PHP INTRODUCTION

DR. SHAWN WANG
DEPARTMENT OF COMPUTER SCIENCE
CAL STATE FULLERTON

A SIMPLE PHP EXAMPLE

```
<html>
 <head>
  <title>First PHP Example</title>
 </head>
 <body>
 <?php echo "<p><h1>Output</h1>";
       echo "<h2>Output</h2>";
       echo "<h3>Output</h3></p>";
  ?>
 <script language="PHP">
  echo "\n<b>More PHP Output</b><br />\n";
 </script>
 </body>
</html>

-----------------------------------------------------------------
<html>
 <head>
  <title>First PHP Example</title>
 </head>
 <body>
 <p><h1>Output</h1><h2>Output</h2><h3>Output</h3></p>
<b>More PHP Output</b><br />
 </body>
</html>
```

USING MYSQL DATABASE

- Connect with MySQL RDBMS
  - $link=mysqli_connect($hostName, $userName, $password, $dbName) or die("Unable to connect to host $hostName");

EXECUTING SQL

- Basic information searches
  - $SQL = "SELECT FirstName, LastName, DOB, Gender FROM Patients WHERE Gender = '$Gender' ORDER BY FirstName DESC";
    $Patients = $link->query($SQL);
- Editing, adding, and deleting records and tables
  - $SQL = "INSERT INTO Patients (FirstName, LastName) VALUES('$firstName', '$lastName')";
    $Patients = $link->query($SQL);

CLOSING THE CONNECTION

- Cleaning up: close the database connection
  - $Patients->free_result();
  - $link->close();

PHP DATA TYPES

boolean

TRUE or FALSE

integer

Platform dependent – size of one machine word 32 bits on most machines

float

Double precision. We could call it a double, but since we don't declare variables (we will discuss shortly) float works

USING STRING

string

- We have single-quoted and double-quoted string literals
  - Double quoted allows for more escape sequences and allows variables to be interpolated into the string
- Length can be arbitrary
- Easy conversion back and forth between strings and numbers
- Can be indexed – the preferred way is using curly braces

```
$mystring = "hello";
echo $mystring{1};
```

  - Output here is 'e'

PHP VARIABLES

- All PHP variables begin with the $
  - Variable names can begin with an underscore
  - Otherwise rules are similar to most other languages
- Variables are dynamically typed
  - No type declarations
    - Variables are BOUND or UNBOUND
      - Unbound variables have the value NULL
    - Type information is obtained from the current bound value

PREDEFINED VARIABLES

PHP programs have access to a large number of predefined variables

- These variables allow the script access to server information, form parameters, environment information, etc.
- Ex:
  - $_SERVER is an array containing much information about the server
  - $_POST is an array containing variables passed to a script via HTTP POST
  - $_ENV is an array containing environment information

PHP EXPRESSIONS AND OPERATORS

- Similar to those in C / C++
- Be careful with a few operators
  - / in PHP is always floating point division
    - To get integer division, we must cast to int

```
$x = 15;
$y = 6;
echo ($x/$y), (int) ($x/$y), "<BR />";
```

      - Output is `2.5 2`
  - Inequality operators do not compare strings
    - Will cast strings into numbers before comparing
    - To compare strings, use the C-like string comparison function, strcmp()

PHP CONTROL STRUCTURES

- Again, these are similar to those in C++
- **if**, **while**, **do**, **for**, **switch** are virtually identical to those in C++
  - PHP allows for an alternative syntax to designate a block in the if, while, for and switch statements
    - Open the block with **:** rather than **{**
    - Close the block with endif, endwhile, endfor, endswitch
      - Advantage to this syntax is readability
      - Now instead of seeing a number of close braces, we see different keywords to close different types of control structures

CREATING ARRAYS

- PHP Arrays can be created in a number of ways
  - Explicitly using the array() construct
  - Implicitly by indexing a variable
    - Unlike Perl, arrays in PHP do not have a different prefix for the variable name. Thus you cannot have a scalar and an array with the same variable name at the same time
    - However, you can test a variable to see if it is an array or not
      - is_array() returns true or false
  - Size will increase dynamically as needed

```
<html>
<body>
<?php
// username and password need to be replaced by your username and password
// dbname is the same as your username

$link = mysqli_connect('mariadbl', 'username', 'password','dbname');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<p>';

$result = $link->query("SELECT * FROM STUDENT");
$nor=$result->num_rows;

for($i=0; $i<$nor; $i++)
{
 $row=$result->fetch_assoc();
echo "SSN: ", $row["ssn"], "<br>";
echo "First NAME: ", $row["fname"], "<br>";
echo "Last NAME: ", $row["lname"], "<br>";
}
$result->free_result();
$link->close();
 ?>

</body>
</html>
```

```
<html>
<body>
<?php
// username and password need to be replaced by your username and password
// dbname is the same as your username

$link = mysqli_connect('mariadb', 'username', 'password','dbname');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<p>';

 $query = "SELECT * FROM STUDENT WHERE ssn=" . $_POST["sno"];
$result = $link->query($query);
$row=$result->fetch_assoc();
printf("SSN: %s<br>\n", $row["ssn"]);
printf("First NAME: %s<br>\n", $row["fname"]);
printf("Last NAME: %s<br>\n", $row["lname"]);
$result->free_result();
$link->close();
?>
</body>
</html>
```

THE HTML FORM FOR USER INPUTS

- <html>
- <body>

- <form action="sample.php" method="POST">
- Enter the social security number: <input type="text" name="sno" />
- <input type="submit" />
- </form>

- </body>
- </html>