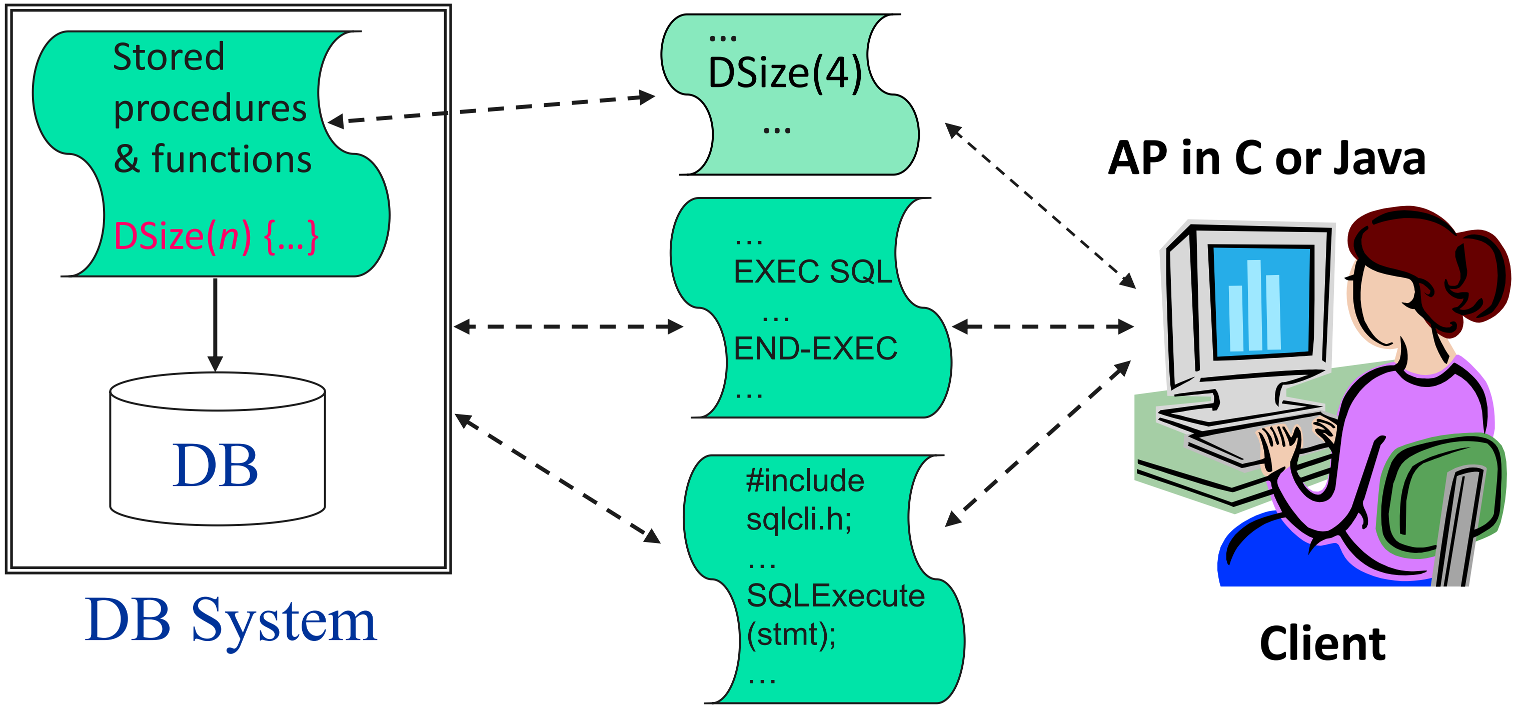
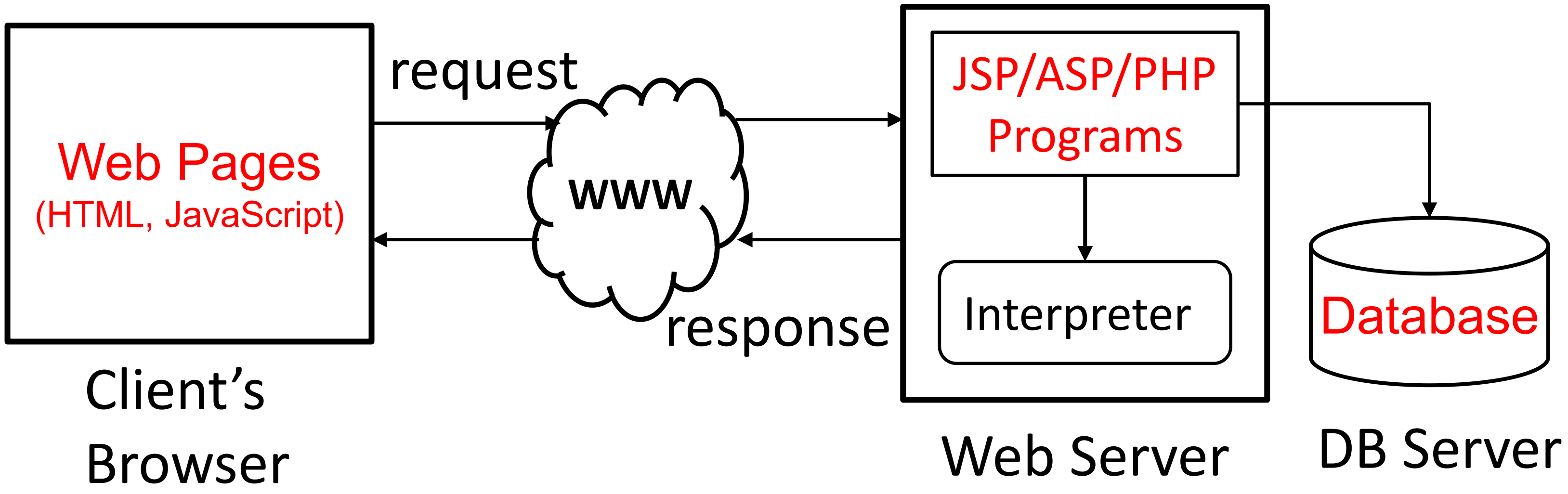
Database Programming with C and Java

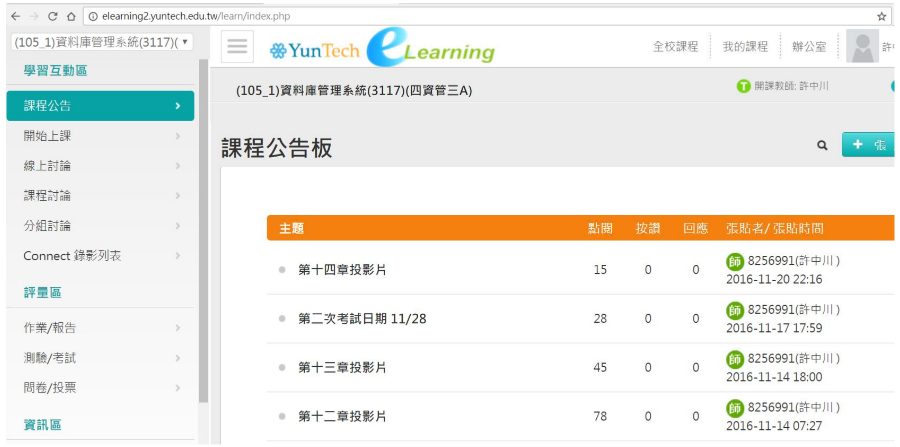
Ch. 12: Database programming
- embedded/dynamic SQL
- function call
- stored procedure and function



Web Database Programming



http://elearning2.yuntech.edu.tw/learn/index.php


elearning2.yuntech.edu.tw/learn/index.php

(105_1)資料庫管理系統(3117)(四資管三A)

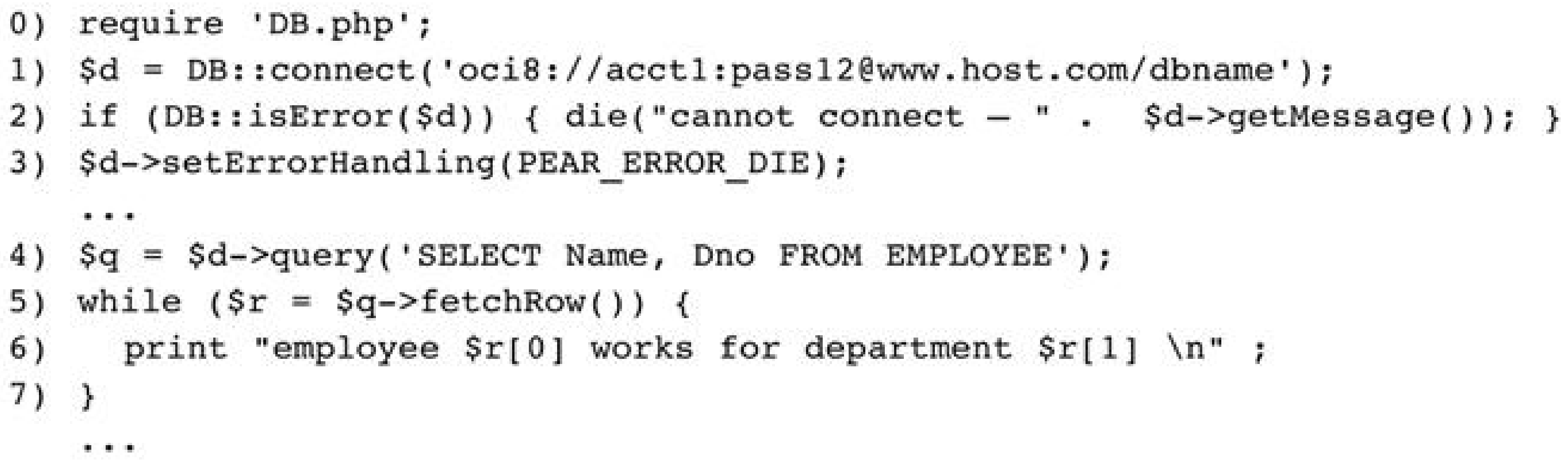
開課教師: 許中川

課程公告板

主題	點閱	按讚	回應	張貼者/ 張貼時間
第十四章投影片	15	0	0	8256991(許中川) 2016-11-20 22:16
第二次考試日期 11/28	28	0	0	8256991(許中川) 2016-11-17 17:59
第十三章投影片	45	0	0	8256991(許中川) 2016-11-14 18:00
第十二章投影片	78	0	0	8256991(許中川) 2016-11-14 07:27

Chapter 13

Web Database Programming using PHP



```
require 'DB.php';
$d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage()); }
$d->setErrorHandling(PEAR_ERROR_DIE);
...
$q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
while ($r = $q->fetchRow()) {
  print "employee $r[0] works for department $r[1] \n" ;
}
...
```

