Unit – 1 PHP Fundamentals

1.1 Concept of PHP and Introduction

The term PHP is an acronym for *PHP: Hypertext Preprocessor*. PHP is a server-side scripting language designed specifically for web development. It is open-source, which means it is free to download and use. It is very simple to learn and use. The files have the extension ".php".

PHP was created by **Rasmus Lerdorf in 1994,** but it was appeared in the market under 1995. Some important points need to be noticed about PHP are as followed:

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

Why PHP?

PHP can actually do anything related to server-side scripting or more popularly known as the backend of a website. For example, PHP can receive data from forms, generate dynamic page content, can work with databases, create sessions, send and receive cookies, send emails, etc. There are also many hash functions available in PHP to encrypt users' data, which makes PHP secure and reliable to be used as a server-side scripting language. So these are some of PHP's abilities that make it suitable to be used as a server-side scripting language. You will get to know more of these abilities in further tutorials.

Even if the above abilities do not convince you of PHP, there are some more features of PHP. PHP can run on all major operating systems like Windows, Linux, Unix, Mac OS X, etc. Almost all of the major servers available today like Apache supports PHP. PHP allows using a wide range of databases. And, the most important factor is that it is free to use and download and anyone can download PHP from its official source: www.php.net.

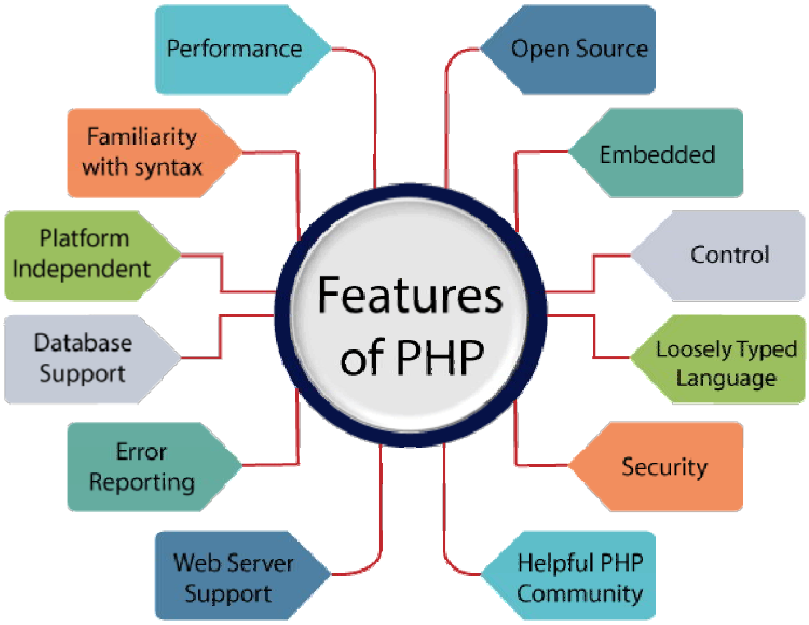
List of PHP Framework: https://www.temok.com/blog/php-framework-list/

PHP Features

PHP is very popular language because of its simplicity and open source. There are some important features of PHP given below:

Performance:

PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.



Open Source:

PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

Familiarity with syntax:

PHP has easily understandable syntax. Programmers are comfortable coding with it.

Embedded:

PHP code can be easily embedded within HTML tags and script.

Platform Independent:

PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

Database Support:

PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

Error Reporting -

PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

Loosely Typed Language:

PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

Web servers Support:

PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

Security:

PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

Control:

Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

1.2 PHP Syntax: Variable, Constants, echo and print commands

The structure which defines PHP, is called PHP syntax.

The PHP script is executed on the server and the HTML result is sent to the browser. It can normally have HTML and PHP tags. PHP or Hypertext Preprocessor is a widely used open-source general-purpose scripting language and can be embedded with HTML. PHP files are saved with the ".php" extension. PHP scripts can be written anywhere in the document within PHP tags along with normal HTML.

Escaping To PHP:

Writing the PHP code inside ***<?php ….?>*** is called Escaping to PHP.

The mechanism of separating a normal HTML from PHP code is called the mechanism of Escaping To PHP. There are various ways in which this can be done. Few methods are already set by default but in order to use few others like Short-open or ASP-style tags, we need to change the configuration of the ***php.ini*** file. These tags are used for embedding PHP within HTML. There are **4 such tags** available for this purpose.

Canonical PHP Tags:

The script **starts with <?php and ends with ?>**. These tags are also called 'Canonical PHP tags'. Everything outside of a pair of opening and closing tags is ignored by the PHP parser. The open and closing tags are called **delimiters**. Every PHP command ends with a semi-colon (;). Let's look at the *hello world* program in PHP.

Code	**Output:**
<?php *# This indicates a comment in PHP* *echo "Hello, world!";* *?>*	Hello, world!

SGML or Short HTML Tags:

These are the shortest option to initialize a PHP code. The script **starts with <? and ends with ?>.** This will only work by **setting the *short_open_tag* setting in the *php.ini* file to 'on'**.

Code	**Output:**
<? *# before using this type of tag one hould keep the short_open_tag value to 'ON' from php.ini* *echo "Hello, world!";* *?>*	Hello, world!

HTML Script Tags:

These are implemented using script tags. This syntax is **removed in PHP 7.0.0.** So its no more used.

Code	**Output:**
<script language="php"> *echo "hello world!";* *</script>*	hello world!

ASP Style Tags:

To use this we need to set the configuration of the *php.ini* file. These are used by Active Server Pages to describe code blocks. These tags start with <% and end with %>.

Code	**Output:**
<% *# Can only be written if setting is turned onto allow %* *echo "hello world";* *%>*	hello world

Variables

In PHP, a variable is declared using a **$ sign** followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax

```
$variablename=value;
```

PHP variable declaration Rules

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

Example:

In the following example we are able to see that how we can declare different types of variables in PHP.

Code:

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

Output:

```
string is: hello string
integer is: 200
float is: 44.6
```

PHP Variable for holding Sum of two variables

Code:

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

Output:

```
11
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

Code:

```
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Output:

```
My car is red
Notice: Undefined variable: COLOR in
C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in
C:\wamp\www\variable.php on line 5
My boat is
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

PHP Variable Scope

The scope of a variable is defined as its range in the program under which it can be accessed. In other words, "**The scope of a variable is the portion of the program within which it is defined and can be accessed**."

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

Local variable

The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an example:

Example

Code:

```
<?php
  function local_var()
  {
    $num = 45;  //local variable
    echo "Local variable declared inside the function is: ". $num;
  }
  local_var();
?>
```

Output:

```
Local variable declared inside
the function is: 45
```

Code:

```
<?php
  function mytest()
  {
    $lang = "PHP";
    echo "Web development language: " .$lang;
  }
  mytest();
  //using $lang (local variable) outside the function will generat
e an error
  echo $lang;
?>
```

Output:

```
Web development language:
PHP
Notice: Undefined variable:
lang in
D:\xampp\htdocs\program\p
3.php on line 28
```

Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

Example:

Code:

```
<?php
  $name = "Sanaya Sharma";      //Global Variable
  function global_var()
  {
    global $name;
    echo "Variable inside the function: ". $name;
    echo "</br>";
  }
  global_var();
  echo "Variable outside the function: ". $name;
?>
```

Output:

```
Variable inside the function: Sanaya
Sharma
Variable outside the function: Sanaya
Sharma
```

Note: Without using the global keyword, if you try to access a global variable inside the function, it will generate an error that the variable is undefined.

Example:

Code:

```
<?php
$name = "Sanaya Sharma";      //global variable
  function global_var()
  {
    echo "Variable inside the function: ". $name;
echo "</br>";
```

Output:

```
Notice: Undefined variable: name in
D:\xampp\htdocs\program\p3.php
on line 6
Variable inside the function:
    }
    global_var();
  ?>
```

Using $GLOBALS instead of global

Another way to use the global variable inside the function is predefined $GLOBALS array.

Example:

Code:

```
  <?php
    $num1 = 5;     //global variable
    $num2 = 13;    //global variable
    function global_var()
    {
        $sum = $GLOBALS['num1'] + $GLOBALS['num2'];
        echo "Sum of global variables is: " .$sum;
    }
    global_var();
  ?>
```

Output:

```
Sum of global variables is: 18
```

If two variables, local and global, have the same name, then the local variable has higher priority than the global variable inside the function.

Example:

Code:

```
  <?php
    $x = 5;
    function mytest()
    {
      $x = 7;
      echo "value of x: " .$x;
    }
    mytest();
  ?>
```

Output:

```
Value of x: 7
```

Note: local variable has higher priority than the global variable.

Static variable

It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.

Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope. Understand it with the help of an example:

Example:

Code:

```
<?php
  function static_var()
  {
    static $num1 = 3;      //static variable
    $num2 = 6;         //Non-static variable
    //increment in non-static variable
    $num1++;
    //increment in static variable
    $num2++;
    echo "Static: " .$num1 ."</br>";
    echo "Non-static: " .$num2 ."</br>";
  }

//first function call
  static_var();

  //second function call
  static_var();
?>
```

Output:

```
Static: 4
Non-static: 7
Static: 5
Non-static: 7
```

You have to notice that $num1 regularly increments after each function call, whereas $num2 does not. This is why because $num1 is not a static variable, so it freed its memory after the execution of each function call.

PHP $ and $$ Variables

The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

Example:

Code:

```
<?php
$x = "abc";
$$x = 200;
echo $x."<br/>";
echo $$x."<br/>";
echo $abc;
?>
```

Explanation:

In the above example, we have assigned a value to the variable **x** as **abc**. Value of reference variable **$$x** is assigned as **200**.

Code:

```
<?php
 $x="U.P";
```

Explanation:

In the above example, we have assigned a value to the variable **x** as **U.P**. Value of reference

$$x="Lucknow"; echo $x. " "; echo $$x. " "; echo "Capital of $x is " . $$x; ?>	variable **$$x** is assigned as **Lucknow.**
Code: <?php $name="Cat"; ${$name}="Dog"; ${${$name}}="Monkey"; echo $name. " "; echo ${$name}. " "; echo $Cat. " "; echo ${${$name}}. " "; echo $Dog. " "; ?>	**Explanation:** In the above example, we have assigned a value to the variable name **Cat**. Value of reference variable **${$name}** is assigned as **Dog** and **${${$name}}** as **Monkey**.

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

1. **Using define() function**
2. **Using const keyword**

Constants are similar to the variable except once they defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. **For example**, it can be started with a letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

Note: Unlike variables, constants are automatically global throughout the script.

PHP constant: define()

Use the define() function to create a constant. It defines constant at run time.

Syntax:

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

Explanation:

name: It specifies the constant name.

value: It specifies the constant value.

case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

Examples:

Code: **<?php** define("MESSAGE","Hello PHP"); echo MESSAGE; **?>**	**Output:** Hello PHP

Code:
<?php
// **Create a constant with case-insensitive name:**
define("MESSAGE","Hello PHP",true);//not case sensitive
echo MESSAGE, "**</br>**";
echo message;   **?>**
	Output:
	Hello PHP
	Hello PHP

Code:
<?php
define("MESSAGE","Hello PHP",false);//case sensitive
echo MESSAGE;
echo message;
?>
	Output:
	Hello PHP
	Notice: Use of undefined constant message - assumed 'message'
	In…..

PHP constant: const keyword

PHP introduced a keyword **const** to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are **case-sensitive**.

Example:

Code: **<?php** const MESSAGE="Hello const by PHP"; echo MESSAGE; **?>**	**Output:** Hello const by PHP

Constant() function

There is another way to print the value of constants using constant() function instead of using the echo statement.

Syntax

```
constant (name)
```

Example:

Code:
<?php
   define("MSG", "WelCome");
   echo MSG, "**</br>**";
   echo constant("MSG");
   //both are similar
?>
	Output:
	WelCome
	WelCome

Constant vs Variables

Constant	Variables
Once the constant is defined, it can never be redefined.	A variable can be undefined as well as redefined easily.
A constant can only be defined using define() function. It cannot be defined by any simple assignment.	A variable can be defined by simple assignment (=) operator.
There is no need to use the dollar ($) sign before constant during the assignment.	To declare a variable, always use the dollar ($) sign before the variable.
Constants do not follow any variable scoping rules, and they can be defined and accessed anywhere.	Variables can be declared anywhere in the program, but they follow variable scoping rules.
Constants are the variables whose values can't be changed throughout the program.	The value of the variable can be changed.
By default, constants are global.	Variables can be local, global, or static.

PHP Echo

PHP echo is a language construct, not a function. Therefore, you don't need to use parenthesis with it. But if you want to use more than one parameter, it is required to use parenthesis.

Syntax:

```
void echo ( string $arg1 [, string $... ] )
```

PHP echo statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc.

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses: echo(), and echo.
- echo does not return any value.
- We can pass multiple strings separated by a comma (,) in echo.
- echo is faster than the print statement.

PHP echo: printing string

Code:	**Output:**
<?php echo "Hello by PHP echo"; **?>**	Hello by PHP echo

PHP echo: printing multi line string

Code:	**Output:**
<?php echo "Hello by PHP echo	Hello by PHP echo this is multi line text printed by PHP echo statement

this is multi line text printed by PHP echo statement "; **?>**	

PHP echo: printing escaping characters

Code: **<?php** echo "Hello escape \"sequence\" characters"; **?>**	**Output:** Hello escape "sequence" characters

PHP echo: printing variable value

Code: **<?php** $msg="Hello PHP"; echo "Message is: $msg"; **?>**	**Output:** Message is: Hello PHP

PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Print statement can be used with or without parentheses: print and print(). Unlike echo, it always returns 1.

Syntax:

```
int print(string $arg)
```

PHP print statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc.

- print is a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than the echo statement.

PHP print: printing string

Code: **<?php** print "Hello by PHP print "; print ("Hello by PHP print()"); **?>**	**Output:** Hello by PHP print Hello by PHP print()

PHP print: printing multi line string

Code:	Output:
<?php print "Hello by PHP print this is multi line text printed by PHP print statement "; **?>**	Hello by PHP print this is multi line text printed by PHP print statement

PHP print: printing escaping characters

Code:	Output:
<?php print "Hello escape \"sequence\" characters by PHP print"; **?>**	Hello escape "sequence" characters by PHP print

PHP print: printing variable value

Code:	Output:
<?php $msg="Hello print() in PHP"; print "Message is: $msg"; **?>**	Message is: Hello print() in PHP

Difference between echo and print

echo

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses.
- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

print

- print is also a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

1.3 Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:Scalar Types (predefined), Compound Types (user-defined), Special Types

PHP Data Types: Scalar Types

It holds only single value. There are 4 scalar data types in PHP.

- **PHP Boolean**

Booleans are the simplest data type works like switch. It holds only two values: TRUE (1) or FALSE (0). It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

Example:

```
<?php
  if (TRUE)
    echo "This condition is TRUE.";
  if (FALSE)
    echo "This condition is FALSE.";
?>
```

Output:

```
This condition is TRUE.
```

- **PHP Integer**

Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.

Rules for integer:

- An integer can be either positive or negative.
- An integer must not contain decimal point.
- Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
- The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

Example:

```
<?php
  $dec1 = 34;
  $oct1 = 0243;
  $hexa1 = 0x45;
  echo "Decimal number: " .$dec1. "</br>";
  echo "Octal number: " .$oct1. "</br>";
  echo "HexaDecimal number: " .$hexa1. "</br>";
?>
```

Output:

```
Decimal number: 34
Octal number: 163
HexaDecimal number: 69
```

- **PHP Float**

A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

Example:

```
<?php
  $n1 = 19.34;
  $n2 = 54.472;
  $sum = $n1 + $n2;
  echo "Addition of floating numbers: " .$sum;  ?>
```

Output:

```
Addition of floating numbers: 73.812
```

- **PHP String**

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within single quotes or in double quotes. But both are treated differently. To clarify this, see the example below:

Example:

```
<?php
  $company = "Javatpoint";
  //both single and double quote statements will treat different
  echo "Hello $company";
  echo "</br>";
  echo 'Hello $company';
?>
```

Output:

```
Hello Javatpoint
Hello $company
```

PHP Data Types: Compound Types

- **PHP Array**

An array is a compound data type. It can store multiple values of same data type in a single variable.

Example:

```
<?php
  $bikes = array ("Royal Enfield", "Yamaha", "KTM");
  var_dump($bikes);   //the var_dump() function returns the datatype and values
  echo "</br>";
  echo "Array Element1: $bikes[0] </br>";
  echo "Array Element2: $bikes[1] </br>";
  echo "Array Element3: $bikes[2] </br>";
?>
```

Output:

```
array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=> string(3) "KTM" }
Array Element1: Royal Enfield
Array Element2: Yamaha
Array Element3: KTM
```

- **PHP object**

Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.

Example:

```
<?php
   class bike {
       function model() {
```

Output:

```
Bike Model: Royal Enfield
```

```
            $model_name = "Royal Enfield";
            echo "Bike Model: " .$model_name;
           }
       }
       $obj = new bike();
       $obj -> model();
    ?>
```

PHP Data Types: Special Types

- **PHP Resource**

Resources are not the exact data type in PHP. Basically, these are used to store some function calls or references to external PHP resources. For example - a database call. It is an external resource.

- **PHP Null**

Null is a special data type that has only one value: NULL. There is a convention of writing it in capital letters as it is case sensitive.

The special type of data type NULL defined a variable with no value.

Example:

```
<?php
  $nl = NULL;
  echo $nl;   //it will not give any output
?>
```

1.4 Operators, Arrays, Sorting Arrays

PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. For example:

$num=10+20;//+ is the operator and 10,20 are operands

In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

- **Arithmetic Operators**

The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

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

The exponentiation (**) operator has been introduced in PHP 5.6.

- **Assignment Operators**

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

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

- **Bitwise Operators**

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise 0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

- **Comparison Operators**

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

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

- **Incrementing/Decrementing Operators**

The increment and decrement operators are used to increase and decrease the value of a variable.

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**

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

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**

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

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**

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

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

- **Type Operators**

The type operator **instanceof** is used to determine whether an object, its parent and its derived class are the same type or not. Basically, this operator determines which certain class the object belongs to. It is used in object-oriented programming.

```
<?php
  //class declaration
  class Developer
  {}
  class Programmer
  {}
  //creating an object of type Developer
  $charu = new Developer();

  //testing the type of object
  if( $charu instanceof Developer)
  {
    echo "Charu is a developer.";
  }
  else
  {
    echo "Charu is a programmer.";
  }
  echo "</br>";
  var_dump($charu instanceof Developer);     //It will return true.
```

Output:

```
Charu is a developer.
bool(true) bool(false)
```

var_dump($charu instanceof Programmer);   //It will return fal
se.
?>	

- **Execution Operators**

PHP has an execution operator **backticks (``)**. PHP executes the content of backticks as a shell command. Execution operator and **shell_exec()** give the same result.

Operator	Name	Example	Explanation
``	backticks	echo `dir`;	Execute the shell command and return the result. Here, it will show the directories available in current folder.

Note: Note that backticks (``) are not single-quotes.

- **Error Control Operators**

PHP has one error control operator, i.e., **at (@) symbol**. Whenever it is used with an expression, any error message will be ignored that might be generated by that expression.

Operator	Name	Example	Explanation
@	At	@file ('non_existent_file')	Intentional file error

- **PHP Operators Precedence**

Operators	Additional Information	Associativity
clone new	clone and new	non-associative
[	array()	left
**	Arithmetic	right
++ -- ~ (int) (float) (string) (array) (object) (bool) @	increment/decrement and types	right
Instanceof	Types	non-associative
!	logical (negation)	right
* / %	Arithmetic	Left
+ - .	arithmetic and string concatenation	Left
<<>>	bitwise (shift)	Left
<<= >>=	Comparison	non-associative
== != === !== <>	Comparison	non-associative
&	bitwise AND	Left
^	bitwise XOR	Left
|	bitwise OR	Left
&&	logical AND	Left
||	logical OR	Left
?:	Ternary	Left
= += -= *= **= /= .= %= &= |= ^= <<= >>= =>	Assignment	Right
And	Logical	Left

Xor	Logical	Left
Or	Logical	Left
,	many uses (comma)	Left

PHP Arrays

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array

Less Code: We don't need to define multiple variables.

Easy to traverse: By the help of single loop, we can traverse all the elements of an array.

Sorting: We can sort the elements of array.

PHP Array Types

There are 3 types of array in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

1st way:

```
$season=array("summer","winter","spring","autumn");
```

2nd way:

```
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
```

Example

<?php $season=array("summer","winter","spring","autumn"); echo "Season are: $season[0], $season[1], $season[2] and $season[3]"; ?>	**Output:** Season are: summer, winter, spring and autumn
<?php $season[0]="summer";	**Output:** Season are: summer, winter,

$season[1]="winter"; $season[2]="spring"; $season[3]="autumn"; echo "Season are: $season[0], $season[1], $season[2] and $season[3]"; ?>	spring and autumn

PHP Associative Array

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

1st way:

```
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
```

2nd way:

```
$salary["Sonoo"]="350000";

$salary["John"]="450000";

$salary["Kartik"]="200000";
```

Example

<?php
$salary=array("Sonoo"=>"350000","John"=>"450000","
Kartik"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>"; ?>
	Output:
	Sonoo salary: 350000
	John salary: 450000
	Kartik salary: 200000

<?php $salary["Sonoo"]="350000"; $salary["John"]="450000"; $salary["Kartik"]="200000"; echo "Sonoo salary: ".$salary["Sonoo"]."<br/>"; echo "John salary: ".$salary["John"]."<br/>"; echo "Kartik salary: ".$salary["Kartik"]."<br/>"; ?>	**Output:** Sonoo salary: 350000 John salary: 450000 Kartik salary: 200000

PHP Multidimensional Array

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

Example

<?php $emp = array (array(1,"sonoo",400000), array(2,"john",500000), array(3,"rahul",300000)	**Output:** 1 sonoo 400000 2 john 500000 3 rahul 300000

```
  );
 for ($row = 0; $row < 3; $row++) {
  for ($col = 0; $col < 3; $col++) {
    echo $emp[$row][$col]."  ";
  }
  echo "<br/>";
 }
 ?>
```

Sorting PHP Array

Sorting refers to ordering data in an alphabetical, numerical order and increasing or decreasing fashion according to some linear relationship among the data items.Sorting greatly improves the efficiency of searching.

Sorting Functions For Arrays In PHP

1. sort() – sorts arrays in ascending order
2. rsort() – sorts arrays in descending order
3. asort() – sorts associative arrays in ascending order, according to the value
4. ksort() – sorts associative arrays in ascending order, according to the key
5. arsort() – sorts associative arrays in descending order, according to the value
6. krsort() – sorts associative arrays in descending order, according to the key

Sort Array in Ascending Order – sort()

The following function sorts the elements of a numerical array in ascending numerical order:

```
<?php
$numbers = array(40, 61, 2, 22, 13);
sort($numbers);

$arrlength = count($numbers);
for($x = 0; $x < $arrlength; $x++) {
  echo $numbers[$x];
  echo "<br>";
}
?>
```

OUTPUT :

```
2
13
22
40
61
```

Sort Array in Descending Order – rsort()

The following function sorts the elements of a numerical array in descending numerical order:

```
<?php
$numbers = array(40, 61, 2, 22, 13);
```

OUTPUT :

```
61
```

```
rsort($numbers);
$arrlength = count($numbers);
for($x = 0; $x < $arrlength; $x++) {
   echo $numbers[$x];
   echo "<br>";
}
?>
```

```
40
22
13
2
```

Sort Array in Ascending Order,According to Value – asort()

The following function sorts an associative array in ascending order, according to the value:

```
<?php
$age = array("ayush"=>"23", "shankar"=>"47",
"kailash"=>"41");
asort($age);
foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>
```

```
OUTPUT :
Key=Ayush,              Value=23
Key=Kailash,            Value=41
Key=Shankar, Value=47
```

Sort Array in Ascending Order, According to Key – ksort()

The following function sorts an associative array in ascending order, according to the key:

```
<?php
$age     =     array("ayush"=>"23",     "shankar"=>"47",
"kailash"=>"41");
ksort($age);
foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>
```

```
OUTPUT :
Key=Ayush,   Value=23
Key=Kailash,  Value=41
Key=Shankar, Value=47
```

Sort Array in Descending Order, According to Value – arsort()

The following function sorts an associative array in descending order, according to the value.

```
<?php
$age     =     array("ayush"=>"23",     "shankar"=>"47",
"kailash"=>"41");
```

```
OUTPUT :

Key=Shankar, Value=47
```

```
arsort($age);
foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}?>
```

```
Key=Kailash,   Value=41
Key=Ayush, Value=23
```

Sort Array in Descending Order, According to Key – krsort()

The following function sorts an associative array in descending order, according to the key.

```
<?php
$age = array("ayush"=>"23", "shankar"=>"47", "kailash"=>"41");
krsort($age);
foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
} ?>
```

```
OUTPUT :

Key=Shankar,     Value=47
Key=Kailash,     Value=41
Key=Ayush, Value=23
```

1.5 Conditional Statements & Loops

PHP If Else

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

PHP If Statement

PHP if statement allows conditional execution of code. It is executed if condition is true.

If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

Syntax

```
if(condition){

//code to be executed

}
```

Example

```
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}
?>
```

Output:

```
12 is less than 100
```

PHP If-else Statement

PHP if-else statement is executed whether condition is true or false.

If-else statement is slightly different from if statement. It executes one block of code if the specified condition is true and another block of code if the condition is false.

Syntax

```
if(condition){
//code to be executed if true
}else{
//code to be executed if false
}
```

Example

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

Output:

```
12 is even number
```

PHP If-else-if Statement

The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

Syntax

```
if (condition1){
//code to be executed if condition1 is true
} elseif (condition2){
//code to be executed if condition2 is true
} elseif (condition3){
//code to be executed if condition3 is true
....
}  else{
//code to be executed if all given conditions are false
}
```

Example

```
<?php
  $marks=69;
  if ($marks<33){
    echo "fail";
  }
  else if ($marks>=34 && $marks<50) {
    echo "D grade";
```

Output:

```
B 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";    }
?>
```

PHP nested if Statement

The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is true.

Syntax

```
if (condition) {
//code to be executed if condition is true
if (condition) {
//code to be executed if condition is true
}
}
```

Example

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

Output:

```
Eligible to give vote
```

PHP Switch Example

```
    <?php
```

Output:

$a = 34; $b = 56; $c = 45;
if ($a < $b) {
if ($a < $c) {
echo "$a is smaller than $b and $c";
}
}
?>
	34 is smaller than 56 and 45

PHP Switch

PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

Syntax

```
switch(expression){

case value1:

 //code to be executed

 break;

case value2:

 //code to be executed

 break;

......

default:

 code to be executed if all cases are not matched;

}
```

Important points to be notice about switch case:

- The default is an optional statement. Even it is not important, that default must always be the last statement.
- There can be only one default in a switch statement. More than one default may lead to a Fatal error.
- Each case can have a break statement, which is used to terminate the sequence of statement.
- The break statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value.
- PHP allows you to use number, character, string, as well as functions in switch expression.
- Nesting of switch statements is allowed, but it makes the program more complex and less readable.
- You can use semicolon (;) instead of colon (:). It will not generate any error.

Example

```
    <?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");
    }
?>
```

Output:

```
number is equal to 20
```

PHP switch statement with character

We will pass a character in switch expression to check whether it is vowel or constant. If the passed character is A, E, I, O, or U, it will be vowel otherwise consonant.

Example

```
<?php
  $ch = 'U';
  switch ($ch)
  {
    case 'a':
      echo "Given character is vowel";
      break;
    case 'e':
      echo "Given character is vowel";
      break;
    case 'i':
      echo "Given character is vowel";
      break;
    case 'o':
      echo "Given character is vowel";
      break;
    case 'u':
      echo "Given character is vowel";
      break;
    case 'A':
      echo "Given character is vowel";
      break;
```

Output:

```
Given character is vowel
```

```
        case 'E':
          echo "Given character is vowel";
          break;
        case 'I':
          echo "Given character is vowel";
          break;
        case 'O':
          echo "Given character is vowel";
          break;
        case 'U':
          echo "Given character is vowel";
          break;
        default:
          echo "Given character is consonant";
          break;
      }
    ?>
```

PHP switch statement with String

PHP allows to pass string in switch expression. Let's see the below example of course duration by passing string in switch case statement.

```
<?php
  $ch = "B.Tech";
  switch ($ch)
  {
    case "BCA":
      echo "BCA is 3 years course";
      break;
    case "Bsc":
      echo "Bsc is 3 years course";
      break;
    case "B.Tech":
      echo "B.Tech is 4 years course";
      break;
    case "B.Arch":
      echo "B.Arch is 5 years course";
      break;
    default:
      echo "Wrong Choice";
      break;
  }
?>
```

Output:

```
B.Tech is 4 years course
```

PHP switch statement is fall-through

PHP switch statement is fall-through. It means it will execute all statements after getting the first match, if break statement is not found.

```
<?php
  $ch = 'c';
  switch ($ch)
  {
    case 'a':
      echo "Choice a";
      break;
    case 'b':
      echo "Choice b";
      break;
    case 'c':
      echo "Choice c";
      echo "</br>";
    case 'd':
      echo "Choice d";
      echo "</br>";
    default:
      echo "case a, b, c, and d is not found";
  }
?>
```

Output:

```
Choice c
Choice d
case a, b, c, and d is not found
```

PHP nested switch statement

Nested switch statement means switch statement inside another switch statement. Sometimes it leads to confusion.

