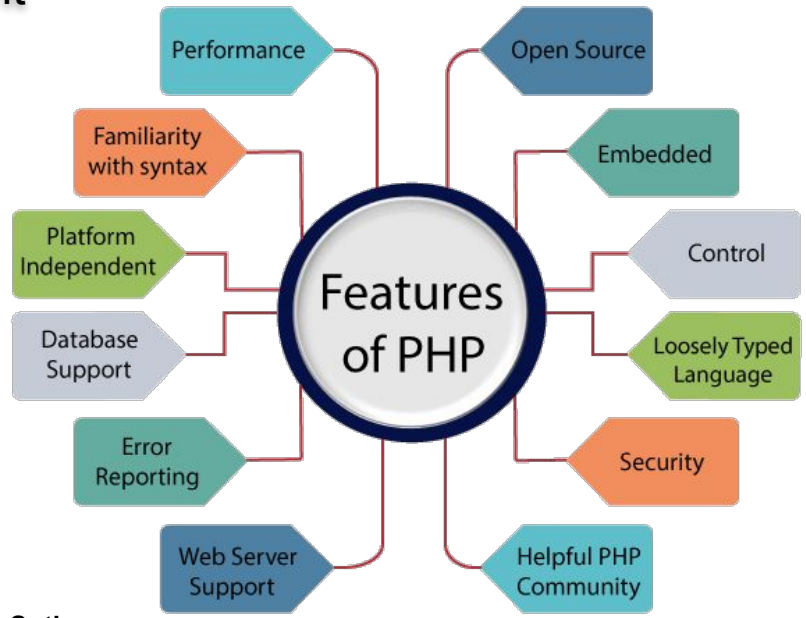
Web Development III

Block: I

Prepared By: Soniya Suthar

Block: I - Agenda

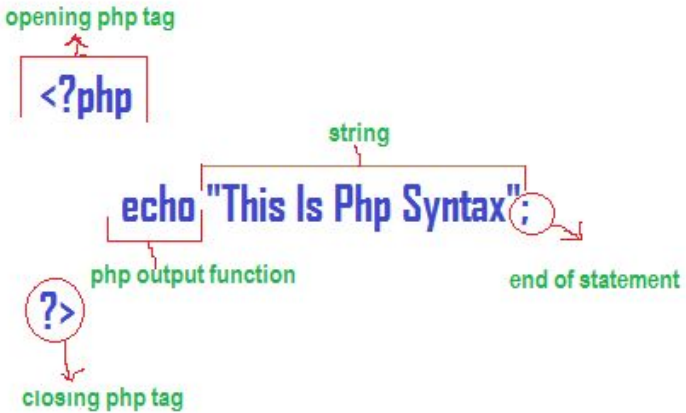
- Introduction, use & Benefits of PHP
- Environment Set-up, Syntax, Echo/Print
- Variables, Data Types, Constants
- String Functions, Array, Operators
- Decision Making & Branching
- Decision Making & Looping



PHP Introduction, use, Benefits and Syntax

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language, free to download and use
- PHP scripts are executed on the server, can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can send and receive cookies, can encrypt data
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- A PHP script can be placed anywhere in the document.
- A PHP script starts with `<?php` and ends with `?>`:

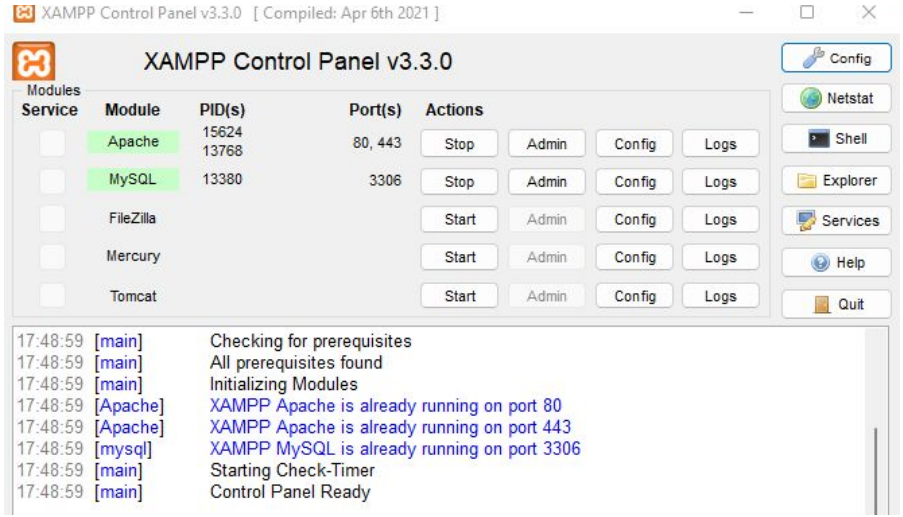
```
<?php

      // PHP code goes here

?>
```



- In PHP, keywords (e.g. `if`, `else`, `while`, `echo`, etc.), classes, functions, and user-defined functions are not case-sensitive.