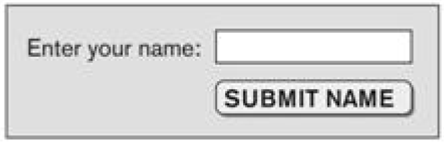
Outline

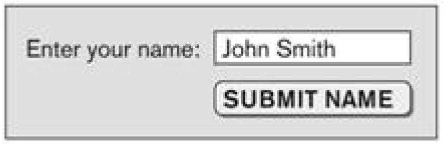
- Overview
- PHP
- Example of PHP
- Basic features of PHP
- Overview of PHP Database programming

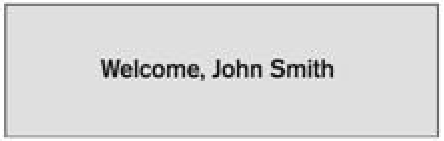
- PHP tutorials:
  http://www.java2s.com/Tutorials/PHP/index.htm
- More sample codes: http://www.java2s.com/

Overview

- Hypertext documents
  - Common method of specifying contents
  - Various languages
    - ✓ HTML (HyperText Markup Language)
      - ➢ Used for generating static web pages
    - ✓ XML (eXtensible Markup Language)
      - ➢ Standard for exchanging data over the web
    - ✓ PHP (PHP Hypertext Preprocessor {recursive acronym})
      - ➢ Dynamic web pages

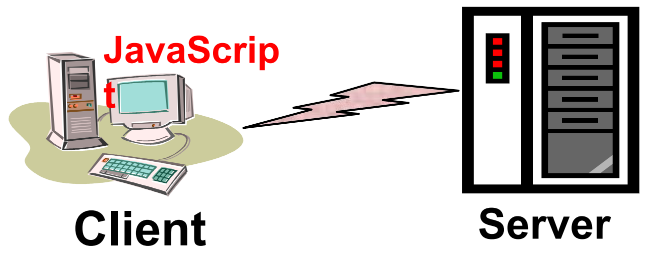
(b)

Enter your name:

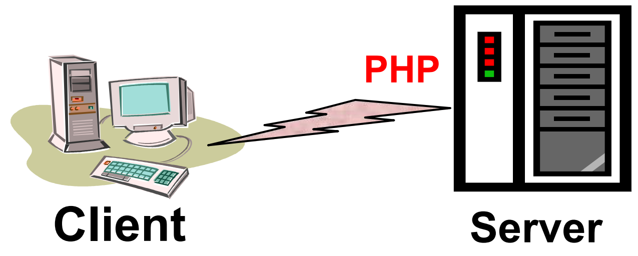
(c)

Enter your name: John Smith

(d)

Welcome, John Smith

PHP

- Open source
- General purpose scripting language
- Interpreter engine in C
  - Can be used on nearly all computer types
- Particularly suited for manipulation of text pages
- Manipulates (dynamic html) at the Web *server*
  - Conversely, JavaScript is downloaded and executed on the client
- Has libraries of functions for accessing databases





A Simple PHP Example

- Type the url www.myserver.com/example/greeting.php, the PHP interpreter will start interpreting produce form in (b)

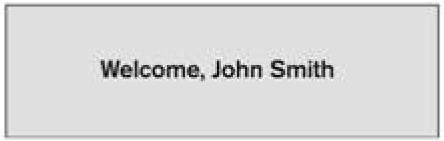
```
//Program Segment P1:
<?php
// Printing a welcome message if the user submitted their name
// through the HTML form
if ($_POST['user_name']) {
  print("Welcome, ") ;
  print($_POST['user_name']);
}
else {
  // Printing the form to enter the user name since no name has
  // been entered yet
  print <<<_HTML_
  <FORM method="post" action="$_SERVER['PHP_SELF']">
  Enter your name: <input type="text" name="user_name">
  <BR/>
  <INPUT type="submit" value="SUBMIT NAME">
  </FORM>
  _HTML_;
}
?>
```

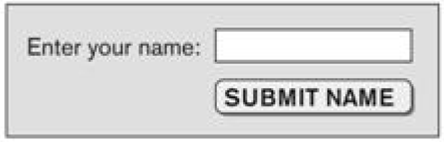
有值顯示welcome

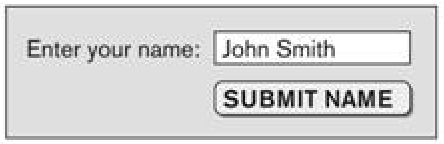
值丟表單

$_POST: an associative array of predefined variables passed to the current script via the HTTP POST method.

post是內建變數
把資料存在變數內


Welcome, John Smith

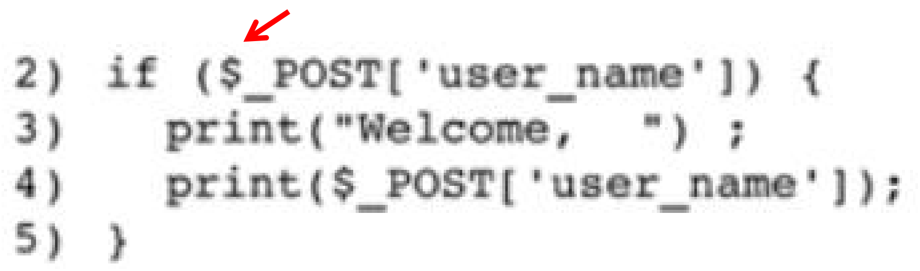

Enter your name:


Enter your name: John Smith

Overview of basic features of PHP

- PHP variables, data types, and programming constructs
  - Variable names start with $ and can include characters, letters, numbers, and _.
    - ✓ No other special characters are permitted
    - ✓ Are case sensitive
    - ✓ Can't start with a number
  - Variables are not typed
    - ✓ Values assigned to variables determine their type
    - ✓ Assignments can change the type
  - Variable assignments are made by =

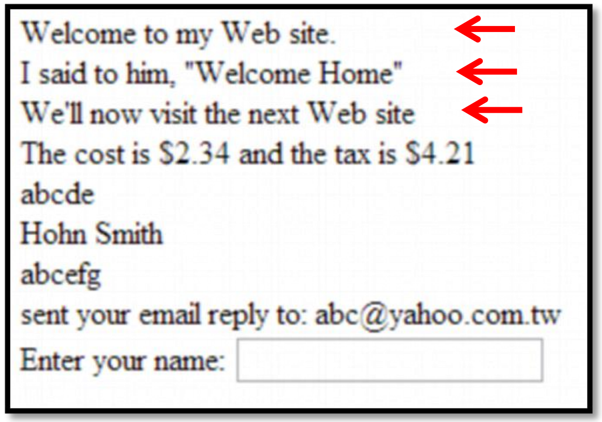
Php有分大小寫 不可開頭為數字



```
if ($_POST['user_name']) {
   print("Welcome,  ") ;
   print($_POST['user_name']);
}
```

Main Ways to Express Strings

- Single-quoted strings (lines 0, 1, 2)
  - ✓ \' represents a quote in a string 字串要用單引號 要反斜線 \’
- Double-quoted strings (line 7)
  - ✓ Variable names can be interpolated
- Here documents (line 8-11)
  - ✓ Enclose a part of a document between <<<DOCNAME and end it with a single line containing the document name DOCNAME
- Single and double quotes (lines 0, 7)
  - ✓ The quotes should be straight quotes (') not (‘) or (’)



```
Welcome to my Web site.
I said to him, "Welcome Home"
We'll now visit the next Web site
The cost is $2.34 and the tax is $4.21
abcde
Hohn Smith
abcefg
sent your email reply to: abc@yahoo.com.tw
Enter your name: 
```

```
 0) print 'Welcome to my Web site.';
 1) print 'I said to him, "Welcome Home"';
 2) print 'We\'ll now visit the next Web site';
 3) printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
 4) print strtolower('AbCdE');
 5) print ucwords(strtolower('JOHN smith'));
 6) print 'abc' . 'efg'
 7) print "send your email reply to: $email_address"
 8) print <<<FORM_HTML
 9) <FORM method="post" action="$_SERVER['PHP_SELF']">
10) Enter your name: <input type="text" name="user_name">
11) FORM_HTML
```

Figure 26.4
Illustrating basic PHP string and text values.

String operations

- String operations
  - Line 4: **strtolower()** 字串改為小寫**(lower case)**
  - Line 5: **ucwords() 字首改為大寫(**uppercases)
  - Line 6: **(.)** is **concatenate**

```
 0) print 'Welcome to my Web site.';
 1) print 'I said to him, "Welcome Home"';
 2) print 'We\'ll now visit the next Web site';
 3) printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
 4) print strtolower('AbCdE'); ←
 5) print ucwords(strtolower('JOHN smith')); ←
 6) print 'abc' . 'efg' ←
 7) print "send your email reply to: $email_address"
 8) print <<<FORM_HTML
 9) <FORM method="post" action="$_SERVER['PHP_SELF']">
10) Enter your name: <input type="text" name="user_name">
11) FORM_HTML
```

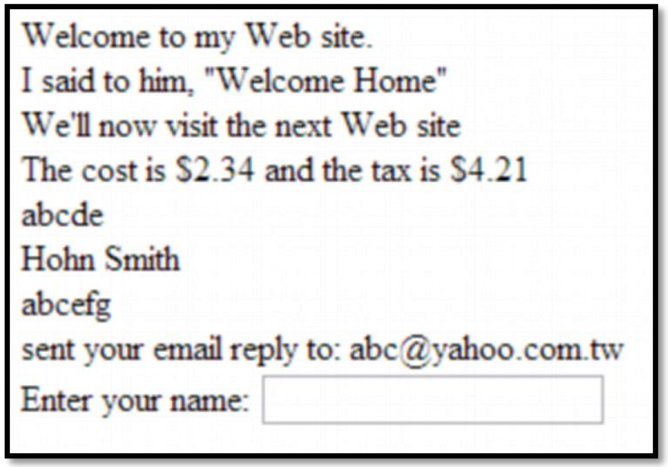
Figure 26.4
Illustrating basic PHP string and text values.

