Agenda:

- Predefined Functions
- Variable Scopes

String Related Functions - strlen

- Gets the length of a string

```php
<?php
    $str = "Personnel";
    echo strlen($str); // The output is 9
```

- The function has only 1 argument, which is the string value
String Related Functions – strpos

- Finds the position of the 1st occurrence of a character or a substring in a string
- The function takes 2 or 3 arguments
  - 1st argument is the string value
  - 2nd argument is a character or a portion of a string you want to search for
  - 3rd argument, which is optional, is a position of where you want your search to begin

String Related Functions – strpos

- An example of using strpos
  ```php
  <php
  $str = "Personal";
  strpos($str, 'e');
  /* The output is 1 because the function looks for the first occurrence of letter 'e' and the counting starts from 0 */
  ?>
  
  - If you want to get a position of the 2nd 'e', then you need to specify the search to start after the 1st 'e' by providing 2 as the 3rd argument

String Related Functions – strpos

- Another example of using strpos
  ```php
  <php
  $str = "the year ended is the best year";
  echo strpos($str, 'e', 10);
  /* The output is 18 because the 1st letter of the portion you are searching for begins at position 10 */
  ?>
  
  - In this example, any number below 19 that you specify as the 3rd argument will not have an effect
String Related Functions - substr

• This function takes 2 or 3 arguments and returns a portion of a string

```php
$str = "Tomato";
echo substr($str, 3); // This is similar to substr($str, 3)
// The output is on. The output will be so if you add .
```

– The 1st argument is string value, the 2nd specifies where the portion should begin and the 3rd specified the length of the portion
– If the 3rd argument is not specified, then the rest of the portion will include the rest of the word

String Related Functions – trim

• Removes whitespace from the beginning and end of a string
• The function accepts one argument, which is the string value to be trimmed

```php
$str = " Hi pilupilu ";
echo "Hello"
echo trim($str); // trim function
echo ", how are you today?";
// The output is "Hi pilupilu, how are you today?"
```

String Related Functions – ltrim

• Removes whitespace from the beginning of a string
• The function accepts one argument, which is the string value to be trimmed

```php
$str = " Hi pilupilu ";
echo "Hello"
echo ltrim($str); // ltrim function
echo ", how are you today?"
// The output is "Hello Hi pilupilu , how are you today?"
```
String Related Functions – rtrim

- Removes whitespace from the end of a string
- The function accepts one argument, which is the string value to be trimmed

```php
<?php
    $str = " Mylupilo ";
    echo "Hello";
    echo rtrim($str); //String function
    echo "; How are you today?";
    //The output is "Hello Mylupilo, how are you today?"
?>
```

String Related Functions – strtolower

- Converts a string to a lower cases
- The function accepts one argument, which is the string value to be converted

```php
<?php
    $str = "MILUPILU";
    echo strtolower($str);
    //The output is 'milupilu'
?>
```

String Related Functions – strtoupper

- Converts a string to a upper cases
- The function accepts one argument, which is the string value to be converted

```php
<?php
    $str = "milupilo";
    echo strtoupper($str);
    //The output is 'MILUPILU'
?>
```
String Related Functions – str_replace

- Replaces all occurrence of the search substring with the specified replacement
- The function accepts three arguments;
  - 1st argument is a character or substring to be replaced
  - 2nd argument is a replacement
  - 3rd argument is the string value

String Related Functions – str_replace

- An example of using str_replace

```php
<?php
  $str = "mpilupulu";
  echo str_replace('i', 'a', $str);
  // The output is 'mpalupalu'
?>
```

Date Display

- Use date() function to display date and time
  ```php
  $date = date("d/m/Y h:i A");
  echo $date;
  //output: 03/01/2016 05:46 PM
  ```
- To display just the date;
  ```php
  $date = date("d/m/Y");
  echo $date;
  //output: 03/01/2016
  ```
Date Display

- To display just the time:
  ```php
  $date = date("h:i A");
  echo $date;
  //output: 05:46 PM
  ```

Date Display

- To display just a year (four digits):
  ```php
  $date = date("Y");
  echo $date;
  //output: 2016
  ```

- To display just a year (two digits):
  ```php
  $date = date("y");
  echo $date;
  //output: 16
  ```

Date Display

- To display just a month (first three letters):
  ```php
  $date = date("M");
  echo $date;
  //output: Jan
  ```

- To display just a month (full):
  ```php
  $date = date("F");
  echo $date;
  //output: January
  ```
Date Display

- To display just a day (first three letters);
  ```php
  $date = date("D");
  echo $date;
  //output: Sun
  ```