Environment setup

- Tools Needed
  - Web Server: (XAMPP server/ WAMP server)
  - Editor : (Notepad, Notepad++ or Dreamweaver editor)
  - Database : (MySQL, Oracle)



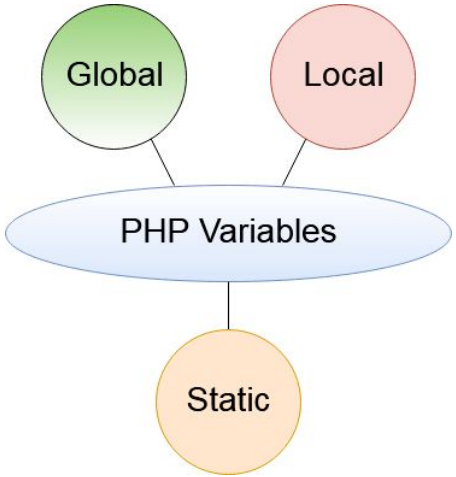
- Now open web browser like Google chrome and type **"localhost"** and check XAMPP server welcome page is appeared or not. If appeared then our server is started.

Echo/print

- With PHP, there are two basic ways to get output: `echo` and `print`.
- `echo` and `print` are more or less the same. They are both used to output data to the screen.
- The differences are small: `echo` has no return value while `print` has a return value of 1 so it can be used in expressions.
- `echo` can take multiple parameters while `print` can take one argument. `echo` is marginally faster than `print`.

```
<?php
    echo "<h2>PHP is Fun!</h2>";
    echo "Hello world!<br>";
    echo "Let's learn PHP!<br>";
    print "Hello world!<br>";
    print "<h2>PHP is Fun!</h2>";
    echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

Variables

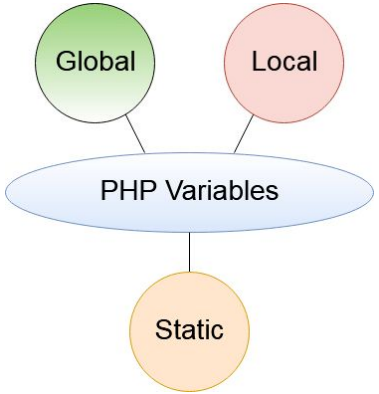


- Variables are "containers" for storing information.
- In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
<?php
	$txt = "Hello world!";
	$a = 5;
	$b = 10.5;
?>
```

- Rules for PHP variables:
  - A variable starts with the $ sign, followed by the name of the variable
  - A variable name must start with a letter or the underscore character, cannot start with a number
  - A variable name can only contain alphanumeric characters and underscores (A-z, 0-9, and _)
  - Variable names are case-sensitive ($abc and $ABC are two different variables)
  - PHP variable names are case-sensitive!

Scope of Variable

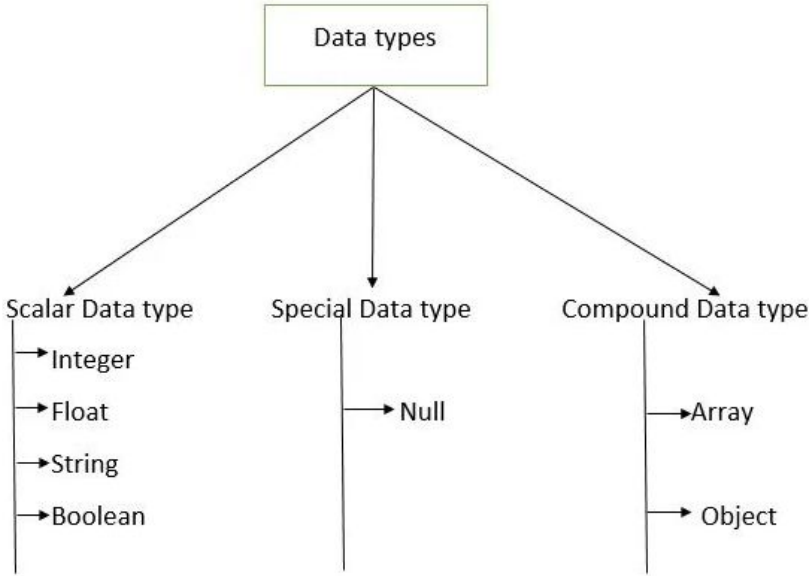


- PHP has three different variable scopes:
  a. Local
  b. Global
  c. Static
- A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function.
- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
- A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function.
- The `global` keyword is used to access a global variable from within a function.
- PHP also stores all global variables in an array called `$GLOBALS[index]`. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. To do this use the `static` keyword when you first declare the variable.

Data Types

- Variables can store data of different types, and different data types can do different things.

