jesuitas Logroño

SERVER-SIDE WEB PROGRAMMING
UNIT2: PROGRAMMING BASED ON EMBEDDED LANGUAGE

CFGS - DAW 2

Index

- Flow-Control Statements:
  - If
  - Switch
  - While
  - Do-While
  - For
  - Foreach

1. Flow-Control Statements

- if:
  - The if statement checks the truthfulness of an expression and, if the expression is true, evaluates a statement. An if statement looks like:

```
if ($user_validated)
    echo "Welcome!";
else
    echo "Access Forbidden!";
```

```
if ($user_validated) {
    echo "Welcome!";
    $greeted = 1;
}
else {
    echo "Access Forbidden!";
exit;}
```

1. Flow-Control Statements

- if:

```php
if ($user_validated):
    echo "Welcome!";
    $greeted = 1;
else:
    echo "Access Forbidden!";
    exit;
endif;
```

```php
<? if ($user_validated) :?>
    <table>
        <tr>
            <td>First Name:</td><td>Sophia</td>
        </tr>
        <tr>
            <td>Last Name:</td><td>Lee</td>
        </tr>
    </table>
<? else: ?>
    Please log in.
<? endif ?>
```

1. Flow-Control Statements

A.2.1. Without running it through the PHP engine, figure out what this program prints…

```
$age = 12;
$shoe_size = 13;
if ($age > $shoe_size) {
    print "Message 1.";
} elseif (($shoe_size++) && ($age > 20)) {
    print "Message 2.";
} else {
    print "Message 3.";
}
print "Age: $age. Shoe Size: $shoe_size";
```

1. Flow-Control Statements

A.2.2. In PHP 7.0 there is a new comparison operator called "spaceship operator <=>"…
Could you check php documentation in order to know what could be the output of this code?

```php
$a = 1 <=> 12.7;
echo "$a\n";

$b = "charlie" <=> "bob";
echo "$b\n";
```

1. Flow-Control Statements

- switch:
  - The value of a single variable may determine one of a number of different choices:

```
switch($name) {
    case 'ktatroe':
        // do something
        break;
    case 'dawn':
        // do something
        break;
    case 'petermac':
        // do something
        break;
    case 'bobk':
        // do something
        break;
}
```

```
switch($name):
case 'ktatroe':
    // do something
    break;
case 'dawn':
    // do something
    break;
case 'petermac':
    // do something
    break;
case 'bobk':
    // do something
    break;
endswitch;
```

1. Flow-Control Statements


```php
<html>
    <head>
        <title>Activity 2 Lesson 3</title>
    </head>
    <body>
        <?php
            $code=5;
            switch ($code) {
                case '0':
                    echo "You have chosen 0";
                    break;
                case '1':
                    echo "You have chosen 1";
                    break;

                case '2':
                    echo "You have chosen 2";
                    break;

                case '3':
                    echo "You have chosen 3";
                    break;

                default:
                    echo "I do not know which one you selected";
                    break;
            }
        ?>
    </body>
</html>
```

1. Flow-Control Statements

- Loops - **while**:
  - The simplest form of loop is the while statement. If the expression evaluates to true, the statement is executed and then the expression.

```php
$total = 0;
$i = 1;
while ($i <= 10) {
    $total += $i;
    $i++;
}
```

```php
$total = 0;
$i = 1;
while ($i <= 10):
    $total += $i;
    $i++;
endwhile;
```

1. Flow-Control Statements

A.2.3. Create a new function called *selection*, that prints a <select> menu with a while(). The method should take two parameters (the number of options) and the name of the <select>.

```
<select name="people"><option>1</option>
<option>2</option>
<option>3</option>
<option>4</option>
<option>5</option>
<option>6</option>
<option>7</option>
<option>8</option>
<option>9</option>
<option>10</option>
</select>[Finished in 0.0s]
```

1. Flow-Control Statements

- Loops - **while**:
  - You can prematurely exit a loop with the **break** keyword.

```php
$total = 0;
$i = 1;
while ($i <= 10) {
    if ($i == 5) {
        break; // breaks out of the loop
    }
    $total += $i;
    $i++;
}
```

1. Flow-Control Statements

- Loops - **while**:
  - Optionally, you can put a number after the break keyword indicating how many levels of loop structures to break out of:

```php
$i = 0;
$j = 0;
while ($i < 10) {
    while ($j < 10) {
        if ($j == 5) {
            break 2; // breaks out of two while loops
        }
        $j++;
    }
    $i++;
}
echo "{$i}, {$j}";
```

1. Flow-Control Statements

- Loops - **while**:
  - Use a do/while loop to ensure that the loop body is executed at least once (the first time):

```php
$total = 0;
$i = 1;
do {
    $total += $i++;
} while ($i <= 10);
```

1. Flow-Control Statements

- Loops - **for**:
  - The for statement is similar to the while statement, except it adds counter initialization and counter manipulation expressions, and is often shorter and easier to read than the equivalent while loop.

```php
$total = 0;
for ($i= 1; $i <= 10; $i++) {
    $total += $i;
}
```

```php
$total = 0;
for ($i = 1; $i <= 10; $i++):
    $total += $i;
endfor;
```

1. Flow-Control Statements

A.2.4. The same as A2.3 but using a *for* loop instead of a while.

1. Flow-Control Statements

- Loops - **for**:
  - You can specify multiple expressions for any of the expressions in a for statement by separating the expressions with commas. For example:

```php
$total = 0;
for ($i = 0, $j = 0; $i <= 10; $i++, $j *= 2) {
    $total += $j;
}
```

1. Flow-Control Statements

- Loops – **foreach!!**:
  - The foreach statement allows you to iterate over elements in an array. To loop over an array, accessing the value at each key, use:

```
foreach ($array as $current) {
// ...
}

foreach ($array as $current):
// ...
endforeach;
```

1. Flow-Control Statements

- Loops - **foreach**:
  - To loop over an array, accessing both key and value, use:

```
foreach ($array as $key => $value) {
// ...
}

foreach ($array as $key => $value):
// ...
endforeach;
```

1. Flow-Control Statements

A.2.5. Create an array this way (copy and paste):

"$ceu = array("Italy"=>"Rome", "Luxembourg"=>"Luxembourg", "Belgium"=> "Brussels", "Denmark"=>"Copenhagen", "Finland"=>"Helsinki", "France" => "Paris", "Slovakia"=>"Bratislava", "Slovenia"=>"Ljubljana", "Germany" => "Berlin", "Greece" => "Athens", "Ireland"=>"Dublin", "Netherlands"=>"Amsterdam", "Portugal"=>"Lisbon", "Spain"=>"Madrid", "Sweden"=>"Stockholm", "United Kingdom"=>"London", "Cyprus"=>"Nicosia", "Lithuania"=>"Vilnius", "Czech Republic"=>"Prague", "Estonia"=>"Tallin", "Hungary"=>"Budapest", "Latvia"=>"Riga", "Malta"=>"Valetta", "Austria" => "Vienna", "Poland"=>"Warsaw") ;"

Making use of a **for-each** flow control print out the capital and country name of the previous array:

"The capital of Italy is Rome

The capital of Luxembourg is Luxembourg

…"