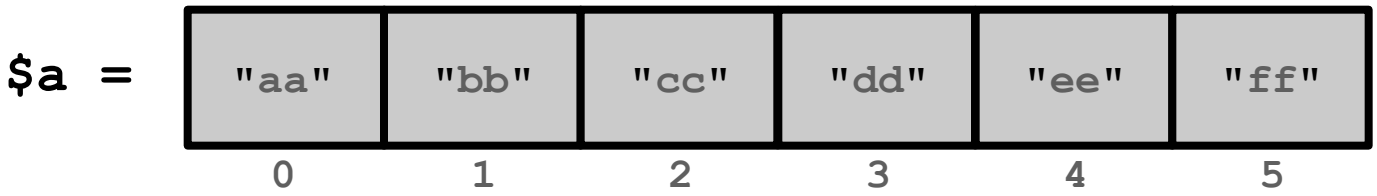
```
for ( $i = 0 ; $i < sizeof($a) ; $i++ ) {
    echo "<li> At index '$i', we have: $a[$i] </li>" ;
}
```

***sizeof*($a)**
returns the number of elements in array **$a**

- At index '0', we have: aa
- At index '1', we have: bb
- At index '2', we have: cc
- At index '3', we have: dd
- At index '4', we have: ee
- At index '5', we have: ff

Arrays

- You can think of an array as a ***sequence of boxes:***



  - Below each box you put a sticker with the ***index*** (the number of the box in the sequence):
    - The index starts at zero
    - The last index is (number-of-elements-in-array minus 1)
  - Each box will have a ***value*** inside which is accessed by $name-of-the-array[index] e.g.: **`$a[3]`**

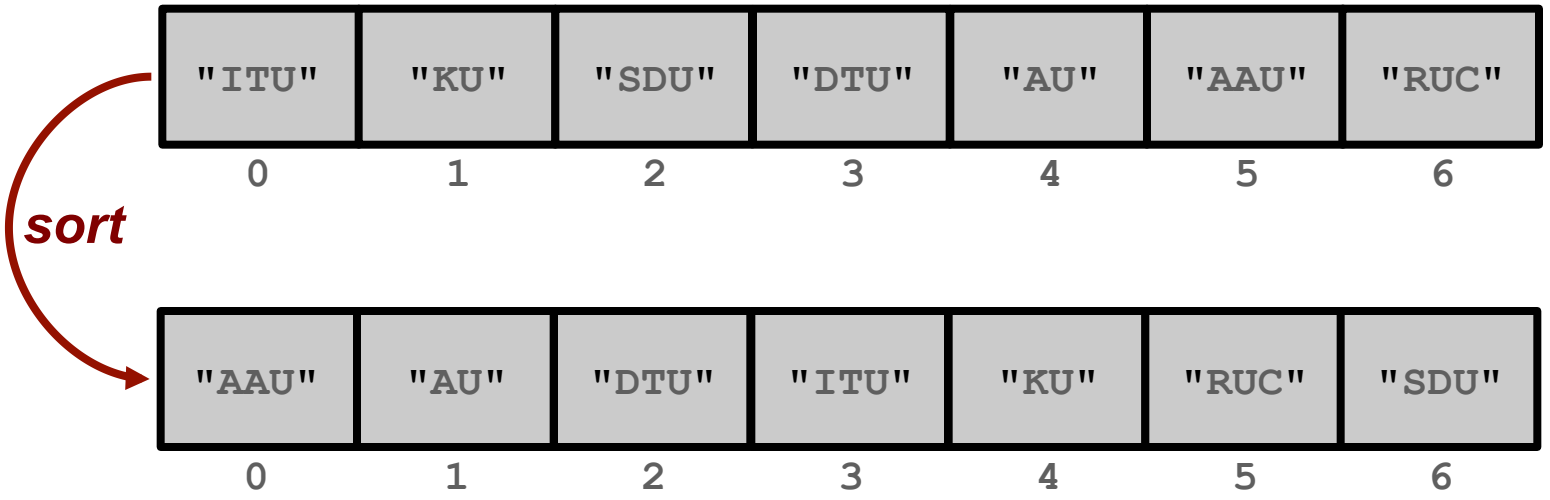
Tip: *var_dump*()

- The built-in function var_dump(x) will print out the value of x:

```
$a = array( "zero", "one", "two", "three", "four", "five") ;

var_dump($a) ;
```

```
array(6) {
  [0] => string(4) "zero"
  [1] => string(3) "one"
  [2] => string(3) "two"
  [3] => string(5) "three"
  [4] => string(4) "four"
  [5] => string(4) "five"
}
```