- PHP supports the following data types:
  - String
  - Integer
  - Float
  - Boolean
  - Array
  - Object
  - NULL



Constants

- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- Constants are automatically global and can be used across the entire script.
- To create a constant, use the `define()` function.
- Syntax:

```
define(name, value, case-insensitive)
```

- Example:

```
<?php
	define("HI", "Soniya Suthar here..", true);
	echo hi;
?>
```

String Functions

- strlen() - Return the Length of a String

```
<?php echo strlen("Hello world!"); // outputs 12 ?>
```

- str_word_count() - Count Words in a String

```
<?php echo str_word_count("Hello world!"); // outputs 2 ?>
```

- strrev() - Reverse a String

```
<?php echo strrev("Hello world!"); // outputs !dlrow olleH ?>
```

- strpos() - Search For a Text Within a String

```
<?php echo strpos("Hello world!", "world"); // outputs 6 ?>
```

- str_replace() - Replace Text Within a String

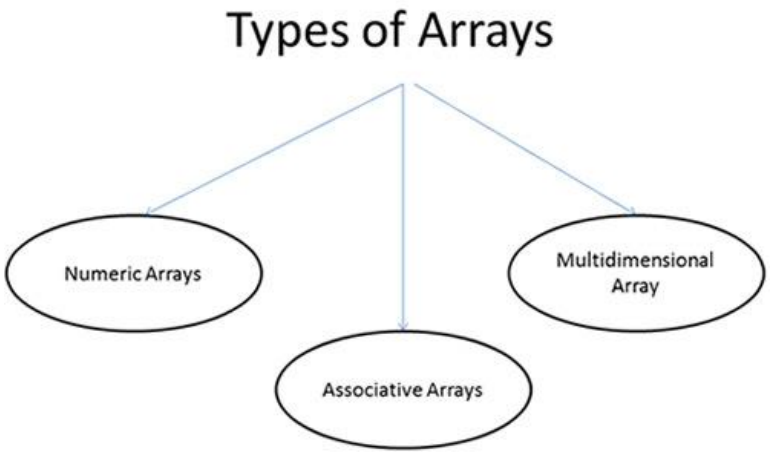
```
<?php echo str_replace("world", "Dear", "Hello world!"); // outputs Hello Dear! ?>
```

Prepared By: Soniya Suthar

Arrays

- An array is a special variable, which can hold more than one value at a time.
- In PHP, the `array()` function is used to create an array.
- print_r() function is used to print whole array.
- To perform operations on array, there are many functions available in PHP like, count(), sort() and is_array() etc.

```php
<?php

$colors = array("blue", "black", "cyan");

echo "I like " . $colors[0] . "the most.";

?>
```

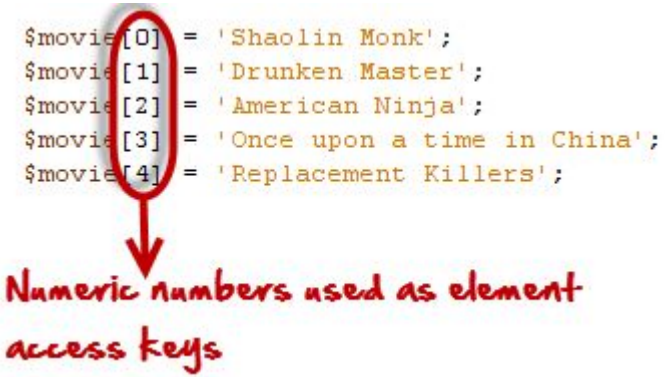


- In PHP, there are three types of arrays:
  - Indexed arrays - Arrays with a numeric index
  - Associative arrays - Arrays with named keys
  - Multidimensional arrays - Arrays containing one or more arrays

Indexed arrays

- Numeric(Indexed) arrays use number as access keys.
- An access key is a reference to a memory slot in an array variable.

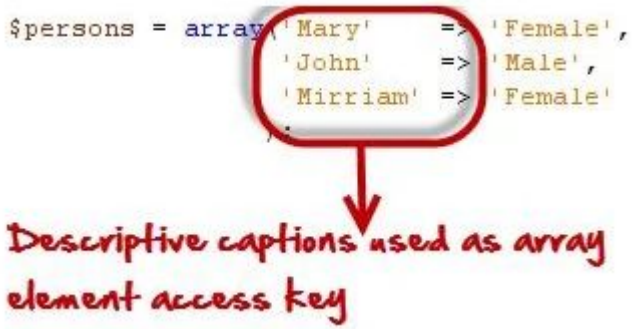
```
<?php
        $movie = array(0 => "Shaolin Monk",
                       1 => "Drunken Master",
                       2 => "American Ninja",
                       3 => "Once upon a time in China",
                       4 =>"Replacement Killers" );
        echo $movie[4];
?>
```



Associative arrays

- Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.

```
<?php
      $persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
      echo $persons;
      echo "";
      echo "Mary is a " . $persons["Mary"];
?>
```