```
<?php
  $car = "Hyundai";
    $model = "Tucson";
    switch( $car )
    {
      case "Honda":
        switch( $model )
        {
          case "Amaze":
              echo "Honda Amaze price is 5.93 -
 9.79 Lakh.";
            break;
          case "City":
              echo "Honda City price is 9.91 -
 14.31 Lakh.";
            break;
        }
        break;
```

Output:

```
Hyundai Tucson price is 22.39 -
32.07 L.
```

```
      case "Renault":
        switch( $model )
        {
          case "Duster":
            echo "Renault Duster price is 9.15 -
 14.83 L.";
            break;
          case "Kwid":
              echo "Renault Kwid price is 3.15 -
 5.44 L.";
            break;
        }
        break;
      case "Hyundai":
        switch( $model )
        {
          case "Creta":
            echo "Hyundai Creta price is 11.42 -
 18.73 L.";
            break;
    case "Tucson":
              echo "Hyundai Tucson price is 22.39 -
 32.07 L.";
            break;
          case "Xcent":
              echo "Hyundai Xcent price is 6.5 -
 10.05 L.";
            break;
        }
        break;
    }
?>
```

PHP Loops

For Loop

PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.

It allows users to put all the loop related statements in one place. See in the syntax given below:

Syntax

```
for(initialization; condition; increment/decrement){
```

```
//code to be executed
}
```

Parameters

The php for loop is similar to the java/C/C++ for loop. The parameters of for loop have the following meanings:

initialization - Initialize the loop counter value. The initial value of the for loop is done only once. This parameter is optional.

condition - Evaluate each iteration value. The loop continuously executes until the condition is false. If TRUE, the loop execution continues, otherwise the execution of the loop ends.

Increment/decrement - It increments or decrements the value of the variable.

Example

```
<?php
for($n=1;$n<=10;$n++){
echo "$n<br/>";
}
?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Example

All three parameters are optional, but semicolon (;) is must to pass in for loop. If we don't pass parameters, it will execute infinite.

```
<?php
  $i = 1;
  //infinite loop
  for (;;) {
    echo $i++;
    echo "</br>";
  }
?>
```

Output:

```
1
2
3
4
.
.
.
```

Example

Below is the example of printing numbers from 1 to 9 in four different ways using for loop.

```
<?php
  /* example 1 */

  for ($i = 1; $i <= 9; $i++) {
  echo $i;
  }
  echo "</br>";
  /* example 2 */
  for ($i = 1; ; $i++) {
    if ($i > 9) {
      break;
    }
    echo $i;
  }
  echo "</br>";
  /* example 3 */

  $i = 1;
  for (; ; ) {
    if ($i > 9) {
      break;
    }
    echo $i;
    $i++;
  }
  echo "</br>";

  /* example 4 */

  for ($i = 1, $j = 0; $i <= 9; $j += $i, print $i, $i++);  ?>
```

Output:

```
123456789
123456789
123456789
123456789
```

PHP Nested For Loop

We can use for loop inside for loop in PHP, it is known as nested for loop. The inner for loop executes only when the outer for loop condition is found **true**.

In case of inner or nested for loop, nested for loop is executed fully for one outer for loop. If outer for loop is to be executed for 3 times and inner for loop for 3 times, inner for loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

Example

```
<?php
for($i=1;$i<=3;$i++){
for($j=1;$j<=3;$j++){
echo "$i   $j<br/>";
}
}
```

Output:

```
1 1
1 2
1 3
2 1
```

```
    ?>
```

```
2 2
2 3
3 1
3 2
3 3
```

PHP For Each Loop

The foreach loop is used to traverse the array elements. It works only on array and object. It will issue an error if you try to use it with the variables of different datatype.

The foreach loop works on elements basis rather than index. It provides an easiest way to iterate the elements of an array.

In foreach loop, we don't need to increment the value.

Syntax

```
foreach ($array as $value) {
   //code to be executed
}
```

```
foreach ($array as $key => $element) {
   //code to be executed
}
```

Example 1:

PHP program to print array elements using foreach loop.

```
<?php
  //declare array
  $season = array ("Summer", "Winter", "Autumn", "Rainy");

  //access array elements using foreach loop
  foreach ($season as $element) {
    echo "$element";
    echo "</br>";
  }
?>
```

Output:

```
Summer
Winter
Autumn
Rainy
```

Example 2:

PHP program to print associative array elements using foreach loop.

```
<?php
  //declare array
  $employee = array (
    "Name" => "Alex",
    "Email" => "alex_jtp@gmail.com",
    "Age" => 21,
    "Gender" => "Male"
  );
```

Output:

```
Name : Alex
Email : alex_jtp@gmail.com
Age : 21
Gender : Male
```

```
    //display associative array element through foreach loop
    foreach ($employee as $key => $element) {
      echo $key . " : " . $element;
      echo "</br>";
    }
  ?>
```

Example 3:

Multi-dimensional array

```
  <?php
    //declare multi-dimensional array
    $a = array();
    $a[0][0] = "Alex";
    $a[0][1] = "Bob";
    $a[1][0] = "Camila";
    $a[1][1] = "Denial";

    //display multi-dimensional array elements through foreach loop
    foreach ($a as $e1) {
      foreach ($e1 as $e2) {
        echo "$e2\n";
      }
    }
  ?>
```

Output:

```
Alex Bob Camila Denial
```

Example 4:

Dynamic array

```
  <?php
    //dynamic array
    foreach (array ('j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't') as $elements) {
      echo "$elements\n";
    }
  ?>
```

Output:

```
j a v a t p o i n t
```

Unit – 2 PHP Functions

2.1 Math Functions, Date and Time Functions, GET and POST Methods

As you know PHP is a sever side script language and is used for creating dynamic web pages. PHP provides a number of built-in math functions which help in performing several operations while dealing with mathematical data. It has nice support for mathematical processing. No installation is required to use these functions.

Math Functions

- **abs()**

This function takes negative value as input and returns the absolute (positive) value of a integer or float number.

Syntax:

```
number abs ( mixed $number )
```

Example:

<?php	**Output**
echo (abs(-7)."<br/>"); // 7 (integer)	7
echo (abs(7)."<br/>"); //7 (integer)	7
echo (abs(-7.2)."<br/>"); //7.2 (float/double)	7.2
?>	

- **ceil()**

This function takes numeric value as argument and returns the next highest integer value by rounding up value if necessary.

Syntax:

```
float ceil ( float $value )
```

Example:

<?php	**Output**
echo (ceil(3.3)."<br/>");// 4	**4**
echo (ceil(7.333)."<br/>");// 8	**8**
echo (ceil(-4.8)."<br/>");// -4	**-4**
?>	

- **floor()**

This function takes numeric value as argument and returns the next lowest integer value (as float) by rounding down value if necessary.

Syntax:

```
float floor ( float $value )
```

Example:

<?php
echo (ceil(3.3)."<br/>");// 4
echo (ceil(7.333)."<br/>");// 8
echo (ceil(-4.8)."<br/>");// -4
?>
	Output
	4
	8
	-4

- **fmod()**

This function takes two arguments as input returns the floating point remainder (modulo) of division of arguments.

Syntax:

float fmod (float $x , float $y);

Example:

<?php $x = 7; $y = 2; echo "Your Given Nos is : $x=7, $y=2" $result ="By using 'fmod()' Function your value is:".fmod($x,$y); echo $result; // $result equals 1, because 2 * 3 + 1 = 7 ?>	**Output** By using 'fmod()' Function your value is:1
<?php $x = 5.7; $y = 1.3; echo "Your Given Nos is : x=5.7, y=1.3"; echo " "."By using 'fmod()' function your value is:".fmod($x, $y); // $r equals 0.5, because 4 * 1.3 + 0.5 = 5.7 ?>	**Output** Your Given Nos is : x=5.7, y=1.3 By using 'fmod()' Function your value is:0.5

- **max()**

The max() function is used to find the highest value from the given numbers or the given array which depends upon the parameter.

Syntax:

max(array_values);

OR

max(value1,value2,value3,value4...);

Example:

<?php $num=max(4,14,3,5,14.2); echo "Your number is =max(4,14,3,5,14.2)".' ';	**Output** Your number is =max(4,14,3,5,14.2)

echo "By using max function Your number is : ".$num; ?>	By using max function Your number is : 14.2
<?php $num=max(.1, .001, .2, -.5); echo "Your number is =max(.1, .001, .2, -.5)".' '; echo "By using max function Your number is : ".$num; ?>	**Output** Your number is =max(.1, .001, .2, -.5) By using max function Your number is : 0.2
<?php $arr= array(110, 20, 52 ,105, 56, 89, 96); echo "Your number is =array(110, 20, 52 ,105, 56, 89, 96)".' '; echo "By using max() function Your number is : ".max($arr); ?>	**Output** Your number is =array(110, 20, 52 ,105, 56, 89, 96) By using max() function Your number is : 110

- **min()**

The min() function is used to find the smallest value from the given numbers or the given array which depends upon the parameter.

Syntax:

```
min(array_values);

OR

min(value1,value2,value3,value4...);
```

Example:

<?php $num=min(4,14,3,5,14.2); echo "Your number is =min(4,14,3,5,14.2)".' '; echo "By using min function Your number is : ".$num; ?>	**Output** Your number is =min(4,14,3,5,14.2) By using min function Your number is : 3
<?php $num=min(.1, .001, .2, -.5); echo "Your number is =min(.1, .001, .2, -.5)".' '; echo "By using min function Your number is : ".$num; ?>	**Output** Your number is =max(.1, .001, .2, -.5) By using max function Your number is : 0.001
<?php $arr= array(110, 20, 52 ,105, 56, 89, 96); echo "Your number is =array(110, 20, 52 ,105, 56, 89, 96)".' '; echo "By using min() function Your number is : ".min($arr); ?>	**Output** Your number is =array(110, 20, 52 ,105, 56, 89, 96) By using max() function Your number is : 20

- **pow()**

The pow() is a PHP mathematic function. It raises the first number to the power of the second number.

Syntax:

```
number pow ( number $base , number $exp )
```

Example:

<?php $num=pow(3, 2); echo "Your number is = pow (3, 2)".' '; echo "By using sqrt function Your number is : ".$num; ?>	**Output** Your number is = pow (3, 2) By using sqrt function Your number is : 9
<?php $num=pow(-5, 2); echo "Your number is = pow (-5, 2)".' '; echo "By using sqrt function Your number is : ".$num; ?>	**Output** Your number is = pow (-5, 2) By using sqrt function Your number is : 25
<?php $num=pow(-3, -3); echo "Your number is = pow (-3, -3)".' '; echo "By using sqrt function Your number is : ".$num; ?>	**Output** Your number is = pow (-3, -3) By using sqrt function Your number is : -0.037037037037037
<?php $num=pow(-3, -1.5); echo "Your number is = pow (-3, -1.5)".' '; echo "By using sqrt function Your number is : ".$num; ?>	**Output** Your number is = pow (-3, -1.5) By using sqrt function Your number is : -NAN

- **rand()**

  The rand() function is used to generate the random integer.

