APPDET

APPLICATION DEVELOPMENT &
EMERGING TECHNOLOGY

Prof. ROEL C. TRABALLO
roel.traballo@umak.edu.ph

INTRODUCTION TO WEB DEVELOPMENT

WEEK-5

- PHP Control Structures/Statements
  - Looping (for, while, do while)

PHP FUNDAMENTALS: LOOPING

PHP FOR LOOP

- The for statement allows you to execute a code block repeatedly.
- Syntax:

```
for (start; condition; increment) {
        statement;
}
```

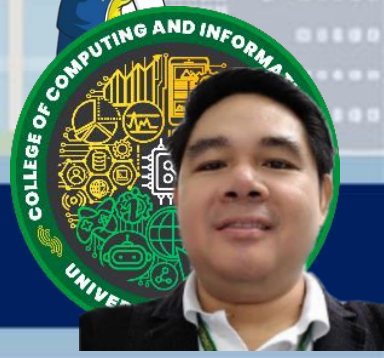
- PHP allows you to specify multiple expressions in the start, condition, and increment of the for statement.
- In addition, you can leave the start, condition, and increment empty, indicating that PHP should do nothing for that phase.
- The following flowchart illustrates how the for statement works:

How it works.

✓ The start is evaluated **once** when the loop starts.

✓ The condition is evaluated once in each iteration. If the condition is true, the statement in the body is executed. Otherwise, the loop ends.

✓ The increment expression is evaluated once after each iteration.



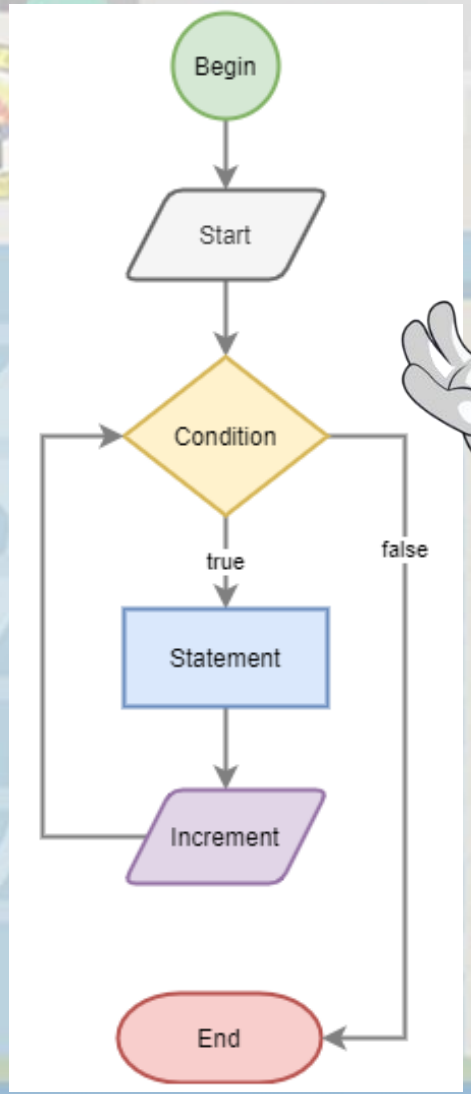
PHP FUNDAMENTALS: LOOPING

PHP FOR Loop

- The for statement allows you to execute a code block repeatedly.
- Syntax:

```
for (start; condition; increment) {
        statement;
}
```

- PHP allows you to specify multiple expressions in the start, condition, and increment of the for statement.
- In addition, you can leave the start, condition, and increment empty, indicating that PHP should do nothing for that phase.
- The following flowchart illustrates how the for statement works:



PHP FUNDAMENTALS: LOOPING

PHP FOR Loop

- When you leave all three parts empty, you should use a break statement to exit the loop at some point. Otherwise, you'll have an infinite loop:
- **Syntax:**

```
for (; ;) {
        // do something
        // ...

        // exit the loop
        if (condition) {
                break;
        }
}
```

Example:

```
<?php
$total = 0;

for ($i = 1; $i <= 10; $i++) {
        $total += $i;
}
echo $total;
?>
```

Output:

```
55
```

How it works.