Numeric data types

- **printf()** follows C rules (See Line 3)

```
 0) print 'Welcome to my Web site.';
 1) print 'I said to him, "Welcome Home"';
 2) print 'We\'ll now visit the next Web site';
 3) printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
 4) print strtolower('AbCdE');
 5) print ucwords(strtolower('JOHN smith'));
 6) print 'abc' . 'efg'
 7) print "send your email reply to: $email_address"
 8) print <<<FORM_HTML
 9) <FORM method="post" action="$_SERVER['PHP_SELF']">
10) Enter your name: <input type="text" name="user_name">
11) FORM_HTML
```

Figure 26.4
Illustrating basic PHP string and text values.



```
Welcome to my Web site.
I said to him, "Welcome Home"
We'll now visit the next Web site
The cost is $2.34 and the tax is $4.21
abcde
Hohn Smith
abcefg
sent your email reply to: abc@yahoo.com.tw
Enter your name:
```

Other programming constructs

- Other programming constructs similar to C language constructs
  - **while-loops**
  - **for-loops**
  - **if-statements**

```
while ($r = $q->fetchRow()) {
   print "employee $r[0] \n" ;
}
```

```
for ($i = 0, $num = count($courses); i < $num; $i++) {
   print '<TR bgcolor="' . $alt_row_color[$i % 2] . '">';
   print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
}
```

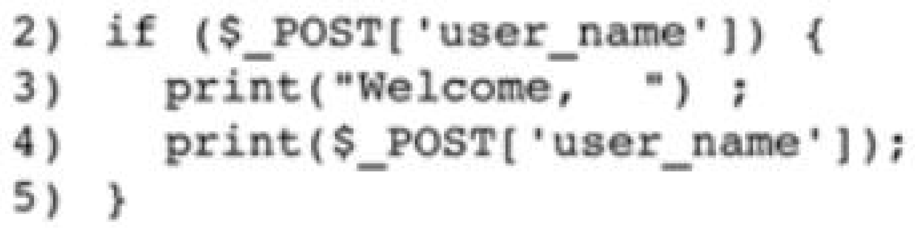
```
if (array_key_exists($course, $teaching_assignments)) {
   $instructor = $teaching_assignments[$course];
   RETURN "$instructor is teaching $course";
}
else {
   RETURN "there is no $course course";
}
```

Boolean logic

- True/false is equivalent to non-zero/zero
- Comparison operators:

  ==, !=, >, >=, <, <=

```
if ($action == "show_version") {
    echo "The version is 1.23";
}
```



```
if ($_POST['user_name']) {
   print("Welcome,  ") ;
   print($_POST['user_name']);
}
```

PHP Arrays

- Allow a list of elements
- Can be **1-dimensional** or **multi-dimensional**
- Can be **numeric** or **associative**
  - Numeric array is based on a numeric index, starting from 0
  - Associative array is based on a **key => value** relationship

```
$courses = array('Database', 'OS', 'Graphics', 'Data Mining');

$teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                  'Graphics' => 'Kam');
```

```
$myCourse = $courses[0];
$myTeacher = $teaching[‘Database’];
```

```
A[0] = 5;
S[“John”] = “555-1122”;
```

PHP Arrays

- Line 0: **$teaching** is an associative array
  - Line 1 shows how the array can be updated/accessed
- Line 5: **$courses** is a numeric array
  - No key is provided => numeric array
  - Line 9 shows how the array can be accessed

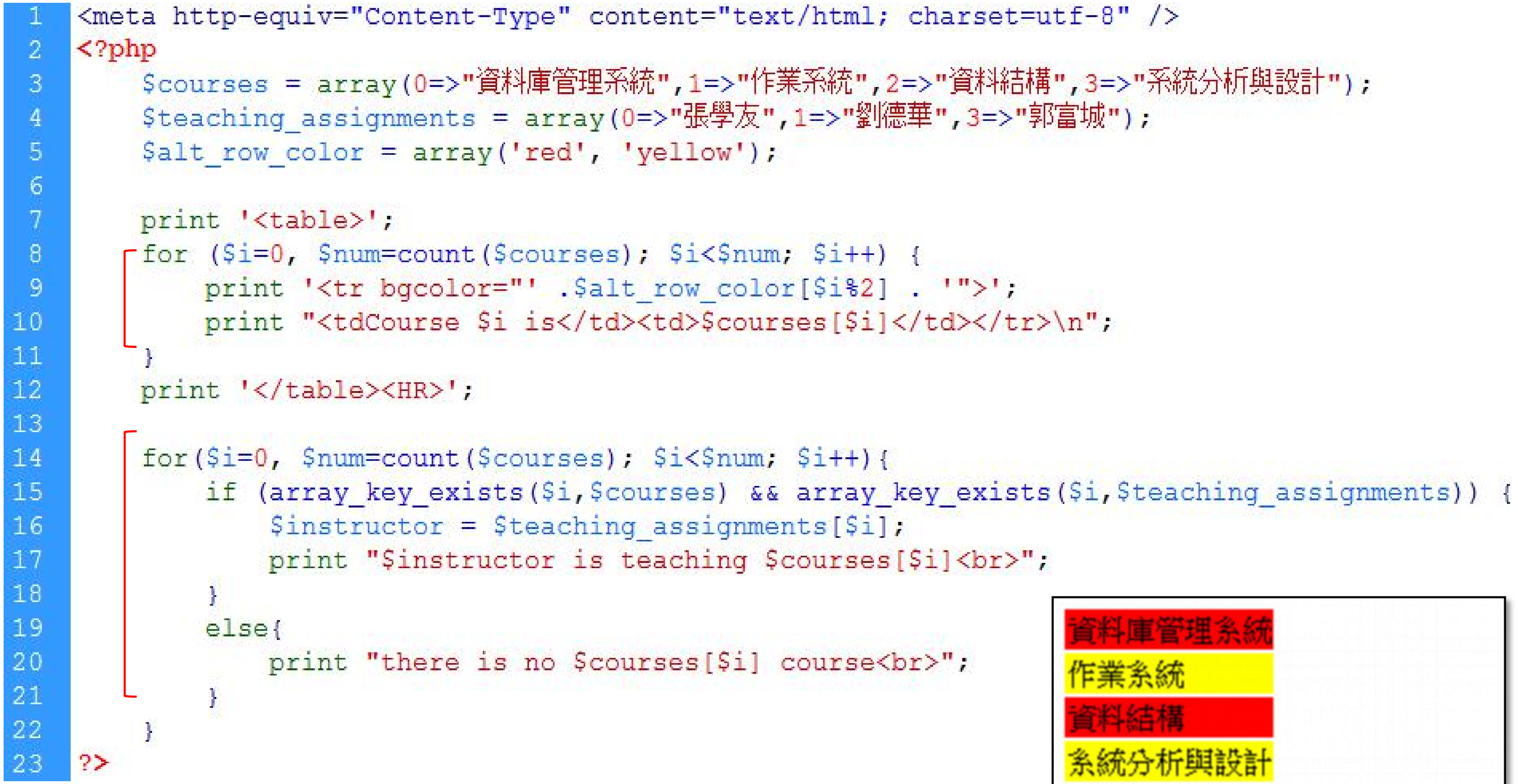
```
 0)  $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                       'Graphics' => 'Kam');
 1)  $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
 2)  krsort($teaching);
 3)  foreach ($teaching as $key => $value) {
 4)    print " $key : $value\n";}
 5)  $courses = array('Database', 'OS', 'Graphics', 'Data Mining');
 6)  $alt_row_color = array('blue', 'yellow');
 7)  for ($i = 0, $num = count($courses); i < $num; $i++) {
 8)    print '<TR bgcolor="' . $alt_row_color[$i % 2] . '">';
 9)    print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
10)  }
```

PHP Arrays and Looping

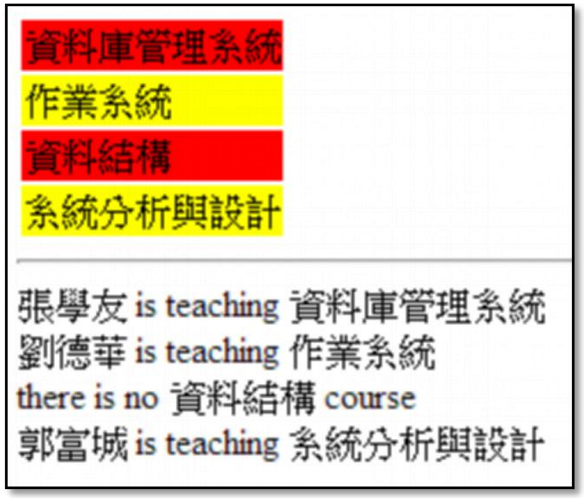
- There are several ways of **looping** through arrays
  - Line 3 and 4 show "**for each**" construct for looping through each and every element in the array
  - Line 7 and 10 show a traditional "**for loop**" construct for iterating through an array

```
 0)  $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                       'Graphics' => 'Kam');
 1)  $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
 2)  krsort($teaching);
 3)  foreach ($teaching as $key => $value) {
 4)    print " $key : $value\n";}
 5)  $courses = array('Database', 'OS', 'Graphics', 'Data Mining');
 6)  $alt_row_color = array('blue', 'yellow');
 7)  for ($i = 0, $num = count($courses); i < $num; $i++) {
 8)    print '<TR bgcolor="' . $alt_row_color[$i % 2] . '">';
 9)    print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
10)  }
```

PHP Arrays and Looping



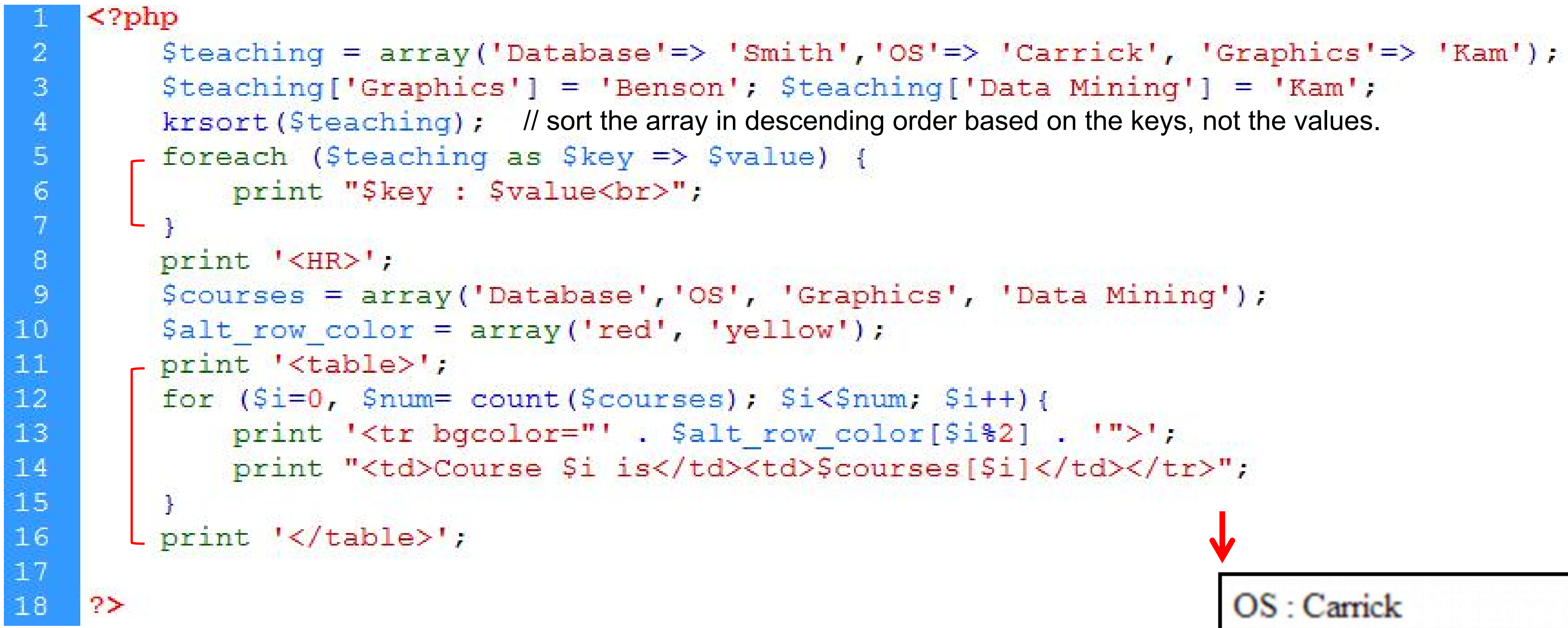
```php
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php
    $courses = array(0=>"資料庫管理系統",1=>"作業系統",2=>"資料結構",3=>"系統分析與設計");
    $teaching_assignments = array(0=>"張學友",1=>"劉德華",3=>"郭富城");
    $alt_row_color = array('red', 'yellow');

    print '<table>';
    for ($i=0, $num=count($courses); $i<$num; $i++) {
        print '<tr bgcolor="' .$alt_row_color[$i%2] . '">';
        print "<tdCourse $i is</td><td>$courses[$i]</td></tr>\n";
    }
    print '</table><HR>';

    for($i=0, $num=count($courses); $i<$num; $i++){
        if (array_key_exists($i,$courses) && array_key_exists($i,$teaching_assignments)) {
            $instructor = $teaching_assignments[$i];
            print "$instructor is teaching $courses[$i]<br>";
        }
        else{
            print "there is no $courses[$i] course<br>";
        }
    }
?>
```