Syntax:

```
Int rand(void)
OR
Int rand(int $min, int $max)
```

Example:

<?php echo "Get Random number by using rand() function: ".(rand( ) . " "); echo "Get Random number by using rand() function: ".(rand( ) . " "); echo "<b>"."Note: Refresh page to get another random value "."<b>"; ?>	**Output** Get Random number by using rand() function: 81627923 Get Random number by using rand() function: 1857469033 Note: Refresh page to get another random value
<?php $num=pow(-5, 2); echo "Your number is = pow (-5, 2)".' ';	**Output** Your number is = pow (-5, 2) By using sqrt function Your number

echo "By using sqrt function Your number is : ".$num; ?>	is : 25
<?php $num=pow(-3, -3); echo "Your number is = pow (-3, -3)".' '; echo "By using sqrt function Your number is : ".$num; ?>	**Output** Your number is = pow (-3, -3) By using sqrt function Your number is : -0.037037037037037
<?php $num=pow(-3, -1.5); echo "Your number is = pow (-3, -1.5)".' '; echo "By using sqrt function Your number is : ".$num; ?>	**Output** Your number is = pow (-3, -1.5) By using sqrt function Your number is : -NAN

- **round()**

The round() function in PHP is used to round a floating-point number. It can be used to define a specific precision value which rounds number according to that precision value.Precision can be also negative or zero.

Syntax:

float round($number, $precision, $mode);

Parameters:

1. **$number**: It is the number which you want to round.
2. **$precision**: It is an optional parameter. It specifies the number of decimal digits to round to. The default value of this parameter is zero.
3. **$mode**: It is an optional parameter. It specifies a constant to specify the rounding mode. The constant can be one of the following:
   - PHP_ROUND_HALF_UP: This mode tells to round up the number specified by parameter *$number* by precision specified by parameter *$precision* away from zero.
   - PHP_ROUND_HALF_DOWN: This mode tells to round up the number specified by parameter *$number* by precision specified by parameter *$precision* towards zero.
   - PHP_ROUND_HALF_EVEN: This mode tells to round up the number specified by parameter *$number* by precision specified by parameter *$precision* towards nearest even value.
   - PHP_ROUND_HALF_ODD: This mode tells to round up the number specified by parameter *$number* by precision specified by parameter *$precision* towards nearest odd value.

Return Value: It returns the rounded value.

Example:

<?php // round to nearest even value	**Output:**

echo(round(7.5,0,PHP_ROUND_HALF_EVEN));
echo "\n";
// round to nearest odd value
echo(round(7.5,0,PHP_ROUND_HALF_ODD));
echo "\n";
// round towards zero
echo(round(7.5,0,PHP_ROUND_HALF_DOWN));
echo "\n";
// round away from zero
echo(round(7.5,0,PHP_ROUND_HALF_UP));
?>
	8
	7
	7
	8

- **sqrt()**

PHP provides us with a built-in function *sqrt()* which can be used to calculate the square root of a number. The *sqrt()* function in PHP is used to calculate the square root of a number.

Syntax:

***float* sqrt(value)**

Parameters: This function accepts a single parameter *$value*. It is the number whose square root you want to know.

Return Value: It returns a floating-point value which is the square root of the argument *$value* passed to it.

Examples:

<?php echo sqrt(25).” ”; echo sqrt(-25) .” ”; echo sqrt(0.09) .” ”; echo sqrt(0) .” ”; ?>	**Output:** 5 NaN 0.3 0

Date and Time Functions

- **date()**

The PHP date() function converts timestamp to a more readable date and time format. The computer stores dates and times in a format called UNIX Timestamp, which measures time as a number of seconds since the beginning of the Unix epoch (midnight Greenwich Mean Time on January 1, 1970, i.e. January 1, 1970, 00:00:00 GMT). Since this is an impractical format for humans to read, PHP converts timestamp to a format that is readable and more understandable to humans.

Syntax:

date(format, timestamp)

Parameter:

- The format parameter in the date() function specifies the format of returned date and time.

  The format parameter of the date() function is a string that can contain multiple characters allowing to generate the dates in various formats. Date-related formatting characters that are commonly used in the format string:

  - d: Represents day of the month; two digits with leading zeros (01 or 31).
  - D: Represents day of the week in the text as an abbreviation (Mon to Sun).
  - m: Represents month in numbers with leading zeros (01 or 12).
  - M: Represents month in text, abbreviated (Jan to Dec).
  - y: Represents year in two digits (08 or 14).
  - Y: Represents year in four digits (2008 or 2014).
  - h: Represents hour in 12-hour format with leading zeros (01 to 12).
  - H: Represents hour in 24-hour format with leading zeros (00 to 23).
  - i: Represents minutes with leading zeros (00 to 59).
  - s: Represents seconds with leading zeros (00 to 59).
  - a: Represents lowercase antemeridian and post meridian (am or pm).
  - A: Represents uppercase antemeridian and post meridian (AM or PM).

  The parts of the date can be separated by inserting other characters, like hyphens (-), dots (.), slashes (/), or spaces to add additional visual formatting.

- The timestamp is an optional parameter, if it is not included then the current date and time will be used.

Example:

```
<?php
 echo "Today's date is :";
 $today = date("d/m/Y");
 echo $today;?>
```

```
Today's date is :05/12/2017
```

- **checkdate()**

The checkdate() function is a built-in function in PHP which checks the validity of the date passed in the arguments. It accepts the date in the format mm/dd/yyyy. The function returns a boolean value. It returns true if the date is a valid one, else it returns false.

Syntax:

```
checkdate ( $month, $day, $year )
```

Parameters:

1. $month – This parameter specifies the month. The month has to be in between 1 to 12 for a valid date.
2. $day – This parameter specifies the day. The day can be in range 1-31 depending on the month entered for it to be a valid day. In case of a leap year, the day is in range 1-29 and for a non-leap year the day is in range 1-28.
3. $year – This parameter specifies the year. The year has to be in range 1-32767 inclusive depending on the $month and $day for it to be a valid date.

Return Value: The function returns a boolean value. It returns true if the passed date is a valid date. It returns false if the passed date is not a valid one. Examples:

Example:

```
<?php
// PHP program to demonstrate the checkdate() function
$month = 12;
$day = 31;
$year = 2017;
// returns a boolean value after validation of date
var_dump(checkdate($month, $day, $year));
var_dump(checkdate(2, 29, 2017));
?>
```

Output:

```
bool(true)
bool(false)
```

- **getdate()**

The getdate() function is an inbuilt function in PHP which is used to get date/time information of the current local date/time.

Syntax:

```
getdate($timestamp)
```

Parameters: The getdate() function accepts one parameter and it is described below:

- **$timestamp:** It is an optional parameter which specifies an integer unix timestamp. Default is the current local time **(**time()**)**.

Return Value:
The function returns an array with information of the timestamp.

- [seconds] – seconds
- [minutes] – minutes
- [hours] – hours
- [mday] – day of the month
- [wday] – day of the week
- [mon] – month
- [year] – year
- [yday] – day of the year
- [weekday] – name of the weekday
- [month] – name of the month
- [0] – seconds since Unix Epoch

Example:

```
<?php
// PHP program to illustrate
```

Output:

```
Array
```

// getdate() function
 	
print_r(getdate());
?>
	(
	[seconds] => 40
	[minutes] => 25
	[hours] => 6
	[mday] => 3
	[wday] => 2
	[mon] => 7
	[year] => 2018
	[yday] => 183
	[weekday] => Tuesday
	[month] => July
	[0] => 1530599140)

- **mktime()**

The mktime() function is an inbuilt function in PHP which is used to return the Unix timestamp for a date. The timestamp returns a long integer containing the number of seconds between the Unix Epoch (January 1, 1970, 00:00:00 GMT) and the time specified. The hour, minute, second, month, day and year are sent as parameters to the mktime() function and it returns an integer Unix timestamp on success and False on error.

Syntax:

```
int mktime( $hour, $minute, $second, $month, $day, $year, $is_dst)
```

Parameters:

- $hour: It is an optional parameter which specifies the hour.
- $minute: It is an optional parameter which specifies the minute.
- $second: It is an optional parameter which specifies the second.
- $month: It is an optional parameter which specifies the month.
- $day: It is an optional parameter which specifies the day.
- $year: It is an optional parameter which specifies the year.
- $is_dst: It is an optional parameter which can be set to 1 if the time is during daylight savings time (DST), or 0 if it is not.

Return Value: This function returns an integer Unix timestamp on success and False on error.

Exceptions:

- PHP 5.3.0 version throws an E_DEPRECATED error if the is_dst parameter is used.
- The mktime() function throws a E_NOTICE on every call to a date/time if the time zone is not valid.

Example:

<?php // Using mktime() function to know the complete date	**Output:** Dec-01-2021

echo date("M-d-Y", mktime(0, 0, 0, 12, 1, 2021)) . "
";
// Using mktime() function to know the
// complete date for an out-of-range input
echo date("M-d-Y", mktime(0, 0, 0, 12, 40, 2022));
?>
	Jan-09-2022

- **time()**

The time() function is a built-in function in PHP which returns the current time measured in the number of seconds since the Unix Epoch. The number of seconds can be converted to the current date using date() function in PHP.

Syntax:

int time()

Parameter: This function does not accepts any parameters as shown above.

Return Value: This function returns the current time measured in the number of seconds since the Unix Epoch.

<?php
// PHP program to demonstrate the use of current
// date since Unix Epoch
// variable to store the current time in seconds
$currentTimeinSeconds = time();
// converts the time in seconds to current date
$currentDate = date('Y-m-d', $currentTimeinSeconds);
// prints the current date
echo ($currentDate);
?>
	Output:
	2022-07-12

Get and Post Methods

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers. HTTP works as a request-response protocol between a client and server. A web browser may be the client, and an application on a computer that hosts a website may be the server. A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

There are 2 HTTP request methods:

- **GET:** Requests data from a specified resource.
- **POST:** Submits data to be processed to a specified resource.