- First, initialize the $total to zero.
- Second, start the loop by setting the variable $i to 1. This initialization will be evaluated once when the loop starts.
- Third, the loop continues as long as $i is less than or equal to 10. The expression $i <= 10 is evaluated once after every iteration.
- Fourth, the expression $i++ is evaluated after each iteration.
- Finally, the loop runs exactly 10 iterations and stops once $i becomes 11.

PHP FUNDAMENTALS: LOOPING

- **PHP FOR Loop**
  - **Alternative Syntax:**

```
for (start; condition; increment):
    statement;
endfor;
```

Example:

```
<?php
$total = 0;

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

echo $total;
```

Output:

```
55
```

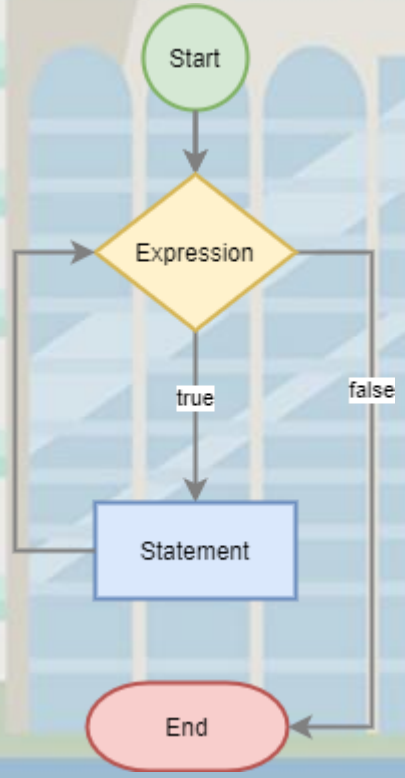

PHP FUNDAMENTALS: LOOPING

PHP WHILE Loop

- The while statement executes a code block as long as an expression is true. The syntax of the while statement is as follows:
- **Syntax:**

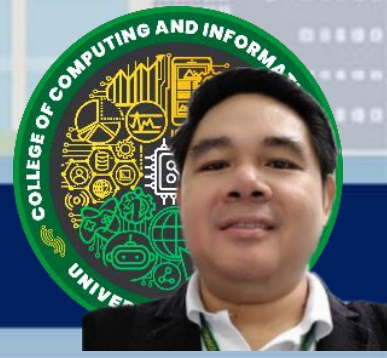
```
while (expression) {
        statement;
}
```

```
while (expression)
        statement;
```



How it works.

- First, PHP evaluates the expression. If the result is true, PHP executes the statement.
- Then, PHP re-evaluates the expression again. If it's still true, PHP executes the statement again. However, if the expression is false, the loop ends.
- If the expression evaluates to false before the first iteration starts, the loop ends immediately.
- Since PHP evaluates the expression before each iteration, the while loop is also known as a pretest loop.



PHP FUNDAMENTALS: LOOPING

PHP WHILE Loop

- The while statement executes a code block as long as an expression is true. The syntax of the while statement is as follows:
- **Syntax:**

```
while (expression) {
        statement;
}
```

```
while (expression)
        statement;
```

Example:

```
<?php
$total = 0;
$number = 1;
while ($number <= 10) {
   $total += $number; $number++;
}
echo $total;
?>
```

Output:

```
55
```

PHP FUNDAMENTALS: LOOPING

PHP WHILE Loop

- The while statement executes a code block as long as an expression is true. The syntax of the while statement is as follows:
- **Alternative Syntax:**

```
while (expression):
        statement;
endwhile;
```

Example:

```
<?php
$total = 0;
$number = 1;

while ($number <= 10) :
    $total += $number; $number++;
endwhile;

echo $total;
```