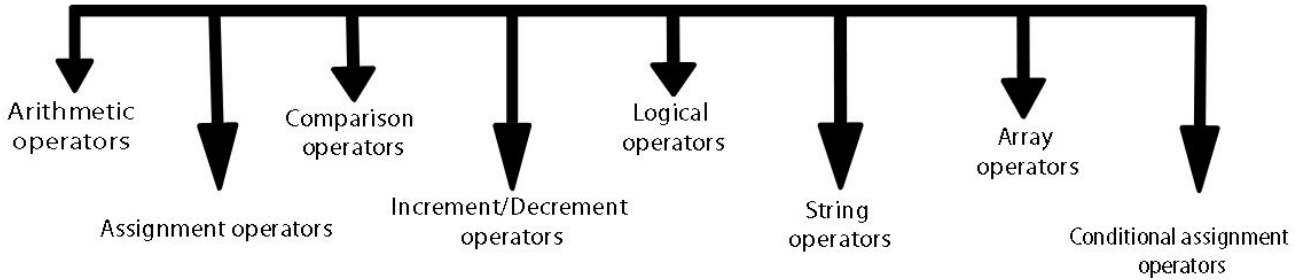


Multidimensional arrays

- These are arrays that contain other nested arrays.
- The advantage of multidimensional arrays is that they allow us to group related data together.

```
<?php
      $movies =array(
      "comedy" => array("Pink Panther", "John English", "See no evil hear no evil"),
      "action" => array("Die Hard", "Expendables"),
      "epic" => array("The Lord of the rings"),
      "Romance" => array("Romeo and Juliet")
      );
      print_r($movies);
?>
```

Operators



Prepared By: Soniya Suthar

Arithmetic Operators

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

Assignment Operators

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

Comparison Operators

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

Increment/Decrement Operators

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one
--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