GET Method: In the GET method, the data is sent as URL parameters that are usually strings of name and value pairs separated by ampersands (&). In general, a URL with GET data will look like this:

Example:

http://www.example.com/action.php?**name=***Sam***&weight=***55*

Here, the bold parts in the URL denote the GET parameters and the italic parts denote the value of those parameters. More than one parameter=value can be embedded in the URL by concatenating with ampersands (&). One can only send simple text data via GET method.

Example:

```
<?php
 error_reporting(0);
 if( $_GET["name"] || $_GET["weight"] )
 {
   echo "Welcome ". $_GET['name']. "<br />";
   echo "You are ". $_GET['weight']. " kgs in weight.";
   exit();
 }
?>
<html>
<body>
 <form action="<?php $_PHP_SELF ?>" method="GET">
  Name: <input type="text" name="name" />
  Weight:<input type="text" name="weight" />
      <input type="submit" />
 </form>
</body>
</html>
```

Advantages:

- Since the data sent by the GET method are displayed in the URL, it is possible to bookmark the page with specific query string values.
- GET requests can be cached and GET requests to remain in the browser history.
- GET requests can be bookmarked.

Disadvantages:

- The GET method is not suitable for passing sensitive information such as the username and password, because these are fully visible in the URL query string as well as potentially stored in the client browser's memory as a visited page.
- Because the GET method assigns data to a server environment variable, the length of the URL is limited. So, there is a limitation for the total data to be sent.

POST Method: In the POST method, the data is sent to the server as a package in a separate communication with the processing script. Data sent through the POST method will not be visible in the URL.

Example:

```
<?php
  error_reporting(0);
  if( $_POST["name"] || $_POST["weight"] )
```

```
     {
      if (preg_match("/[^A-Za-z'-]/",$_POST['name'] ))
       {
         die ("invalid name and name should be alpha");
      }
     echo "Welcome ". $_POST['name']. "<br />";
     echo "You are ". $_POST['weight']. " kgs in weight.";
     exit();
       }
?>
<html>
<body>
  <form action = "<?php $_PHP_SELF ?>" method = "POST">
    Name: <input type = "text" name = "name" />
    Weight: <input type = "text" name = "weight" />
           <input type = "submit" />
  </form>
</body>
</html>
```

Advantages:

- It is more secure than GET because user-entered information is never visible in the URL query string or in the server logs.
- There is a much larger limit on the amount of data that can be passed and one can send text data as well as binary data (uploading a file) using POST.

Disadvantages:

- Since the data sent by the POST method is not visible in the URL, so it is not possible to bookmark the page with a specific query.
- POST requests are never cached
- POST requests do not remain in the browser history.
- **Difference between HTTP GET and HTTP POST**

HTTP GET	HTTP POST
In GET method we can not send large amount of data rather limited data is sent because the request parameter is appended into the URL.	In POST method large amount of data can be sent because the request parameter is appended into the body.
GET request is comparatively better than Post so it is used more than the Post request.	POST request is comparatively less better than Get so it is used less than the Get request.
GET request is comparatively less secure because the data is exposed in the URL bar.	POST request is comparatively more secure because the data is not exposed in the URL bar.
Request made through GET method are stored in Browser history.	Request made through POST method is not stored in Browser history.

GET method request can be saved as bookmark in browser.	POST method request can not be saved as bookmark in browser.
Request made through GET method are stored in cache memory of Browser.	Request made through POST method are not stored in cache memory of Browser.
Data passed through GET method can be easily stolen by attackers.	Data passed through POST method can not be easily stolen by attackers.
In GET method only ASCII characters are allowed.	In POST method all types of data is allowed.

2.2 Files: Include, File Parsing, Directories, File Uploading, File Downloading

PHP allows us to create various functions and various elements that are used multiple times in multiple pages. Scripting the same function in multiple pages is a task of great effort and would consume lots of time & also impacts the execution of the code. This can be avoided if we follow and use the concept of file inclusion which helps us to include various files including text or codes into a single program which saves the effort of writing the full function or code multiple times. This also provides another advantage. If we want to change any code then instead of editing it in all the files, we just need to edit the source file and all codes will be automatically changed.

There are two functions that help us to include files:

include():

This function is used to copy all the contents of a file called within the function, text wise into a file from which it is called. This happens before the server executes the code.

Example:

even.php

```
<?php

 // File to be included
 echo "Hello from PHP";
?>
```

Now, try to include this file into another PHP file index.php file, will see the contents of both the file are shown.

- index.php

```
<?php
  include("even.php");
  echo "<br>Above File is Included"
?>
```

require():

The require() function performs same as the include() function. It also takes the file that is required and copies the whole code into the file from where the require() function is called.

Example: This example is using the require() function in PHP.

- even.php

```
<?php
 // File to be included
 echo "Hello GeeksforGeeks";
?>
```

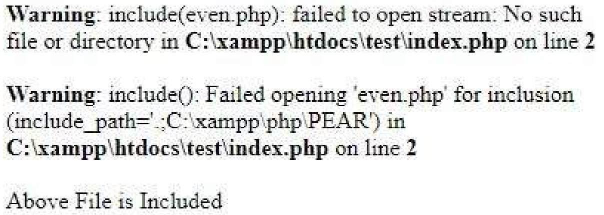
Now, if we try to include this file using require() function this file into a web page we need to use a index.php file. We will see that the contents of both files are shown.

- index.php

```
<?php
   require("even.php");
   echo "<br>Above File is Required"
?>
```

include() vs require(): Both functions act as same and produce the same results, but if by any chance a fatal error arises, then the difference comes to the surface, which we will see following this example. Consider the following code:

```
<?php
   include("even.php");
   echo "<br>Above File is Included"
?>
```



```
Warning: include(even.php): failed to open stream: No such file or directory in C:\xampp\htdocs\test\index.php on line 2

Warning: include(): Failed opening 'even.php' for inclusion (include_path='.;C:\xampp\php\PEAR') in C:\xampp\htdocs\test\index.php on line 2

Above File is Included
```

Output: Now, if we don't have a file named even.php, then in the case of the include(), the following output will be shown with warnings about a missing file, but at least the output will be shown from the index.php file:

Warning message in include() function

In the case of the require(), if the file PHP file is missing, a fatal error will rise and no output is shown and the execution halts.

Warning message in require() function

This is the only difference. This also shows that require() function is better than the include() function since the script should not continue executing if files are missing or such an error is generated.

include()	**require()**
The include() function does not stop the execution of the script even if any error occurs.	The require() function will stop the execution of the script when an error occurs.
The include() function does not give a fatal error.	The require() function gives a fatal error
The include() function is mostly used when the file is not required and the application should continue to execute its process when the file is not found.	The require() function is mostly used when the file is mandatory for the application.

The include() function will only produce a warning (E_WARNING) and the script will continue to execute.	The require() will produce a fatal error (E_COMPILE_ERROR) along with the warning.

File Parsing

File handling is needed for any application. For some tasks to be done file needs to be processed. File handling in PHP is similar as file handling is done by using any programming language like C. PHP has many functions to work with normal files. Those functions are:

fopen()

The *fopen()* function in PHP is an inbuilt function which is used to open a file or an URL. It is used to bind a resource to a steam using a specific filename. The filename and mode to be checked are sent as parameters to the *fopen()* function and it returns a file pointer resource if a match is found and a False on failure. The error output can be hidden by adding an '@' in front of the function name.

Syntax:

```
resource fopen ( $file, $mode, $include_path, $context)
```

Parameters:

- ***$file***: It is a mandatory parameter which specifies the file.
- ***$mode***: It is a mandatory parameter which specifies the access type of the file or stream.
  It can have the following possible values:
  - **"r":** It represents Read only. It starts at the beginning of the file.
  - **"r+":** It represents Read/Write.It starts at the beginning of the file.
  - **"w":** It represents Write only.It opens and clears the contents of file or create a new file if it doesn't exist.
  - **"w+":** It represents Read/Write. It opens and clears the contents of file or creates a new file if it doesn't exist.
  - **"a":** It represents Write only. It opens and writes to the end of the file or creates a new file if it doesn't exist.
  - **"a+":** It represents Read/Write. It preserves the file's content by writing to the end of the file.
  - **"x":** It represents Write only. It creates a new file and returns FALSE and an error if the file already exists.
  - **"x+":** It represents Read/Write.It creates a new file and returns FALSE and an error if file already exists.
- ***$include_path***: It is an optional parameter which is set to 1 if you want to search for the file in the include_path (Ex. php.ini).

- ***$context***: It is an optional parameter which is used to set the behavior of the stream.

Return Value:

It returns a file pointer resource on success, or FALSE on error.

Exceptions:

- When writing to a text file, the correct line-ending character should be used based on the platform.For example Unix systems use \n, Windows systems use \r\n, and Macintosh systems use \r as the line ending character.
- It is recommended to use the 'b' flag when opening files with fopen().
- An error of level E_WARNING is generated if the open fails.
- When safe mode is enabled, PHP checks whether the directory in which the script is operating has the same UID (owner) as the script that is being executed.
- If you are unsure whether filename is a file or a directory, you may need to use the is_dir() function before calling fopen() since fopen() function may also succeed when filename is a directory.

fread()

The *fread()* function in PHP is an inbuilt function which reads up to length bytes from the file pointer referenced by file from an open file. The *fread()* function stops at the end of the file or when it reaches the specified length passed as a parameter, whichever comes first. The file and the length which has to be read are sent as parameters to the *fread()* function and it returns the read string on success, or FALSE on failure.

Syntax:

```
string fread ( $file, $length )
```

Parameters:

- ***$file***: It is a mandatory parameter which specifies the file.
- ***$length***: It is a mandatory parameter which specifies the maximum number of bytes to be read.

Return Value:

- It returns the read string on success, or False on failure.

Exceptions:

- Both binary data, like images and character data, can be written with this function since *fread()* is binary-safe.
- To get the contents of a file only into a string, use *file_get_contents()* as it has much better performance than the code above.

- Since systems running Windows differentiate between binary and text files, the file must be opened with 'b' included in *fopen()* mode parameter.

fwrite()

The fwrite() function in PHP is an inbuilt function which is used to write to an open file. The fwrite() function stops at the end of the file or when it reaches the specified length passed as a parameter, whichever comes first. The file, string and the length which has to be written are sent as parameters to the fwrite() function and it returns the number of bytes written on success, or FALSE on failure.