Output:

```
55
```

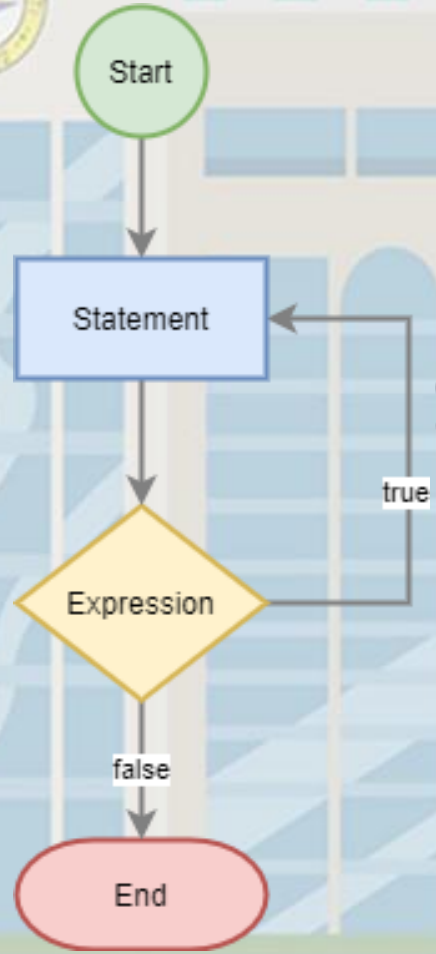
PHP FUNDAMENTALS: LOOPING

PHP DO WHILE Loop

- The PHP do...while statement allows you to execute a code block repeatedly based on a Boolean expression.
- **Syntax:**

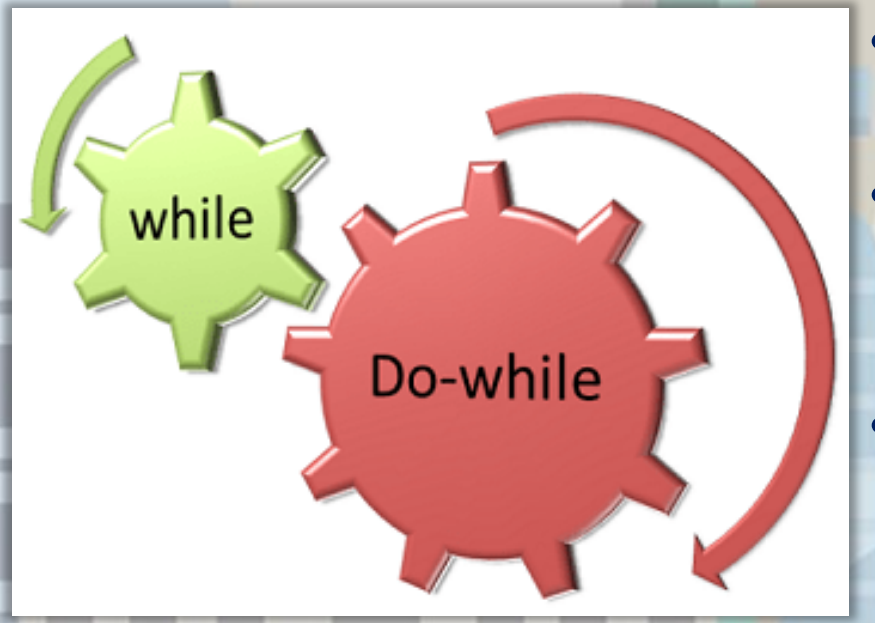
```
do {
        statement;
} while (expression);
```

- Unlike the **while** statement, PHP evaluates the expression at the end of each iteration. It means that the loop always executes at least once, even the expression is false before the loop enters.



PHP FUNDAMENTALS: LOOPING

- **PHP WHILE versus DO WHILE Loop**



- **The differences between the do...while and while statements are:**
- PHP executes the statement in do...while at least once, whereas it won't execute the statement in the while statement if the expression is false.
- PHP evaluates the expression in the do...while statement at the end of each iteration. Conversely, PHP evaluates the expression in the while statement at the beginning of each iteration.