Logical Operators

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

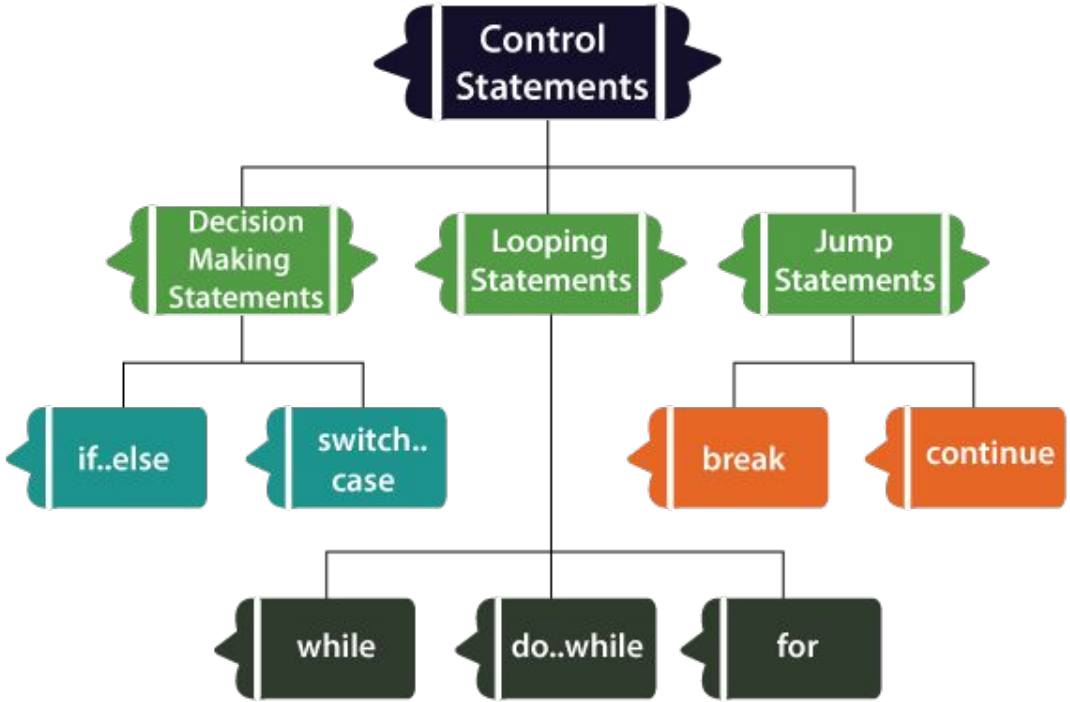
String Operators

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

Array Operators

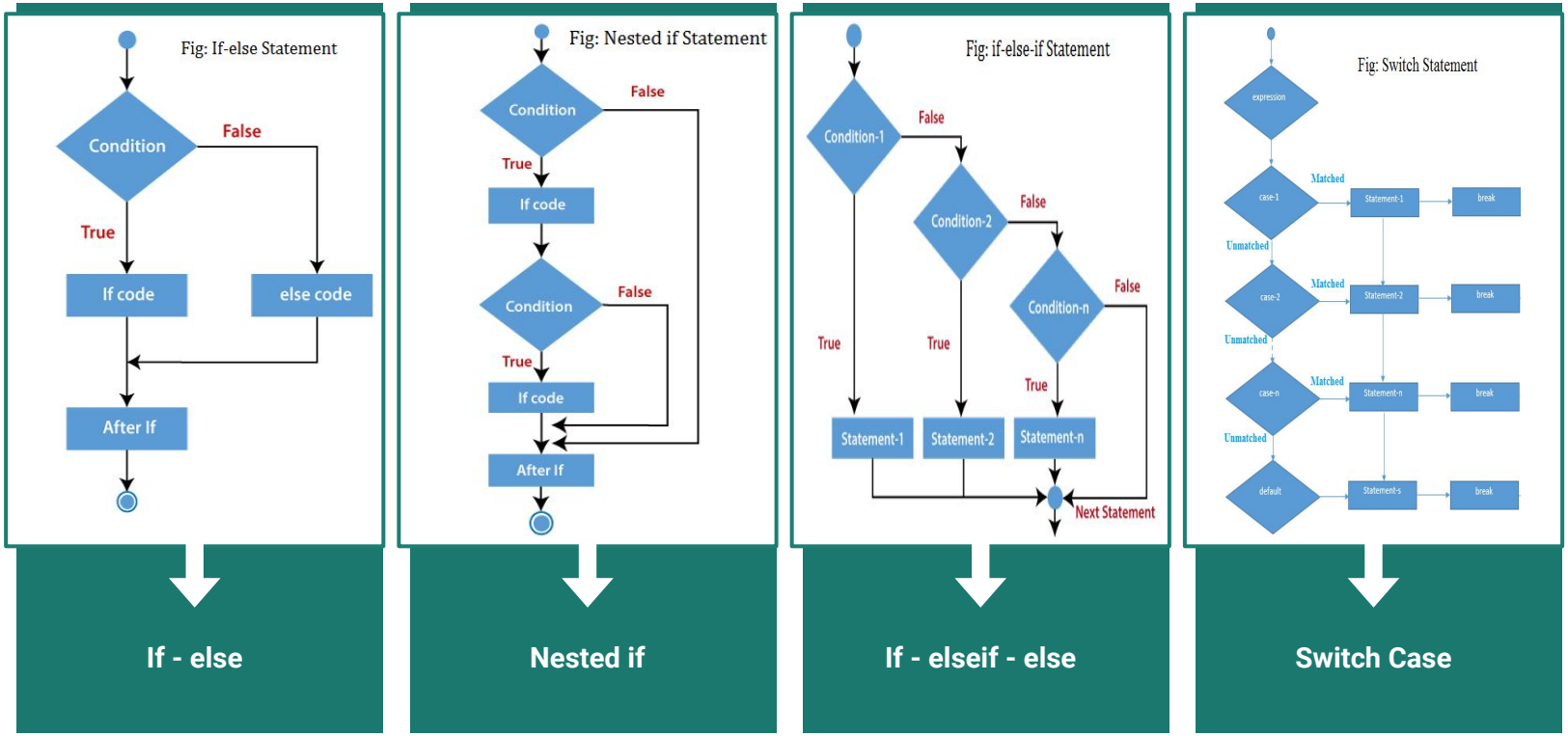
Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

Prepared By: Soniya Suthar



Prepared By: Soniya Suthar

Decision Making & Branching



Prepared By: Soniya Suthar

If - else Statement

```
<?php
      $num=12;
      if($num%2==0){
            echo "$num is even number";
      }
      else{
      echo "$num is odd number";
      }
?>
```