- To display just a day (full);
  ```php
  $date = date("l");
  echo $date;
  //output: Sunday
  ```

Variable Scopes

- Variables can be declared inside and outside of a function

```php
<?php
$price = 10000;  // $price is declared outside the function
echo VAT($price);
function VAT($a) {  // $a and $result are declared inside the function
    $result = $a * 0.18;
    return $result;
}
?>
```

Variable Scopes: Global

- Variables declared outside a function have a **global scope**
  - Values from such variables can only be accessed outside a function
  - If $a is declared outside function1, then the value stored in $a can only be used outside function1 or any other function
Variable Scopes: Global

- Values stored in variables with a global scope can only be accessed outside a function

```php
<?php
$a = 10; // Global variable
echo $a; /* No error will be raised because $a is a global variable */
?>
```

Variable Scopes: Global

- Values from variables with a global scope cannot be accessed from within a function

```php
<?php
$name = "Mpiupiliu";

function display(){
    echo $name; /* This will raise an error because a processor refers to $name, which has not been assigned any value */
}
display();
?>
```

Variable Scopes: Global

- The error from the previous example can be resolved by assigning a value to $name

```php
<?php
$name = "Mpiupiliu";

function display(){
    echo $name; /* This will display "Mpiupiliu" because Mpiupiliu is stored in a global variable, which cannot be accessed from the function */
}
display();
?>
```

- This means same name can be used to declare different variables inside and outside a function
Variable Scopes: Local

- Variables declared inside a function has a **local scope**
  - Values of such variables can only be used within that function
  - If $a$ is declared in function1, then a value stored in $a$ cannot be used outside function1
  - Such variables are created when a function is called and deleted when execution of the function is completed

Variable Scopes: Local

- An example showing inappropriate reference to a local variable

```php
<?php

function add() {
    $a = 10 + 10; // Local variable
    add();
}

echo $a;  // This will raise an error because $a is a local variable */
```

Variable Scopes: Local (global)

- Naturally, values from global variables cannot be accessed from within a function

```php
<?php

$name = "Mphupulu";

function display() {
    echo $name;  // This will raise an error because a processor refers to $name, which has not been assigned any value */
}

display();
```
Variable Scopes: Local (global)

• To access such values from within a function, the *global* keyword is used
  – The keyword should be used inside a function before the global variables that you wish to access their values from within the function
  – Since the keyword is used to refer to global variables, no assignment operator should be used along with the keyword

Variable Scopes: Local (global)

• An example of using the global keyword
  – By declaring $name with the global keyword, any reference to $name will refer to the value stored in the global variable

```
<?php
$name = "$pilupils";
function display(){
  global $name;
  echo $name;
}
display(); // This will display $pilupils
```

Variable Scopes: Local (global)

• An example of using global variable inside a nested function

```
<?php
Sa = 10;
function outer(){
  global Sa;
  function inner(){
    global Sa;
    Sa = Sa;
  }
  inner(); /* Call for the nested function that modifies $a by adding 2 */
  echo Sa; /* Call for outer function
```
Variable Scopes: Local (global)

• An example showing inappropriate reference to a global variable from a nested function

```php
<?php
  function outer(){
    global $a;
    $a = 10;
    inner(); // Call for the nested function that modifies $a by adding 2 */
  }
  echo $a; // Call for outer function
  outer();
?>
```

Variable Scopes: Local (global)

• The example in the previous slide will raise an error because:
  – To a nested function, a global variable is a variable declared as global inside the function that the nested function is defined
  – Thus, the nested function in the example is referring to a global variable that does not exist i.e variable that has not been defined in the parent function

Variable Scopes: Local (static)

• Local variables are deleted when the execution of a function is completed

```php
<?php
  function display(){
    $a = 10;
    return $a;
  }
  echo display();  // This will display 10
  echo display();  // This will display 10
  display();       // This will display 10
?>
```
Variable Scopes: Local (static)

- To stop local variables from being deleted, the `static` keyword is used
  - The keyword should be used before local variables you wish to retain their values
  - Unlike the global keyword, the assignment operator (=) can be used along with the static keyword
  - Such variables continue to store values even after the execution of a function is completed

Variable Scopes: Local (static)

- An example of using static keyword

  ```php
  function display()
  {
  static $a = 10;
  $a++;
  return $a;
  }
  echo display();"<br />"; // This will display 11
  echo display();"<br />"; // This will display 12
  echo display(); // This will display 13
  }
  ```

- Every time the function is called, $a stores a value it was last assigned when the function was executed