PHP FUNDAMENTALS: LOOPING

- **PHP DO WHILE Loop**

Example #1:

```
<?php
$i = 0;
do {
   echo $i;
}
while ($i > 0);
?>
```

- In the following example, the code block inside the do...while loop statement executes precisely one time.
- The code inside the loop body executes first to display the variable $i. Because the value of the $i is 0, the condition is met, the loop stops.

Example #2:

```
<?php
$i = 10;
do {
   echo $i  . '<br>';
   $i--;
}
while ($i > 0);
?>
```

- In the following example, the code block inside the do...while loop executes ten times:

PHP FUNDAMENTALS: JUMP STATEMENTS

PHP BREAK

- The break statement terminates the execution of the current *for*, *do...while*, *while*, or *switch* statement. This tutorial focuses on how to use the break statement with the loops.
- Typically, you use the break statement with the *if* statement that specifies the condition for the terminating loop.
- The break statement accepts an optional number that specifies the number of nested enclosing structures to be broken out of.

Example:

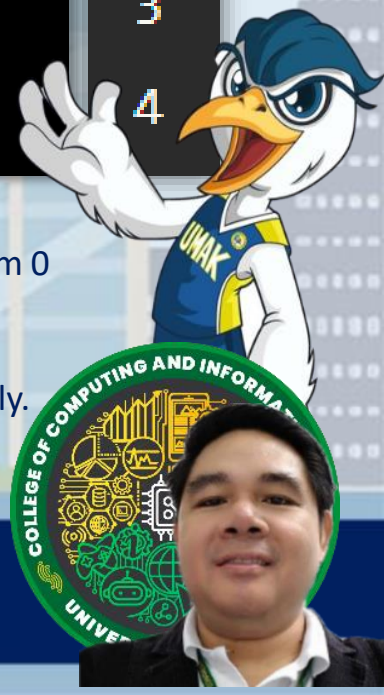
```
<?php
for ($i = 0; $i < 10; $i++) {
    if ($i === 5) {
        break;
    }
    echo "$i\n";
}
?>
```

Output:

```
0
1
2
3
4
```

How it works.

- The for statement would execute 10 times from 0 to 9.
- However, when the $i variable reaches 5, the break statement ends the loop immediately.
- The control is passed to the statement after the for statement.



PHP FUNDAMENTALS: JUMP STATEMENTS

PHP CONTINUE

- The continue statement is used within a loop structure such as for, do...while, and while loop. The continue statement allows you to immediately skip all the statements that follow it and start the next iteration from the beginning.
- Like the break statement, the continue statement also accepts an optional number that specifies the number of levels of enclosing loops it will skip.
- If you don't specify the number that follows the continue keyword, it defaults to 1. In this case, the continue statement only skips to the end of the current iteration.
- Typically, you use the continue statement with the if statement that specifies the condition for skipping the current iteration.

Example:

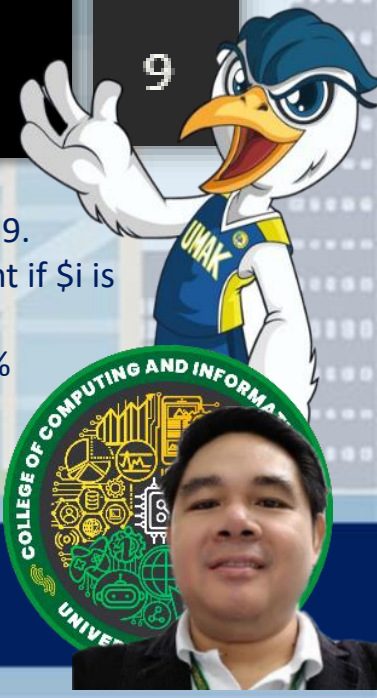
```
<?php
for ($i = 0; $i < 10; $i++)  {
    if ($i % 2 === 0) {
        continue;
    }
    echo "$i\n";
}
```

Output:

```
1
3
5
7
9
```

How it works.

- First, use a for loop to iterate from 0 to 9.
- Second, skip the current echo statement if $i is an even number.
- The $i is an even number when the $i % 2 returns 0.
- As a result, the output shows only the odd numbers.