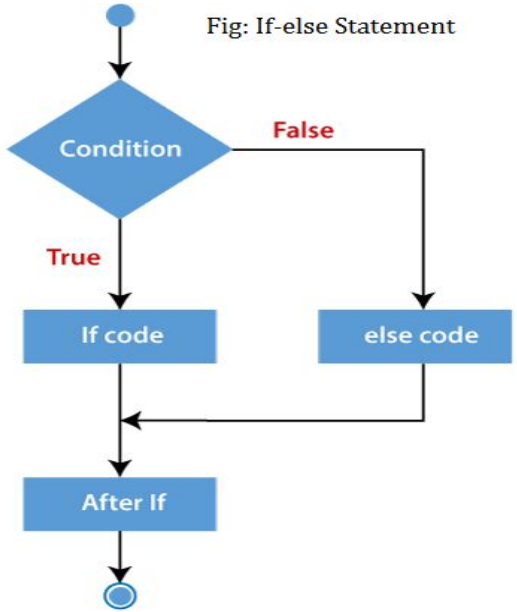
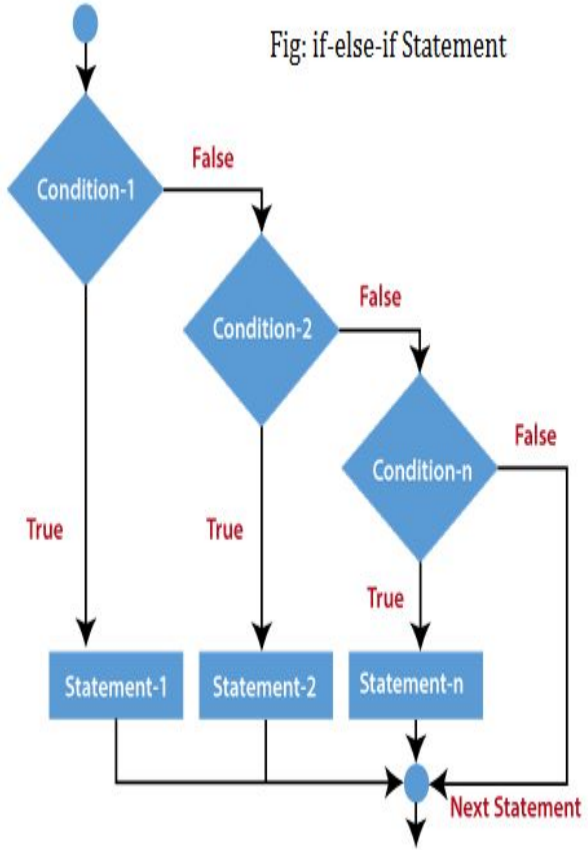


Fig: If-else Statement

If - elseif - else Statement

```php
<?php
    $marks=69;
    if ($marks<33){
        echo "fail";
    }
    else if ($marks>=34 && $marks<50) {
        echo "D grade";
    }
    else if ($marks>=50 && $marks<65) {
       echo "C grade";
    }
    else if ($marks>=65 && $marks<80) {
        echo "B grade";
    }
    else if ($marks>=80 && $marks<90) {
        echo "A grade";
    }
    else if ($marks>=90 && $marks<100) {
        echo "A+ grade";
    }
   else {
        echo "Invalid input";
    }
?>
```



Nested if Statement

```
<?php
        $age = 23;
        $nationality = "Indian";
        if ($nationality == "Indian")
        {
              if ($age >= 18) {
                 echo "Eligible to give vote";
               }
              else {
                 echo "Not eligible to give vote";
               }
        }
?>
```

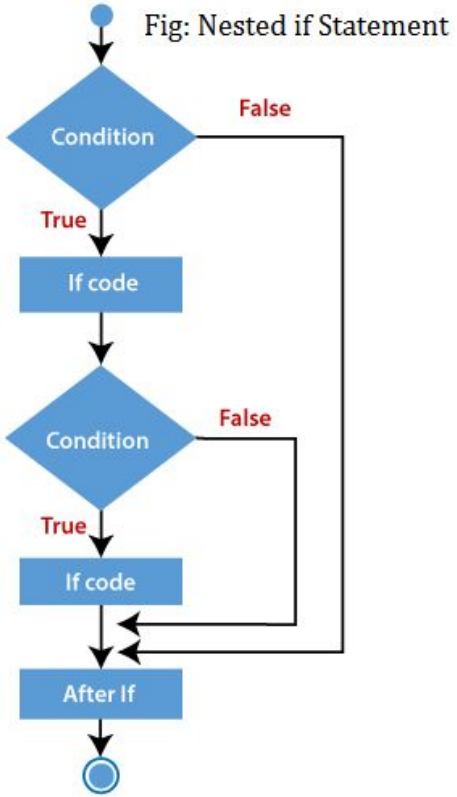


Fig: Nested if Statement

Switch Case Statement

```
<?php
      $num=20;
      switch($num){
            case 10:echo("number is equals to 10");
                  break;
            case 20:echo("number is equal to 20");
                  break;
            case 30:echo("number is equal to 30");
                  break;
            default:echo("number is not equal to 10, 20 or 30");
      }
?>
```