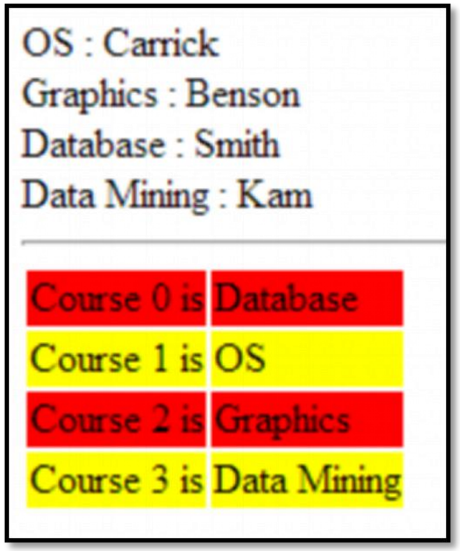
```
資料庫管理系統
作業系統
資料結構
系統分析與設計
────────────────
張學友 is teaching 資料庫管理系統
劉德華 is teaching 作業系統
there is no 資料結構 course
郭富城 is teaching 系統分析與設計
```

Web site: http://140.125.84.81:81/db_example/example3.php

PHP Array Sorting



```php
<?php
    $teaching = array('Database'=> 'Smith','OS'=> 'Carrick', 'Graphics'=> 'Kam');
    $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
    krsort($teaching);    // sort the array in descending order based on the keys, not the values.
    foreach ($teaching as $key => $value) {
        print "$key : $value<br>";
    }
    print '<HR>';
    $courses = array('Database','OS', 'Graphics', 'Data Mining');
    $alt_row_color = array('red', 'yellow');
    print '<table>';
    for ($i=0, $num= count($courses); $i<$num; $i++){
        print '<tr bgcolor="' . $alt_row_color[$i%2] . '">';
        print "<td>Course $i is</td><td>$courses[$i]</td></tr>";
    }
    print '</table>';

?>
```



```
OS : Carrick
Graphics : Benson
Database : Smith
Data Mining : Kam
--------------------------------
Course 0 is  Database
Course 1 is  OS
Course 2 is  Graphics
Course 3 is  Data Mining
```

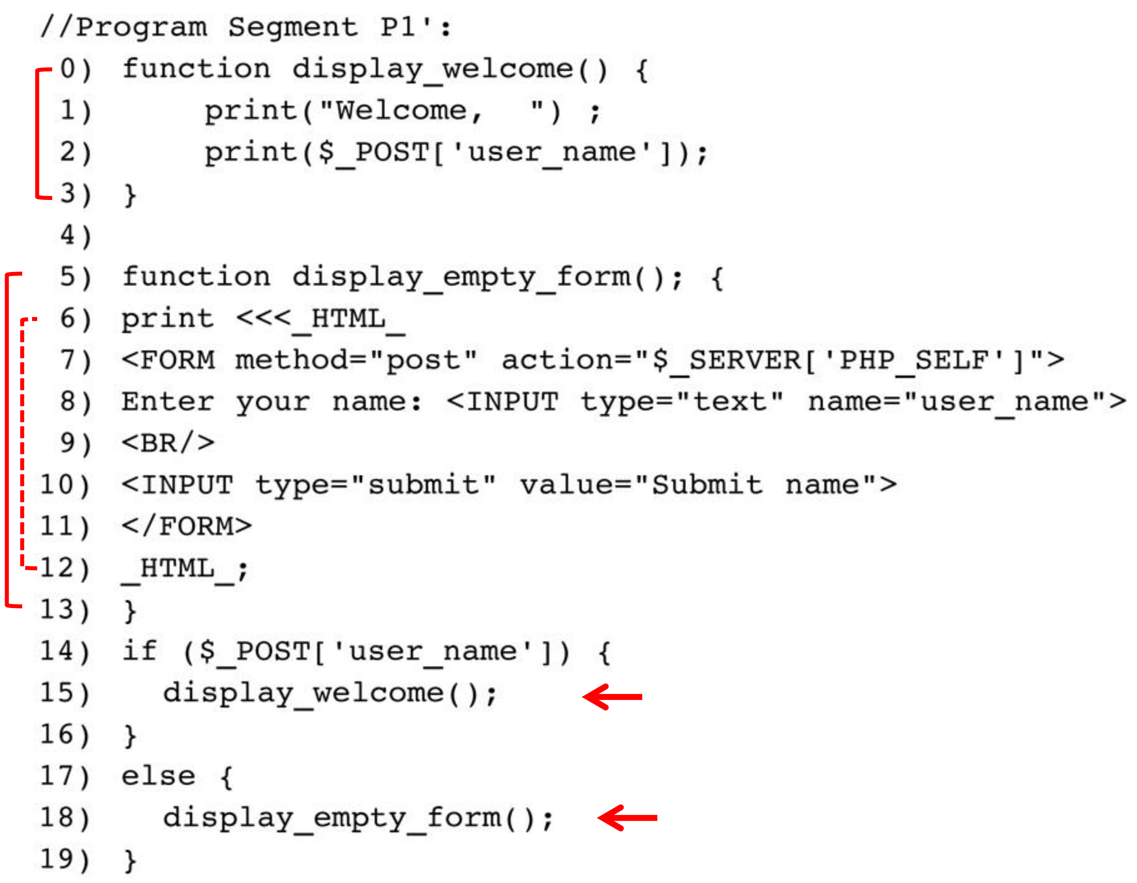
Sorting:

ksort($teaching);   // sort in ascending key order
asort($teaching);   // sort an associative array in ascending value order
sort($teaching);   // sort in ascending value order and keys will be replaced by integers.
krsort(), arsort(), rsort(); // sort in reverse (descending) order

Web site: http://140.125.84.81:81/db_example/example4.php

PHP Functions

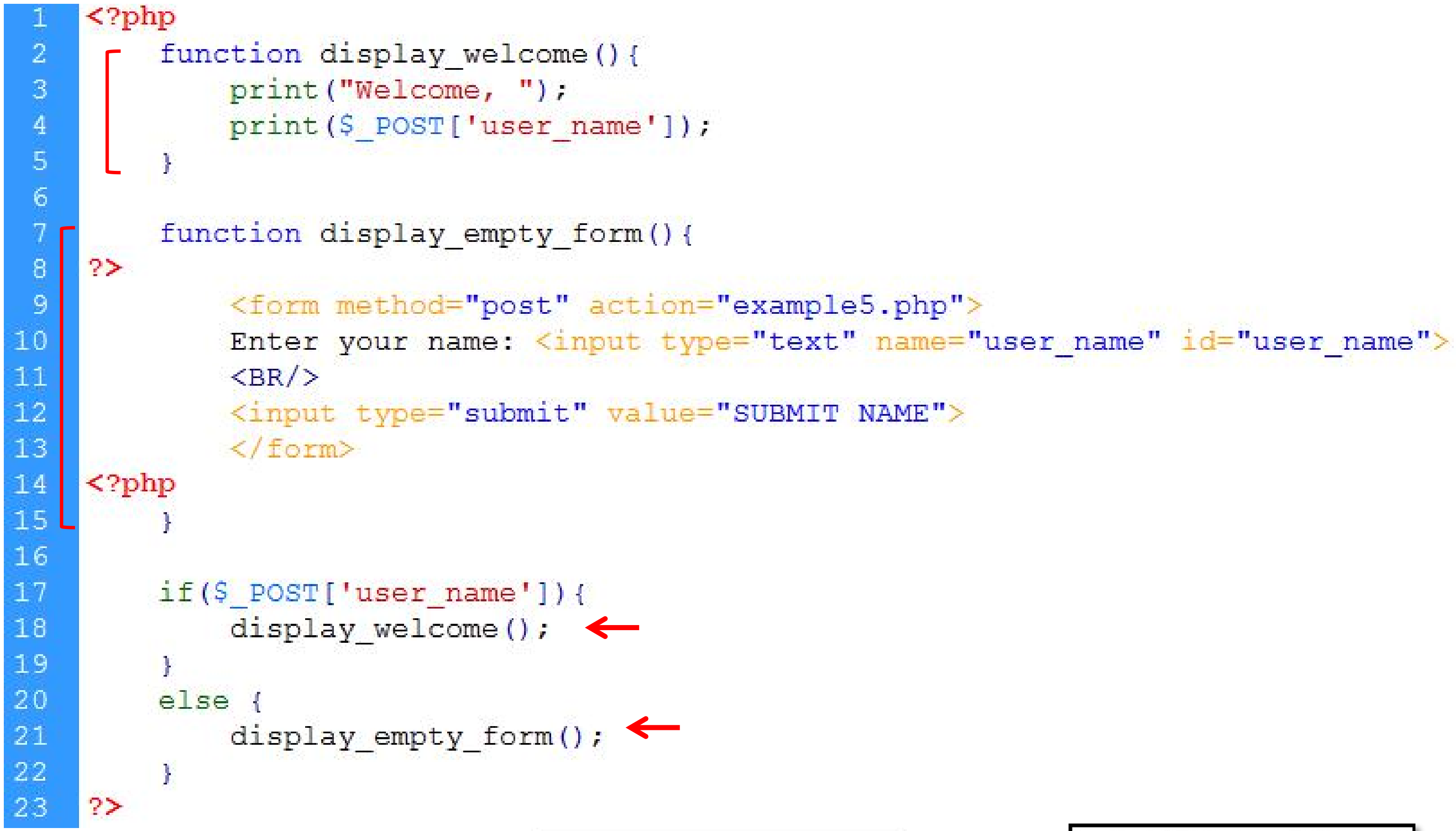
- Code segment P1' has two functions
  - **display_welcome()**
  - **display_empty_form()**
- Line 14-19 show how these functions can be called



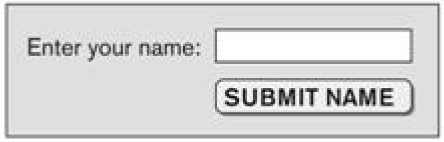
```
//Program Segment P1':
 0)  function display_welcome() {
 1)      print("Welcome,  ") ;
 2)      print($_POST['user_name']);
 3)  }
 4)
 5)  function display_empty_form(); {
 6)  print <<<_HTML_
 7)  <FORM method="post" action="$_SERVER['PHP_SELF']">
 8)  Enter your name: <INPUT type="text" name="user_name">
 9)  <BR/>
10)  <INPUT type="submit" value="Submit name">
11)  </FORM>
12)  _HTML_;
13)  }
14)  if ($_POST['user_name']) {
15)    display_welcome();
16)  }
17)  else {
18)    display_empty_form();
19)  }
```

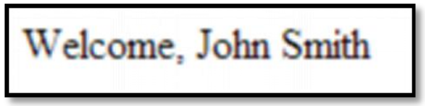
Figure 26.6
Rewriting program segment P1 as P1′ using functions.