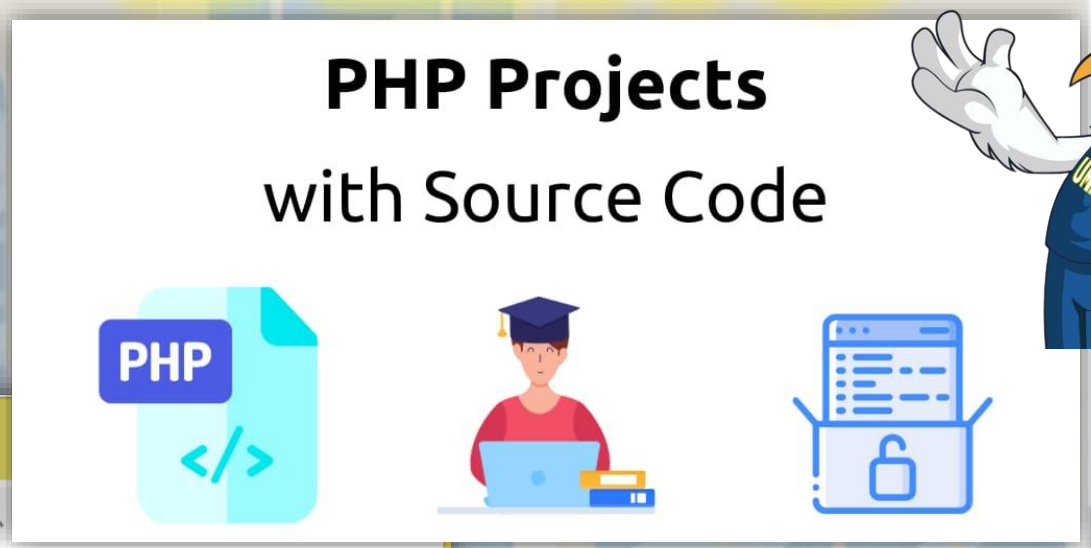


PHP FUNDAMENTALS – LOOPING

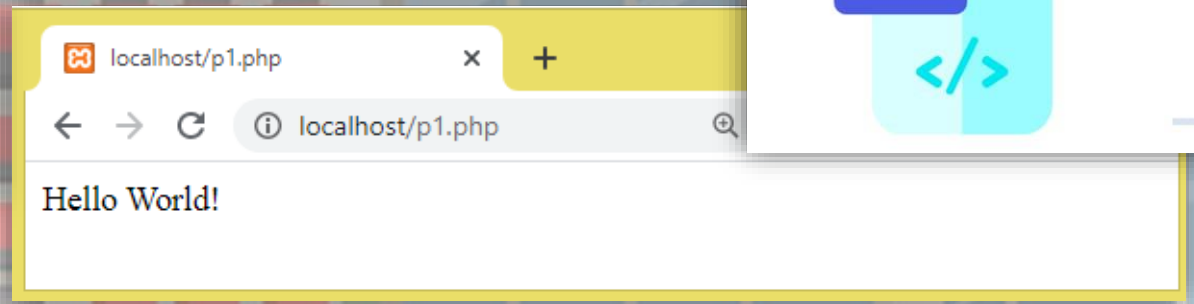
How to run PHP code in XAMPP

- **Step 1**: Create a simple PHP program like hello world.
- **Step 2**: Save the file with p1.php name in the **htdocs** folder, which resides inside the xampp folder.
  - ✓ Note: PHP program must be saved in the htdocs folder, which resides inside the xampp folder, where you installed the XAMPP. Otherwise it will generate an error - Object not found.
- **Step 3**: Run the XAMPP server and start the Apache and MySQL.
- **Step 4**: Now, open the web browser and type localhost http://localhost/hello.php on your browser window.
- **Step 5**: The output of ***p1.php*** program will be shown in the browser

DEMO



```
<?php
   echo "Hello World!";
?>
```



APPDET

APPLICATION DEVELOPMENT & EMERGING TECHNOLOGY