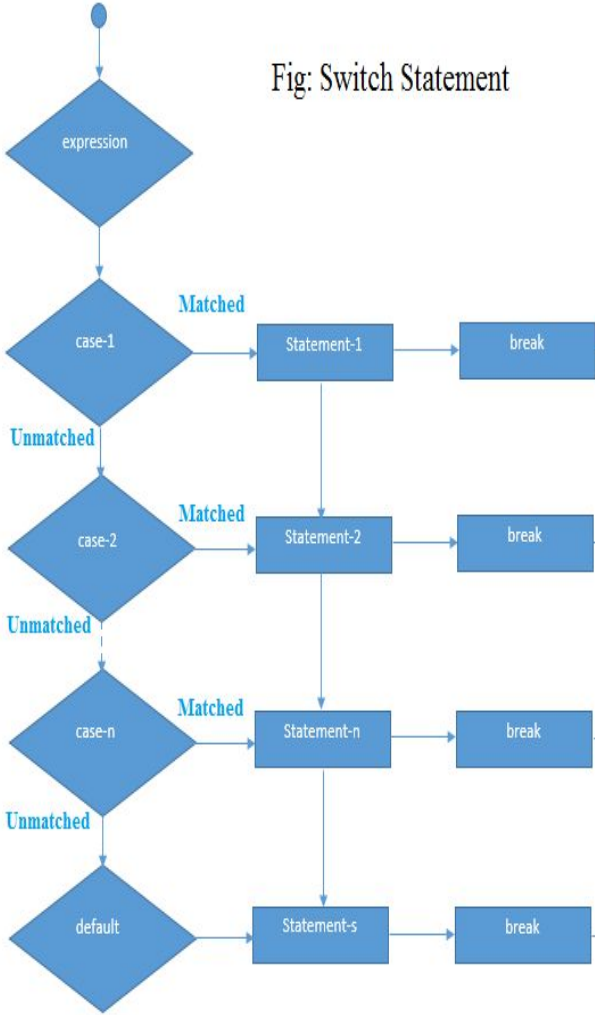
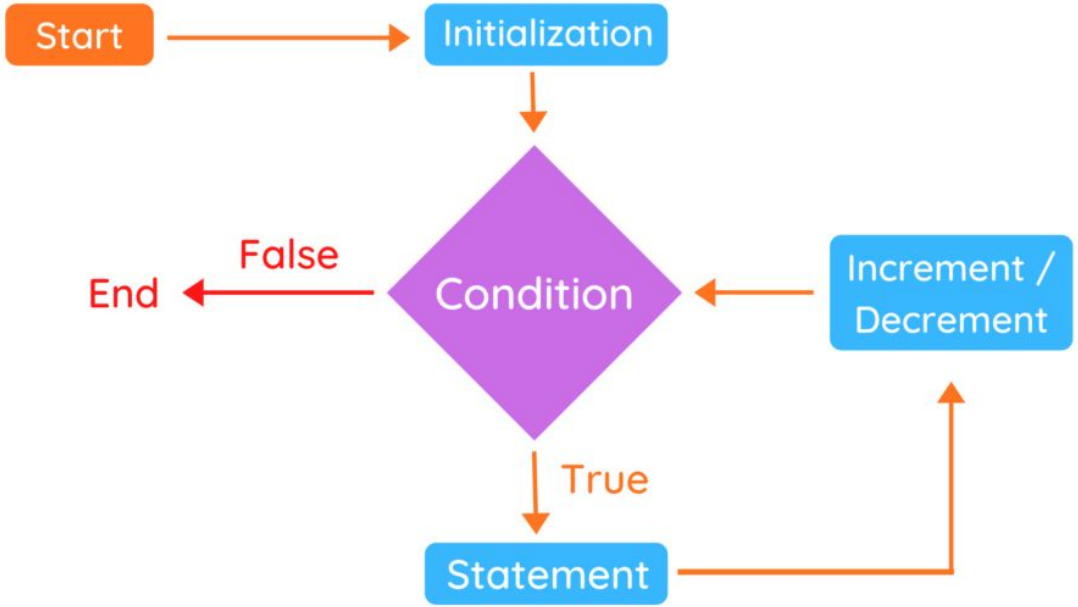
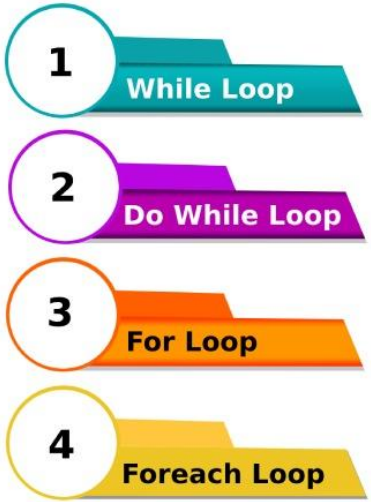


Fig: Switch Statement

Prepared By: Soniya Suthar

Decision Making & Looping

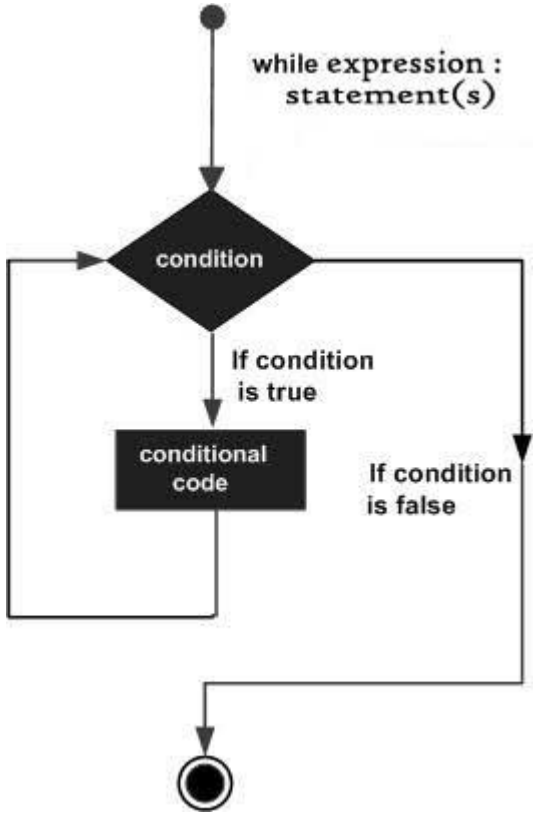




Prepared By: Soniya Suthar

While Loop

```php
<?php
	$x = 1;

	while($x <= 5) {
		echo "The number is: $x <br>";
		$x++;
	}
?>
```