PHP Functions



```php
<?php
    function display_welcome(){
        print("Welcome, ");
        print($_POST['user_name']);
    }

    function display_empty_form(){
?>
        <form method="post" action="example5.php">
        Enter your name: <input type="text" name="user_name" id="user_name">
        <BR/>
        <input type="submit" value="SUBMIT NAME">
        </form>
<?php
    }

    if($_POST['user_name']){
        display_welcome();
    }
    else {
        display_empty_form();
    }
?>
```





Welcome, John Smith

Web site: http://140.125.84.81:81/db_example/example5.php

PHP Observations in Function

- Built-in PHP function array_key_exists($k,$a) returns true if the value in $k as a key is in the associative array $a
- Function arguments are passed by value
- Return values are placed after the RETURN keyword
- Scope rules apply as with other programming languages

```
function course_instructor ($course, $teaching_assignments) {
  if (array_key_exists($course, $teaching_assignments)) {  ←
    $instructor = $teaching_assignments[$course];
    RETURN "$instructor is teaching $course"; ←
  }
  else {
    RETURN "there is no $course course";
  }
}
```

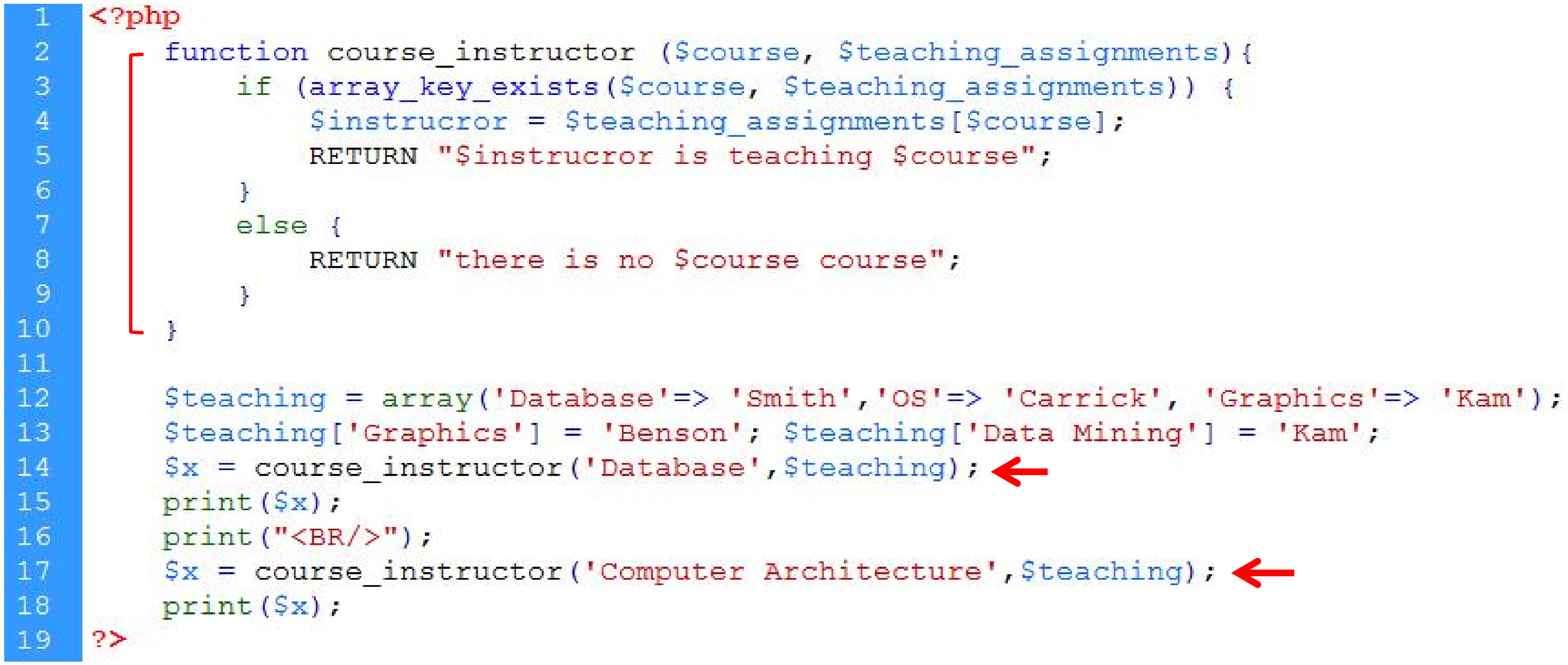
PHP Functions-Example

- The code segment has function
  - **course_instructor($course, $teaching_assignments)**
    - ✓ $course: holding the course name
    - ✓ $teaching_assignments: holding the teacher associated with the course

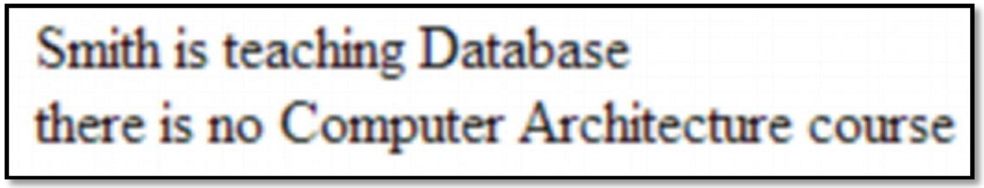
```
function course_instructor ($course, $teaching_assignments) {
  if (array_key_exists($course, $teaching_assignments)) {
    $instructor = $teaching_assignments[$course];
    RETURN "$instructor is teaching $course";
  }
  else {
    RETURN "there is no $course course";
  }
}
$teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                  'Graphics' => 'Kam');
$teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
$x = course_instructor('Database', $teaching);  ← // Call function
print($x);
$x = course_instructor('Computer Architecture', $teaching);  ←
print($x);
```

Figure 26.7
Illustrating a function with arguments and return value.

PHP Functions-Example



```php
<?php
    function course_instructor ($course, $teaching_assignments){
        if (array_key_exists($course, $teaching_assignments)) {
            $instrucror = $teaching_assignments[$course];
            RETURN "$instrucror is teaching $course";
        }
        else {
            RETURN "there is no $course course";
        }
    }

    $teaching = array('Database'=> 'Smith','OS'=> 'Carrick', 'Graphics'=> 'Kam');
    $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
    $x = course_instructor('Database',$teaching);
    print($x);
    print("<BR/>");
    $x = course_instructor('Computer Architecture',$teaching);
    print($x);
?>
```

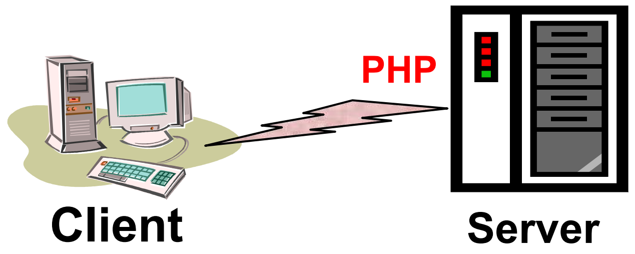


```
Smith is teaching Database
there is no Computer Architecture course
```

Web site: http://140.125.84.81:81/db_example/example6.php

PHP Server Variables and Forms

- There are a number of built-in entries in PHP function. Some examples are:
  - $_SERVER['SERVER_NAME']
    - ✓ This provides the **Website name of the server** computer where PHP interpreter is running
  - $_SERVER['REMOTE_ADDRESS']
    - ✓ **IP address of client user computer** that is accessing the server
  - $_SERVER['REMOTE_HOST']
    - ✓ **Website name of the client** user computer



```
<?php
echo $_SERVER['SERVER_NAME'];
?>
```

```
www.example.com
```
⟵ output

PHP Server Variables and Forms

- Examples contd.
  - $_SERVER['PATH_INFO']
    - ✓ Contains any client-provided pathname information **trailing the actual script filename** but **preceding the query string**, if available.
  - $_SERVER['QUERY_STRING']
    - ✓ The string that holds the parameters in the URL **after ?**.
  - $_SERVER['DOCUMENT_ROOT']
    - ✓ **The root directory** that holds the files on the Web server

http://www.example.com/php/path_info.php/some/stuff?foo=bar

Connecting to the database

- Must load PEAR DB library module DB.php
- DB library functions are called using