Syntax:

```
fwrite(file, string, length)
```

Parameters:

1. **file** : It is a mandatory parameter which specifies the file.
2. **string** : It is a mandatory parameter which specifies the string to be written.
3. **length** : It is an optional parameter which specifies the maximum number of bytes to be written.

Return Value: It returns the number of bytes written on success, or False on failure.

Exceptions:

1. Both binary data, like images and character data, can be written with this function since fwrite() is binary-safe.
2. If writing operation is performed twice to the file pointer, then the data will be appended to the end of the file content.

fclose()

The fclose() function in PHP is an inbuilt function which is used to close a file which is pointed by an open file pointer. The fclose() function returns true on success and false on failure. It takes the file as an argument which has to be closed and closes that file.

Syntax:

```
bool fclose( $file )
```

Parameters: The fclose() function in PHP accepts only one parameter which is $file. This parameter specifies the file which has to be closed.

Return Value: It returns true on success and false on failure.

Errors And Exception:

1. A file has to be closed first using the fclose() function if it has been written via fwrite() function and you have to read the contents of the file.
2. The fclose() function in PHP doesn't works for remote files.It only works on files which are accessible by the server's filesystem.

Common examples for all file parsing commands are as follows:

Example:

<?php
// Opening a file using fopen()
// function in read only mode
$myfile = fopen("test.txt", "r") or die("File does not exist!");
?>
	Output:
	File does not exist!

<?php
// Opening a file using fopen()
// function in read/write mode
$myfile = fopen("test.txt", 'r+')
    or die("File does not exist!");
 	
$pointer = fgets($myfile);
echo $pointer;
fclose($myfile);
?>
	Output:
	One line from test,txt file will be printed

<?php
// Opening a file using fopen() function
// in read mode along with b flag
$myfile = fopen("test.txt", "rb");
$contents = fread($myfile, filesize($myfile));
fclose($myfile);
print $contents;
?>
	Output:
	Complete test.txt file get printed

Directory Handling in PHP

PHP includes a set of directory functions to deal with the operations, like, listing directory contents, and create/ open/ delete specified directory and etc. These basic functions are listed below.

- *mkdir()*: To make new directory.
- *opendir()*: To open directory.
- *readdir()*: To read from a directory after opening it.
- closedir(): To close directory with resource-id returned while opening.
- *rmdir()*: To remove directory.

Creating Directory

For creating a new directory using PHP programming script, *mkdir()* function is used, and, the usage of this function is as follows.

```
<?php
mkdir($directory_path, $mode, $recursive_flag, $context);
?>
```

This function accepts four arguments as specified. Among them, the first argument is mandatory, whereas, the remaining set of arguments is optional.

- *$directory_path*: By specifying either relative and absolute path, a new directory will be created in such location if any, otherwise, will return an error indicating that there are no such locations.
- $mode: The mode parameter accepts octal values on which the accessibility of the newly created directory depends.
- $recursive: This parameter is a flag and has values either *true* or *false*, that allow or refuse to create nested directories further.
- $context: As similar as we have with PHP *unlink()* having a stream for specifying protocols and etc.

This function will return boolean data, that is, *true* on successful execution, *false* otherwise.

Listing the Directory Content

For listing the contents of a directory, we require two of the above-listed PHP directory functions, these are, *opendir()* and *readdir()*. There are two steps in directory listing using the PHP program.

- Step 1: Opening the directory.
- Step 2: Reading content to be listed one by one using a PHP loop.

Step 1: Opening Directory Link

As its name, *opendir*() function is used to perform this step. And, it has two arguments, one is compulsory for specifying the path of the directory, and the other is optional, expecting stream context if any. The syntax will be,

```
<?php
opendir($directory_path, $context);
?>
```

Unlike PHP *mkdir()* returning boolean value, this function will return resource data as like as fopen(), mysql_query() and etc. After receiving the resource identifier returned by this function, then only we can progress with the subsequent steps to read, rewind, or close the required directory with the reference of this resource id.

Otherwise, a PHP error will occur for indicating to the user, that the resource id is not valid.

Step 2: Reading Directory Content

For performing this step, we need to call *readdir()* function recursively until the directory handle reaches the end of the directory. For that, we need to specify the resource-id returned while invoking *opendir()*, indicated as directory handle.

PHP *readdir()* will return string data on each iteration of the loop, and this string will be the name of each item stored in the directory with its corresponding extension. For example,

```
<?php
$directory_handle = opendir($directory_path);
```

```
while ($directory_item = readdir($directory_handle)) {
  echo $directory_item . "<br>";
}
?>
```

And, thereby executing the above code sample, we can list the content of a directory as expected.

Closing Directory Link

Once the directory link is opened to perform a set of dependent operations like reading directory content, we need to close this link after completing the related functionalities required. For example,

```
<?php
$directory_handle = opendir($directory_path);
...
...
closedir($directory_handle);
?>
```

Removing Directory

For removing the entire directory, PHP provides a function named as *rmdir()* which accepts the same set of arguments, as *mkdir()*.

These are, the *$directory_path* and *$context*(Optional) as stated below.

```
<?php
rmdir($directory_path, $mode, $recursive_flag, $context); ?>
```

But, this function will remove the directory, if and only if it is empty. For removing the non-empty directory, we need to create a user define function that recursively calls *unlink()* function to remove each file stored in the directory to be deleted.

Finding and Changing the Current Working Directory

It is unlikely that a web application will be able to perform all of its file related tasks in a single directory. For this reason, it is vital to be able to both find out the current working directory, and change to another directory from within a PHP script.

The current working directory can be identified using the ***getCwd()*** function:

```
<?php
$current_dir = getCwd();
echo "Current directory is $current_dir";
?>
```

The current working directory can be changed using the *chdir()* function. *chdir()* takes as the only argument the path of the new directory:

```
<?php
$current_dir = getCwd();
```

```
echo "Current directory is $current_dir <br>";
chdir ("/tmp");
$current_dir = getCwd();
echo "Current directory is now $current_dir <br>";
?>
```

Listing Files in a Directory

The files in a directory can be read using the PHP *scandir()* function. *scandir()* takes two arguments. The first argument is the path the directory to be scanned. The second optional argument specifies how the directory listing is to be sorted. If the argument is 1 the listing is sorted reverse-alphabetically. If the argument is omitted or set to 0 the list is sorted alphabetically:

```
<?php
chdir ("/tmp");
$current_dir = getCwd();
echo "Current directory is now $current_dir";
$array = scandir(".", 1);
print_r($array);
?>
```

Copying Files from One Location to Another

You can copy a file from one location to another by calling PHP copy() function with the file's source and destination paths as arguments. If the destination file already exists it'll be overwritten. Here's an example which creates a copy of "example.txt" file inside backup folder.

Example

```
<?php
// Source file path
$file = "example.txt";
// Destination file path
$newfile = "backup/example.txt";
// Check the existence of file
if(file_exists($file)){
   // Attempt to copy file
   if(copy($file, $newfile)){
      echo "File copied successfully.";
   } else{
      echo "ERROR: File could not be copied.";
   }
} else{
   echo "ERROR: File does not exist.";?>
```

To make this example work, the target directory which is *backup* and the source file i.e. "example.txt" has to exist already; otherwise PHP will generate an error.

File Uploading and Downloading

PHP allow you to upload any type of a file i.e. image, binary or text files.etc..,PHP has one in built global variable i.e. **$_FILES**, it contains all information about file.By the help of **$_FILES** global variable, we can get file name, file type, file size and temparary file name associated with file.

- **$_FILES['filename']['name']** - returns file name.
- **$_FILES['filename']['type']** - returns MIME type of the file.
- **$_FILES['filename']['size']** - returns size of the file (in bytes).
- **$_FILES['filename']['tmp_name']** - returns temporary file name of the file which was stored on the server.

Step 1: Creating an HTML form to upload the file

Below is the HTML source code for the HTML form for uploading the file to the server. In the HTML <form> tag, we are using "enctype='multipart/form-data" which is an encoding type that allows files to be sent through a POST method. Without this encoding, the files cannot be sent through the POST method. We must use this enctype if you want to allow users to upload a file through a form.

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>File Upload Form</title>
</head>
<body>
        <form action="file-upload-manager.php" method="post" enctype="multipart/form-data">
        <!--multipart/form-data ensures that form data is going to be encoded as MIME data-->
                <h2>Upload File</h2>
                <label for="fileSelect">Filename:</label>
                <input type="file" name="photo" id="fileSelect">
                <input type="submit" name="submit" value="Upload">
                <!-- name of the input fields are going to be used in our php script-->
<p><strong>Note:</strong>Only .jpg, .jpeg, .png formats allowed to a max size of 2MB.</p>
        </form>
</body>
</html>
```

Note: In addition to a file-select field the upload form must use the HTTP post method and must contain an enctype="multipart/form-data" attribute. This attribute ensures that the form data is encoded as mulitpart MIME data — which is required for uploading the large quantities of binary data such as image, audio, video, etc.

Step 2: Processing the uploaded file

Here's the complete code of our "upload-manager.php" file. It will store the uploaded file in a "upload" folder on permanent basis as well as implement some basic security check like file type and file size to ensure that users upload the correct file type and within the allowed limit.