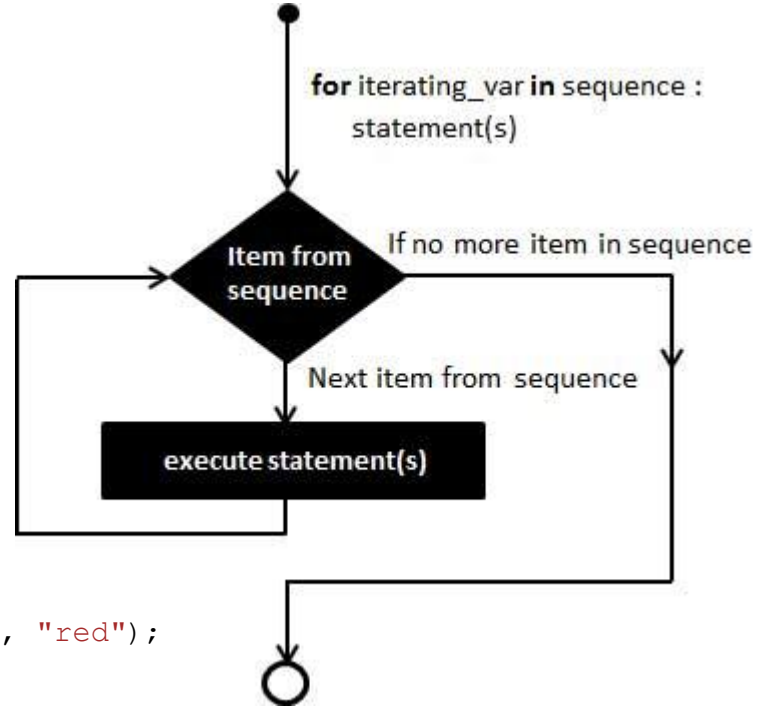
Do While Loop

```php
<?php
	$x = 1;

	do {
		echo "The number is: $x <br>";
		$x++;
	} while ($x <= 5);
?>
```

Prepared By: Soniya Suthar

For Loop

```php
<?php
	for ($x = 0; $x <= 10; $x++) {
		echo "The number is: $x <br>";
	}
?>
```

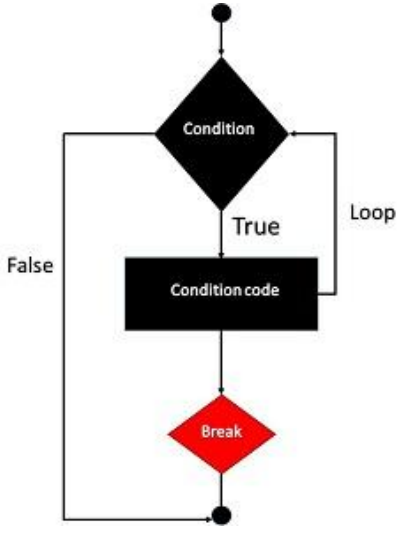


For each Loop

```php
<?php
	$colors = array("blue", "black", "cyan", "red");

	foreach ($colors as $value) {
		echo "$value <br>";
	}
?>
```

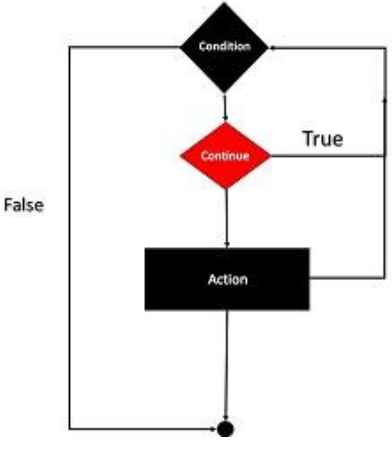
Break Statement

```php
<?php
    for ($x = 0; $x < 10; $x++) {
        if ($x == 4) {
            break;
        }
        echo "The number is: $x <br>";
    }
?>
```



Continue Statement

```php
<?php
    for ($x = 0; $x < 10; $x++) {
        if ($x == 4) {
            continue;
        }
        echo "The number is: $x <br>";
    }
?>
```



Prepared By: Soniya Suthar

Prepared By: Soniya Suthar