  DB::<function_name>

  For example:

  ```
  DB::connect(…);
  DB::isError(…);
  ```

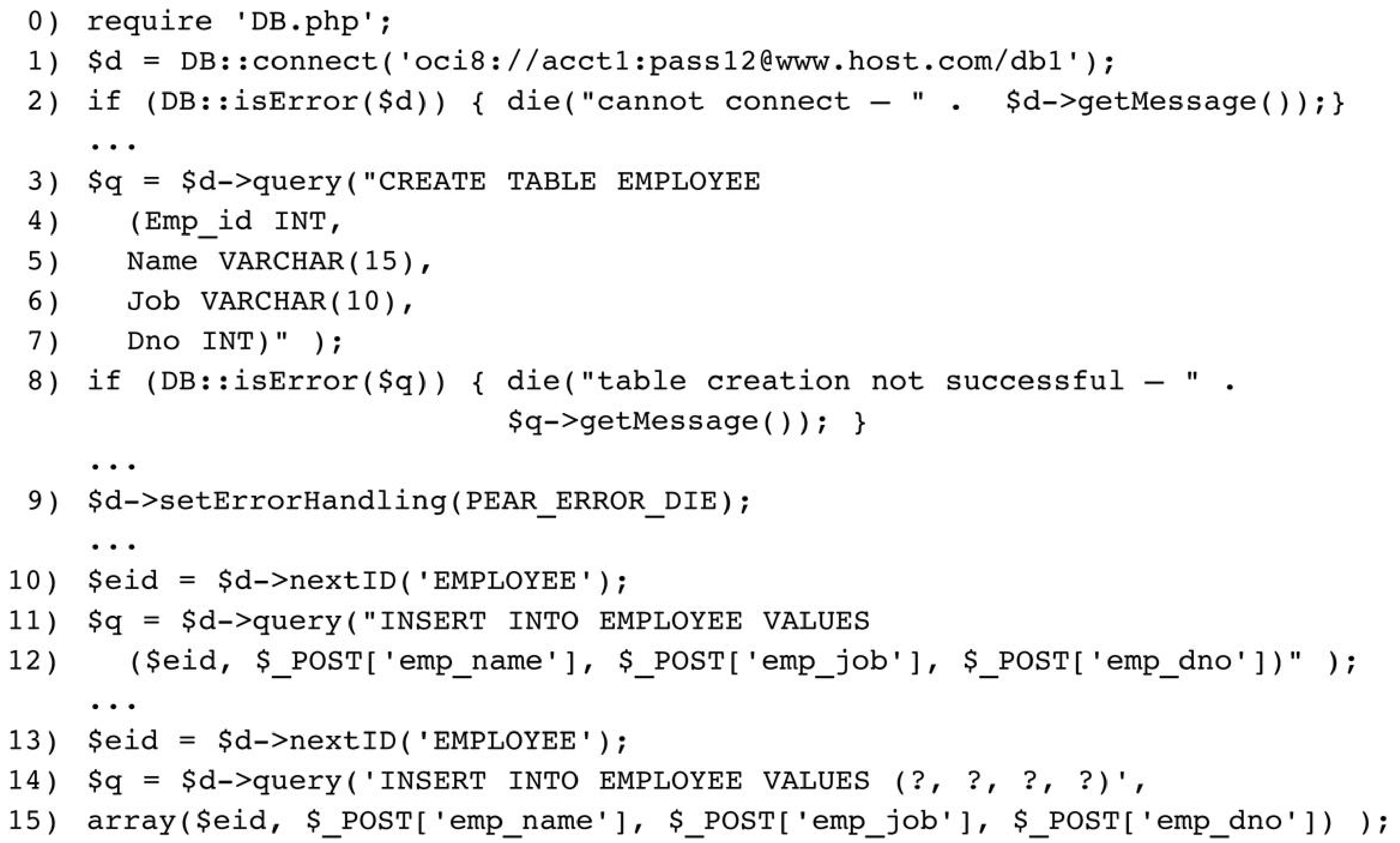
- The format for the connect string is:
  - **<DBMS>://<userid>:<password>@<DBserver>**

    For example:

    ```
    $d = DB::connect('oci8://ac1:pass12@www.abc.com/db1');
    ```

Example of PHP Database Programming

Figure 26.8
Connecting to a database, creating a table, and inserting a record.



```
require 'DB.php';
$d = DB::connect('oci8://acct1:pass12@www.host.com/db1');
if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage());}
...
$q = $d->query("CREATE TABLE EMPLOYEE
  (Emp_id INT,
  Name VARCHAR(15),
  Job VARCHAR(10),
  Dno INT)" );
if (DB::isError($q)) { die("table creation not successful – " .
                        $q->getMessage()); }
...
$d->setErrorHandling(PEAR_ERROR_DIE);
...
$eid = $d->nextID('EMPLOYEE');
$q = $d->query("INSERT INTO EMPLOYEE VALUES
  ($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'])" );
...
$eid = $d->nextID('EMPLOYEE');
$q = $d->query('INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)',
array($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno']) );
```

Overview of PHP Database Programming

- Examples of DB connections
  - MySQL: mysql
  - Oracle: oci8 (for versions 7, 8, 9)
  - SQLite: sqlite
  - MS SQL Server: mssql
  - Mini SQL: msql
  - Informix: ifx
  - Sybase: sybase
  - Any ODBC compliant DB: odbc
  - Others…

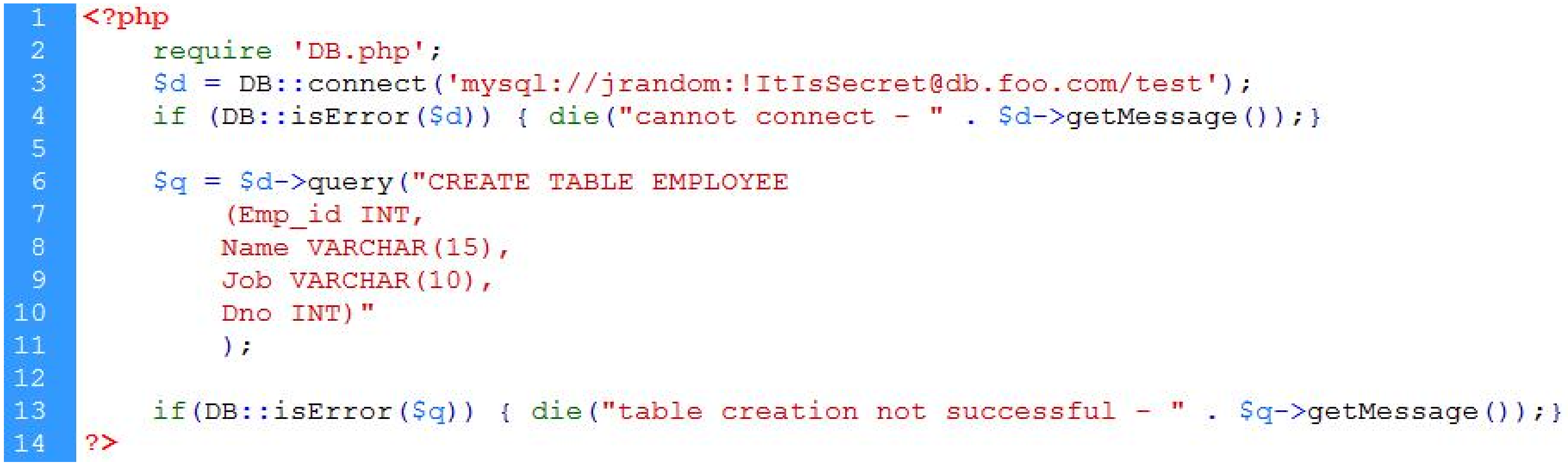
```
$d =
DB::connect(‘mysql://ac1:pass12@www.abc.com/db1')
```

Connect to DB and Create Table

- Line 1 connects; Line 2 tests the connection; Line 3-8 creates a table; Line 9 sets error handling

```
 0) require 'DB.php';
 1) $d = DB::connect('oci8://acct1:pass12@www.host.com/db1'); ←
 2) if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage());} ←
    ...                                                   //die: terminate the program
 3) $q = $d->query("CREATE TABLE EMPLOYEE ←
 4)   (Emp_id INT,
 5)   Name VARCHAR(15),                            //create table EMPLOYEE
 6)   Job VARCHAR(10),
 7)   Dno INT)" );
 8) if (DB::isError($q)) { die("table creation not successful – " . ←
                          $q->getMessage()); }
    ...                                            //terminate the program and
 9) $d->setErrorHandling(PEAR_ERROR_DIE);    ←     print the default error messages
    ...                                            if any subsequent errors occur
10) $eid = $d->nextID('EMPLOYEE');                 when accessing DB thru $d.
11) $q = $d->query("INSERT INTO EMPLOYEE VALUES
12)   ($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'])" );
    ...
13) $eid = $d->nextID('EMPLOYEE');
14) $q = $d->query('INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)',
15) array($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno']) );
```

CREATE A TABLE



```php
<?php
    require 'DB.php';
    $d = DB::connect('mysql://jrandom:!ItIsSecret@db.foo.com/test');
    if (DB::isError($d)) { die("cannot connect - " . $d->getMessage());}

    $q = $d->query("CREATE TABLE EMPLOYEE
        (Emp_id INT,
        Name VARCHAR(15),
        Job VARCHAR(10),
        Dno INT)"
        );

    if(DB::isError($q)) { die("table creation not successful - " . $q->getMessage());}
?>
```

Web site: http://140.125.84.81:81/db_example/DB/example7.php

Form data collection and record insertion

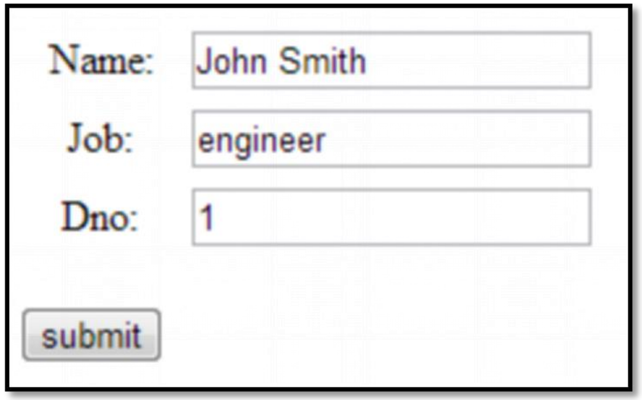
- Line 10-12 shows how information collected via forms can be stored in the database; Line 13-15 the other type of insertion

```
 0) require 'DB.php';
 1) $d = DB::connect('oci8://acct1:pass12@www.host.com/db1');
 2) if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage());}
    ...
 3) $q = $d->query("CREATE TABLE EMPLOYEE
 4)   (Emp_id INT,
 5)   Name VARCHAR(15),
 6)   Job VARCHAR(10),
 7)   Dno INT)" );
 8) if (DB::isError($q)) { die("table creation not successful – " .
                        $q->getMessage()); }
    ...
 9) $d->setErrorHandling(PEAR_ERROR_DIE);
    ...
10) $eid = $d->nextID('EMPLOYEE');
11) $q = $d->query("INSERT INTO EMPLOYEE VALUES
12)   ($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'])" );
    ...
13) $eid = $d->nextID('EMPLOYEE');
14) $q = $d->query('INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)',
15) array($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno']) );
```

two types of insertions:
- with one argument
- with two arguments

INSERT INTO DATABASE

```php
<?php
    require 'DB.php';

    $emp_name = $_POST['emp_name'];
    $emp_job = $_POST['emp_job'];
    $emp_dno = $_POST['emp_dno'];

    $d = DB::connect('mysql://jrandom:!ItIsSecret@db.foo.com/test');
    if (DB::isError($d)) { die("cannot connect - " . $d->getMessage());}

    $d->setErrorHandling(PEAR_ERROR_DIE);
    $eid = $d->nextID('EMPLOYEE');
    $q = $d->query('INSERT INTO EMPLOYEE VALUES (?,?,?,?)',
        array($eid, $emp_name, $emp_job, $emp_dno));

    print "<BR/>";
    if($q)
        print "SUCCEED";
?>
```