Sorting Arrays



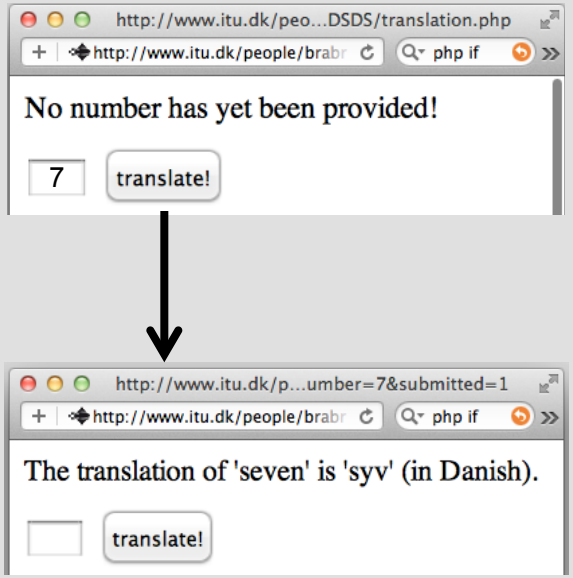
- We can easily ***sort*** arrays:

```
$unis = array("ITU", "KU", "SDU", "DTU", "AU", "AAU", "RUC");

sort($unis) ;  // sort universities (in alphabetical order)

for ( $i = 0 ; $i < sizeof($unis) ; $i++ ) {
    echo "<li> $unis[$i] </li>" ;
}
```

- AAU
- AU
- DTU
- ITU
- KU
- RUC
- SDU

Example: Number Translation

```
<html><body>
<?php
  $english = array("zero", "one", "two", "three", "four",
                   "five", "six", "seven", "eight", "nine");
  $danish  = array("nul", "en", "to", "tre", "fire",
                   "fem", "seks", "syv", "otte", "ni");

  $number = $_REQUEST['number'] ;
  $index = $number + 1;

  echo "The translation of '$english[$index]'
        is '$danish[$index]' (in Danish)." ;

?>
<form action="">
  <input type="text" name="number"
         size="1" maxlength="1" />
  <input type="submit" value="translate!" />
</form></body></html>
```



Agenda

- RECAP ('while' and 'for' loops)
- FUNCTIONS
- INCLUDING FILES
- ARRAYS
- ASSOCIATIVE ARRAYS ('foreach')

Associative Arrays

- Alternatively can be constructed via the "***array()***" array constructor function:

```
<?php
   $capital['Denmark'] = 'Copenhagen';
   $capital['England'] = 'London';
   $capital['France'] = 'Paris';

   var_dump($capitals) ;
?>
```

```
array(3) {
   ["Denmark"] => string(10) "Copenhagen"
   ["England"] => string(6)  "London"
   ["France"]  => string(5)  "Paris"
}
```

Associative Arrays

- Alternatively can be constructed via the "***array()***" array constructor function:

```
<?php
   $capital['Denmark'] = 'Copenhagen';
   $capital['England'] = 'London';
   $capital['France'] = 'Paris';

   echo "The capital of Denmark is: " ;
   echo $capital['Denmark'] ;
?>
```

The capital of Denmark is: Copenhagen

The 'foreach' loop

- Alternatively can be constructed via the "*array()*" array constructor function:

```
<?php
   $capital['Denmark'] = 'Copenhagen';
   $capital['England'] = 'London';
   $capital['France'] = 'Paris';

   foreach ( $capital as $key => $value ) {
      echo "<li> The capital of $key is: $value </li>" ;
   }
?>
```

- The capital of Denmark is: Copenhagen
- The capital of England is: London
- The capital of France is: Paris

My Very Own SPAM Service !?!

```php
$emails = array("Anders" => "aaa@itu.dk",
                "Bjarke" => "bbb@itu.dk",
                ...,
                "Zidane" => "zzz@itu.dk") ;

$sender    = "Master Kofu George" ;
$email     = "kofu.george@yahoo.com" ;
$subject   = "Business Opportunity" ;
$header    = "From: " . $sender . " <" . $email . ">\r\n" ;

foreach ($emails as $name => $email) {
   $message = "Dear $name, I have a Business Opportunity..." ;
   $message = strtoupper($message) ;
   mail($email, $subject, $message, $header) ;
}

$count = sizeof($emails) ;
echo "SPAM completed: $count SPAM mails sent." ;
```

DON'T RUN THIS !

Any questions?