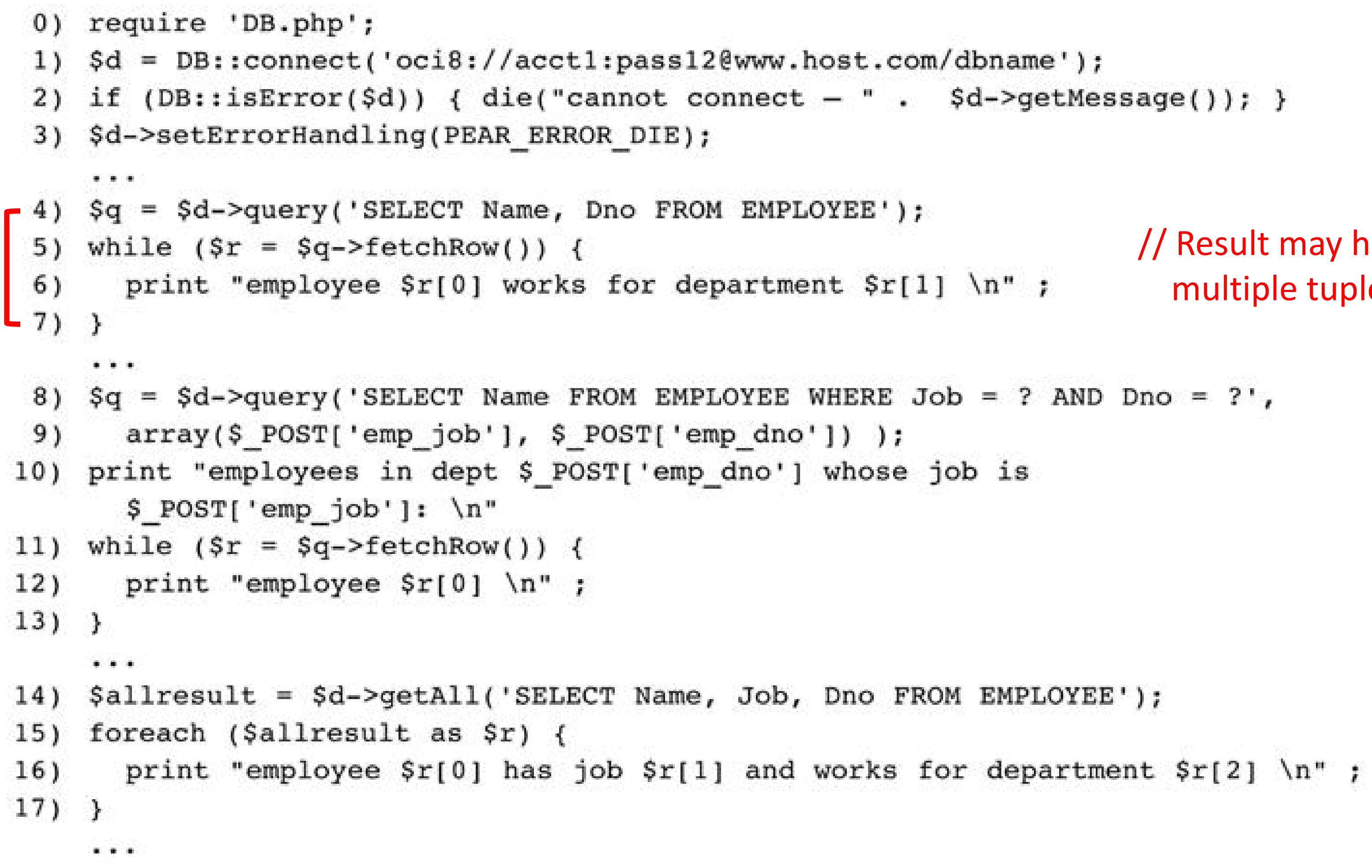


Name:	John Smith
Job:	engineer
Dno:	1

Web site: http://140.125.84.81:81/db_example/DB/example8-1.php

Retrieve Data from Table

- Lines 4-7 retrieves name and department number of all employee records
  - Uses variable $q to store query results
  - $q->fetchrow retrieves the next row/record



```
 0) require 'DB.php';
 1) $d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
 2) if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage()); }
 3) $d->setErrorHandling(PEAR_ERROR_DIE);
    ...
 4) $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
 5) while ($r = $q->fetchRow()) {
 6)   print "employee $r[0] works for department $r[1] \n" ;
 7) }
    ...
 8) $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
 9)   array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
      $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)   print "employee $r[0] \n" ;
13) }
    ...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)   print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
    ...
```

// Result may have multiple tuples.

Dynamic Query based on User Input

- Lines 8-13 is a dynamic query (conditions based on user selection)
- Retrieves names of employees who have specified job and work in a particular department
  - Values for these are entered through forms

```
 0) require 'DB.php';
 1) $d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
 2) if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage()); }
 3) $d->setErrorHandling(PEAR_ERROR_DIE);
    ...
 4) $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
 5) while ($r = $q->fetchRow()) {
 6)   print "employee $r[0] works for department $r[1] \n" ;
 7) }
    ...
 8) $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
 9)   array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
      $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)   print "employee $r[0] \n" ;
13) }
    ...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)   print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
    ...
```

← //$d->query: one arguments

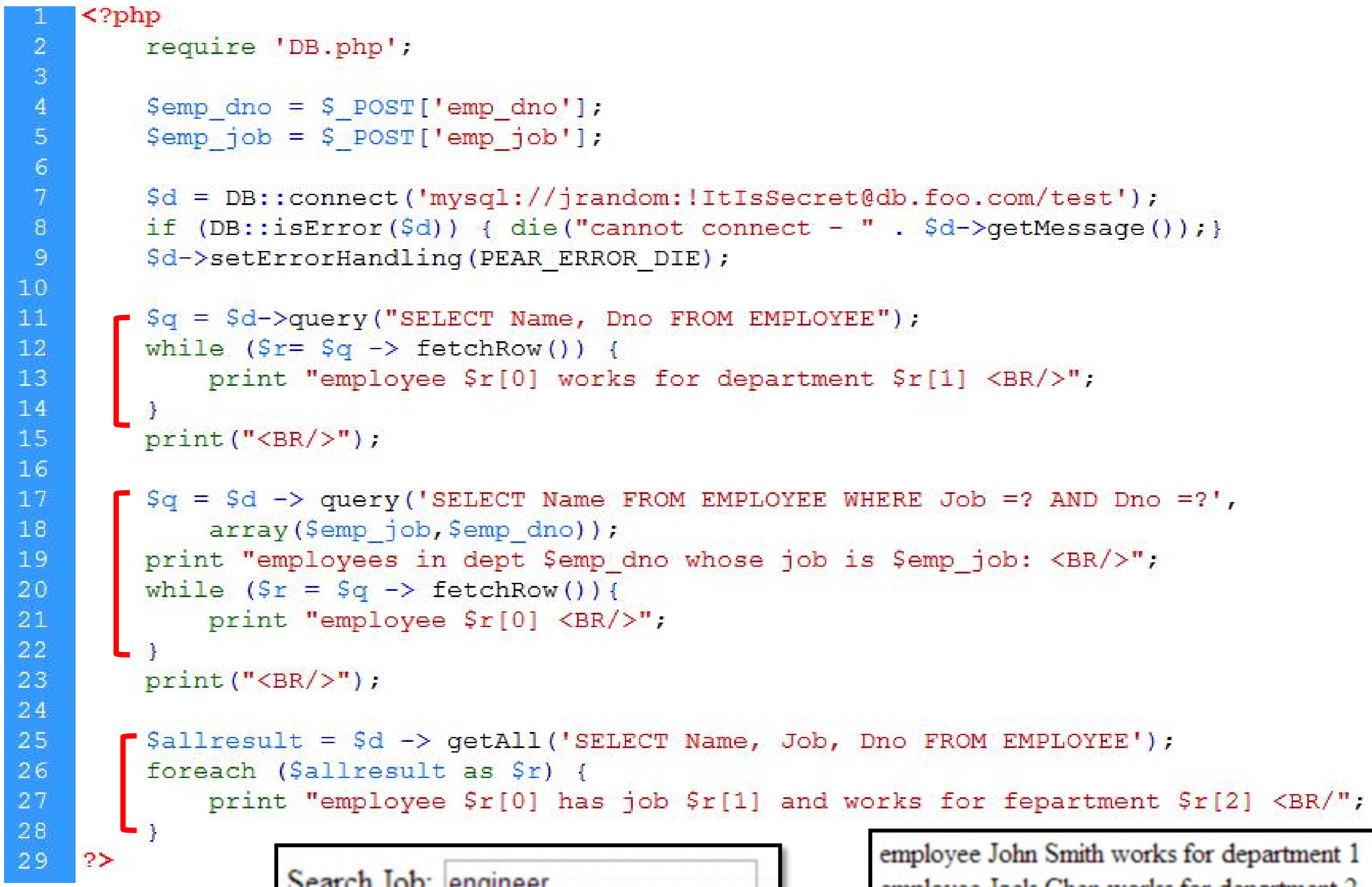
//$d->query: two arguments

Query and Looping over Retrieved Data

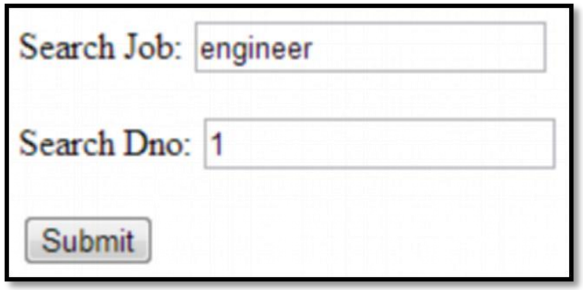
- Lines 14-17 is an alternative way of specifying a query and looping over its records
  - Function $d->getAll holds all the records in $allresult
  - For loop iterates over each row

```
 0) require 'DB.php';
 1) $d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
 2) if (DB::isError($d)) { die("cannot connect – " .  $d->getMessage()); }
 3) $d->setErrorHandling(PEAR_ERROR_DIE);
    ...
 4) $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
 5) while ($r = $q->fetchRow()) {
 6)   print "employee $r[0] works for department $r[1] \n" ;
 7) }
    ...
 8) $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
 9)   array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
      $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)   print "employee $r[0] \n" ;
13) }
    ...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)   print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
    ...
```

QUERY DATABASE

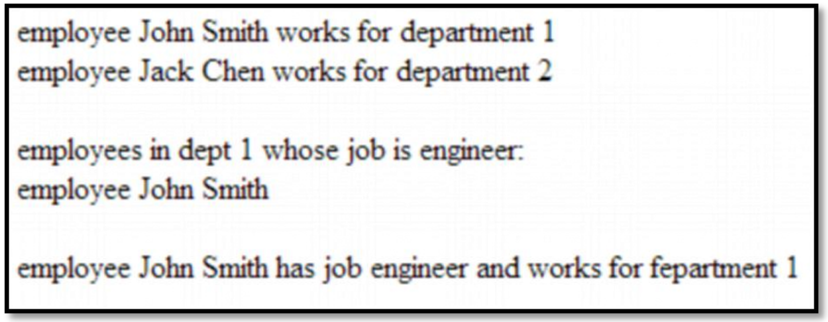


```php
<?php
    require 'DB.php';

    $emp_dno = $_POST['emp_dno'];
    $emp_job = $_POST['emp_job'];

    $d = DB::connect('mysql://jrandom:!ItIsSecret@db.foo.com/test');
    if (DB::isError($d)) { die("cannot connect - " . $d->getMessage());}
    $d->setErrorHandling(PEAR_ERROR_DIE);

    $q = $d->query("SELECT Name, Dno FROM EMPLOYEE");
    while ($r= $q -> fetchRow()) {
        print "employee $r[0] works for department $r[1] <BR/>";
    }
    print("<BR/>");

    $q = $d -> query('SELECT Name FROM EMPLOYEE WHERE Job =? AND Dno =?',
        array($emp_job,$emp_dno));
    print "employees in dept $emp_dno whose job is $emp_job: <BR/>";
    while ($r = $q -> fetchRow()){
        print "employee $r[0] <BR/>";
    }
    print("<BR/>");

    $allresult = $d -> getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
    foreach ($allresult as $r) {
        print "employee $r[0] has job $r[1] and works for fepartment $r[2] <BR/";
    }
?>
```



Search Job: engineer

Search Dno: 1



```
employee John Smith works for department 1
employee Jack Chen works for department 2

employees in dept 1 whose job is engineer:
employee John Smith

employee John Smith has job engineer and works for fepartment 1
```

Web site: http://140.125.84.81:81/db_example/DB/example9-1.php

PHP Connect to MySQL

- PHP 5 and later can work with a MySQL database using
  - MySQLi extension (the ‘i‘ stands for improved)
  - PDO (PHP Data Objects)
- Three ways of working with PHP and MySQL
  - MySQLi (object-oriented)
  - MySQLi (procedural)
  - PDO (can work on 12 different DB systems)
- MySQLi extension is automatically installed with PHP
- Need to install PDO

Current version as of 2017.10: PHP 7.1
Reference: https://www.w3schools.com/php/php_mysql_connect.asp

Connect to MySQL using MySQLi (Object-Oriented)

```php
<!DOCTYPE html>
<html>
<body>

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<br> id: ". $row["id"]. " - Name: ". $row["firstname"]. " " . $row["lastname"] . "<br>";
    }
} else {
    echo "0 results";
}

$conn->close();
?>

</body>
</html>
```

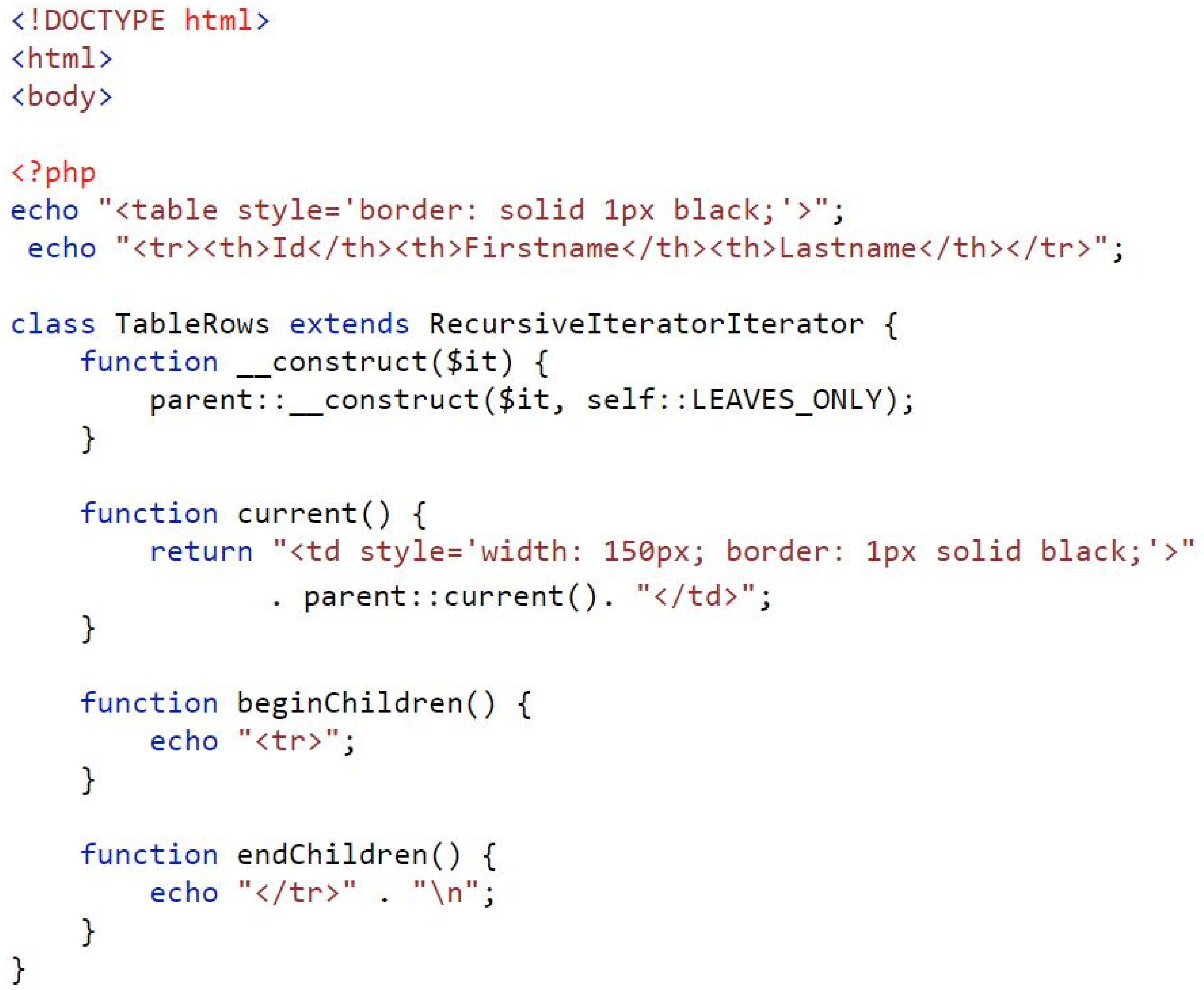
```
id: 1 - Name: John Doe
id: 2 - Name: Mary Moe
id: 3 - Name: Julie Dooley
```

Connect to MySQL using MySQLi (Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
             $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

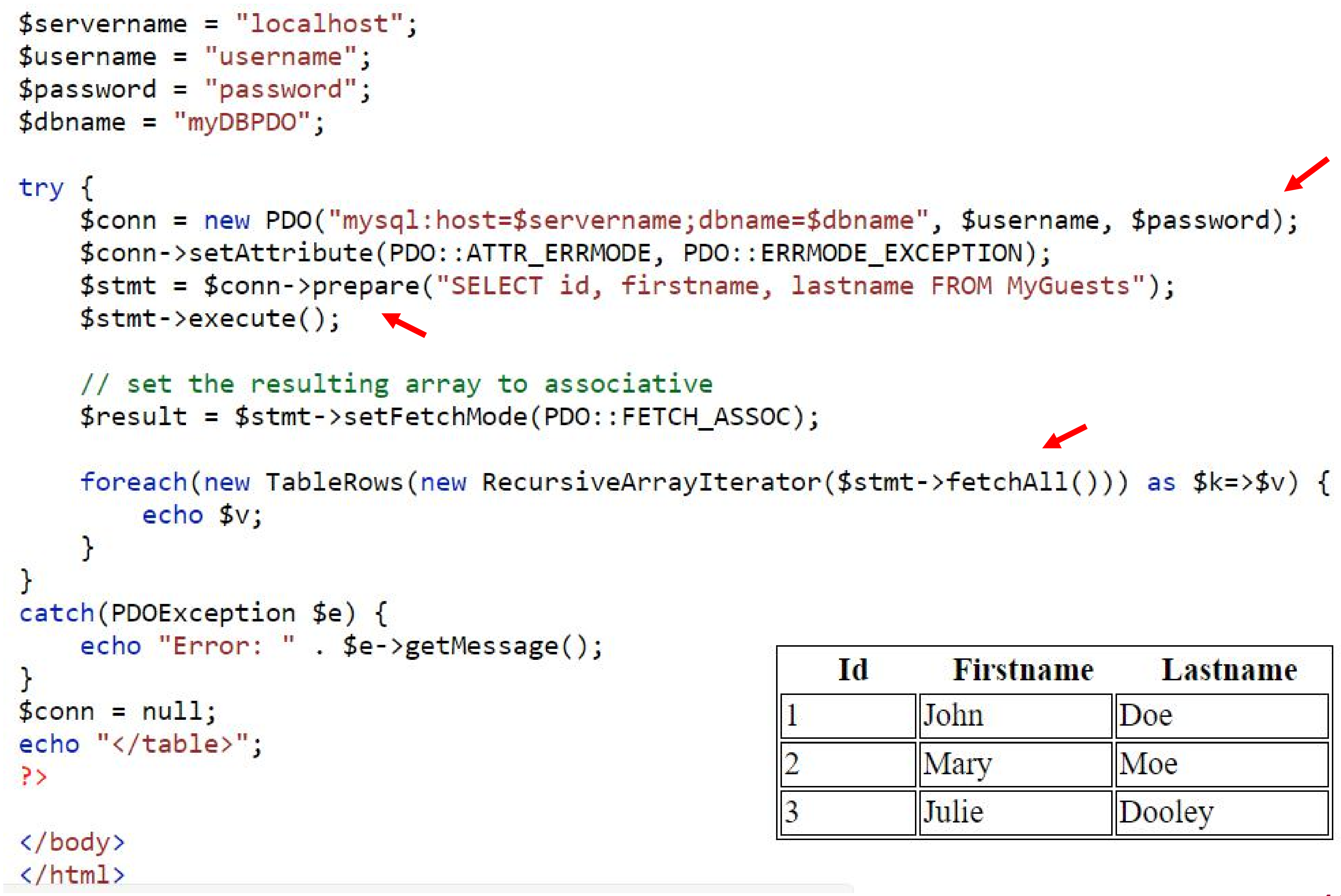
```
id: 1 - Name: John Doe
id: 2 - Name: Mary Moe
id: 3 - Name: Julie Dooley
```

Connect to MySQL using PDO$_{\text{--1}}$

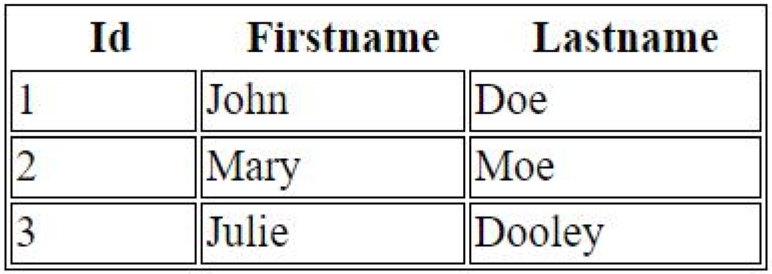


```
<!DOCTYPE html>
<html>
<body>

<?php
echo "<table style='border: solid 1px black;'>";
 echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";

class TableRows extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width: 150px; border: 1px solid black;'>"
               . parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}
```

Connect to MySQL using PDO$_{--2}$



```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);

    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>

</body>
</html>
```



Id	Firstname	Lastname
1	John	Doe
2	Mary	Moe
3	Julie	Dooley