```
<?php
// Check if the form was submitted
if($_SERVER["REQUEST_METHOD"] == "POST")
{
    // Check if file was uploaded without errors
    if (isset($_FILES["photo"]) && $_FILES["photo"]["error"] == 0)
    {
        $allowed_ext = array("jpg" => "image/jpg",
                                "jpeg" => "image/jpeg",
                                "gif" => "image/gif",
                                "png" => "image/png");
        $file_name = $_FILES["photo"]["name"];
        $file_type = $_FILES["photo"]["type"];
        $file_size = $_FILES["photo"]["size"];
        // Verify file extension
        $ext = pathinfo($filename, PATHINFO_EXTENSION);
        if (!array_key_exists($ext, $allowed_ext))
            die("Error: Please select a valid file format.");
        // Verify file size - 2MB max
        $maxsize = 2 * 1024 * 1024;
        if ($file_size > $maxsize)
            die("Error: File size is larger than the allowed limit of 2MB");
        // Verify MYME type of the file
        if (in_array($file_type, $allowed_ext))
        {
            // Check whether file exists before uploading it
            if (file_exists("upload/".$_FILES["photo"]["name"]))
                echo $_FILES["photo"]["name"]." already exists!";
            else
            {
                move_uploaded_file($_FILES["photo"]["tmp_name"],
                        "uploads/".$_FILES["photo"]["name"]);
                echo "Your file uploaded successfully.";
            }
        }
        else
        {
            echo "Error: Please try again!";
        }
    }
    else
    {
        echo "Error: ". $_FILES["photo"]["error"];
    }
}
?>
```

Once the form is submitted information about the uploaded file can be accessed via PHP superglobal array called $_FILES. For example, our upload form contains a file select field called photo (i.e. name="photo"), if any user uploaded a file using this field, we can obtains its details like the name, type, size, temporary name or any error occurred while attempting the upload via the $_FILES["photo"] associative array, like this:

- $_FILES["photo"]["name"] — This array value specifies the original name of the file, including the file extension. It doesn't include the file path.
- $_FILES["photo"]["type"] — This array value specifies the MIME type of the file.
- $_FILES["photo"]["size"] — This array value specifies the file size, in bytes.
- $_FILES["photo"]["tmp_name"] — This array value specifies the temporary name including full path that is assigned to the file once it has been uploaded to the server.
- $_FILES["photo"]["error"] — This array value specifies error or status code associated with the file upload, e.g. it will be 0, if there is no error.

move_uploaded_file() function

The move_uploaded_file() function is used to move the uploaded file to a new location. It moves the file only if it is uploaded through the POST request.

Syntax:

```
move_uploaded_file ( string $filename , string $destination )
```

First Configure the php.ini File by ensure that PHP is configured to allow file uploads. In your php.ini file, search for the file_uploads directive, and set it to On i.e. file_uploads = On

2.3 Cookies and Sessions, Send Email

Session

session refers to a frame of communication between two medium. A PHP session is used to store data on a server rather than the computer of the user. Session identifiers or SID is a unique number which is used to identify every user in a session based environment. The SID is used to link the user with his information on the server like posts, emails etc.

How are sessions better than cookies?

Although cookies are also used for storing user related data, they have serious security issues because cookies are stored on the user's computer and thus they are open to attackers to easily modify the content of the cookie. Addition of harmful data by the attackers in the cookie may result in the breakdown of the application. Apart from that cookies affect the performance of a site since cookies send the user data each time the user views a page. Every time the browser requests a URL to the server, all the cookie data for that website is automatically sent to the server within the request.

Below are different steps involved in PHP sessions:

- **Starting a PHP Session**: The first step is to start up a session. After a session is started, session variables can be created to store information. The PHP **session_start()** function is used to begin a new session.It also creates a new session ID for the user.

Below is the PHP code to start a new session:

```
<?php
session_start();
?>
```

- **Storing Session Data**: Session data in key-value pairs using the **$_SESSION[]** superglobal array.The stored data can be accessed during lifetime of a session.

Below is the PHP code to store a session with two session variables Rollnumber and Name:

```
<?php
session_start();
$_SESSION["Rollnumber"] = "11";
$_SESSION["Name"] = "Ajay";
?>
```

- **Accessing Session Data**: Data stored in sessions can be easily accessed by firstly calling **session_start()** and then by passing the corresponding key to the **$_SESSION** associative array.

The PHP code to access a session data with two session variables Rollnumber and Name is shown below:

```
<?php
session_start();
echo 'The Name of the student is :' . $_SESSION["Name"] . '<br>';
echo 'The Roll number of the student is :' . $_SESSION["Rollnumber"] . '<br>';
?>
```

Output:

```
The Name of the student is :Ajay
The Roll number of the student is :11
```

- **Destroying Certain Session Data**: To delete only a certain session data,the unset feature can be used with the corresponding session variable in the **$_SESSION** associative array.

The PHP code to unset only the “Rollnumber” session variable from the associative session array:

```
<?php
session_start();
if(isset($_SESSION["Name"])){
  unset($_SESSION["Rollnumber"]);
}
?>
```

- **Destroying Complete Session**: The **session_destroy()** function is used to completely destroy a session. The session_destroy() function does not require any argument.

```
<?php
session_start();
session_destroy();
?>
```

Important Points

1. The session IDs are randomly generated by the PHP engine .
2. The session data is stored on the server therefore it doesn’t have to be sent with every browser request.
3. The session_start() function needs to be called at the beginning of the page, before any output is generated by the script in the browser.

Cookies

A **cookie** in PHP is a small file with a maximum size of 4KB that the web server stores on the client computer. They are typically used to keep track of information such as a username that the site can retrieve to personalize the page when the user visits the website next time. A cookie can only be read from the domain that it has been issued from. Cookies are usually set in an HTTP header but JavaScript can also set a cookie directly on a browser.

Setting Cookie In PHP: To set a cookie in PHP, the **setcookie()** function is used. The setcookie() function needs to be called prior to any output generated by the script otherwise the cookie will not be set.

Syntax:

```
setcookie(name, value, expire, path, domain, security);
```

Parameters:

- **Name:** It is used to set the name of the cookie.
- **Value:** It is used to set the value of the cookie.

- **Expire:** It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed.
- **Path:** It is used to specify the path on the server for which the cookie will be available.
- **Domain:** It is used to specify the domain for which the cookie is available.
- **Security:** It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

Below are some operations that can be performed on Cookies in PHP:

- **Creating Cookies**: Creating a cookie named Auction_Item and assigning the value Luxury Car to it. The cookie will expire after 2 days(2 days * 24 hours * 60 mins * 60 seconds).

Example: This example describes the creation of the cookie in PHP.

```
<!DOCTYPE html>
<?php
  setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);
?>
<html>
<body>
  <?php
    echo "cookie is created."   ?>
  <p>
    <strong>Note:</strong>
    You might have to reload the
    page to see the value of the cookie.
  </p>
</body>
</html>
```

Output:

Cookie creation in PHP

Note: Only the name argument in the setcookie() function is mandatory. To skip an argument, the argument can be replaced by an empty string("").

Checking Whether a Cookie Is Set Or Not: It is always advisable to check whether a cookie is set or not before accessing its value. Therefore to check whether a cookie is set or not, the PHP isset() function is used. To check whether a cookie "Auction_Item" is set or not, the isset() function is executed as follows:

Example: This example describes checking whether the cookie is set or not.

```
<!DOCTYPE html>
<?php
  setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 *
60);
?>
<html>
<body>
```

Output:

Checking for the cookie to be set

```
  <?php
  if (isset($_COOKIE["Auction_Item"]))
  {
    echo "Auction Item is a  " . $_COOKIE["Auction_Item"];
  }
  else
  {
    echo "No items for auction.";
  }
  ?>
  <p>
    <strong>Note:</strong>
    You might have to reload the page
    to see the value of the cookie.
  </p>
</body>
</html>
```

Accessing Cookie Values: For accessing a cookie value, the PHP $_COOKIE superglobal variable is used. It is an associative array that contains a record of all the cookies values sent by the browser in the current request. The records are stored as a list where the cookie name is used as the key. To access a cookie named "Auction_Item", the following code can be executed.

Example: This example describes accessing & modifying the cookie value.

```
<!DOCTYPE html>
<?php
  setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 *
60);
?>
<html>
<body>
<?php
  echo "Auction Item is a  " . $_COOKIE["Auction_Item"];
?>
  <p>
    <strong>Note:</strong>
    You might have to reload the page
    to see the value of the cookie.
  </p>
</body>
</html>
```

Output:
Accessing the Cookie value

Deleting Cookies: The setcookie() function can be used to delete a cookie. For deleting a cookie, the setcookie() function is called by passing the cookie name and other arguments or empty strings but however this time, the expiration date is required to be set in the past. To delete a cookie named “Auction_Item”, the following code can be executed.

Example: This example describes the deletion of the cookie value.

```
<!DOCTYPE html>
<?php
  setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);
?>
<html>
<body>
  <?php
    setcookie("Auction_Item", "", time() - 60);
  ?>
  <?php
    echo "cookie is deleted"
  ?>
  <p>
    <strong>Note:</strong>
    You might have to reload the page
    to see the value of the cookie.
  </p>
</body>
</html>
```

Output:

Deleting the Cookie

Important Points:

- If the expiration time of the cookie is set to 0 or omitted, the cookie will expire at the end of the session i.e. when the browser closes.
- The same path, domain, and other arguments should be passed that were used to create the cookie in order to ensure that the correct cookie is deleted.

Send Mail in PHP

PHP is a server side scripting language that is enriched with various utilities required. Mailing is one of the server side utilities that is required in most of the web servers today. Mailing is used for advertisement, account recovery, subscription etc.

In order to send mails in PHP, one can use the mail() method.

Syntax:

```
bool mail(to , subject , message , additional_headers , additional_parameters)
```

Parameters:

- **to**: Specifies the email id of the recipient(s). Multiple email ids can be passed using commas
- **subject**: Specifies the subject of the mail.
- **message**: Specifies the message to be sent.

- **additional-headers**(Optional): This is an optional parameter that can create multiple header elements such as From (Specifies the sender), CC (Specifies the CC/Carbon Copy recipients), BCC (Specifies the BCC/Blind Carbon Copy Recipients. **Note:** In order to add multiple header parameters one must use '\r\n'.
- **additional-parameters**(Optional): This is another optional parameter and can be passed as an extension to the additional headers. This can specify a set of flags that are used as the sendmail_path configuration settings.

Return Type: This method returns TRUE if mail was sent successfully and FALSE on Failure.

Examples:

```
<?php
 $to = "recipient@example.com";
 $sub = "Generic Mail";
 $msg="Hello Geek! This is a generic email.";
 if (mail($to,$sub,$msg))
   echo "Your Mail is sent successfully.";
 else
   echo "Your Mail is not sent. Try Again.";
?>
```

Output :

Your Mail is sent successfully.

Sending a Mail with Additional Options

```
<?php
 $to = "recipient@example.com";
 $sub = "Generic Mail";
 $msg = "Hello Geek! This is a generic email.";
 $headers = 'From: sender@example.com' . "\r\n"
.'CC: another@example.com';
 if(mail($to,$sub,$msg,$headers))
   echo "Your Mail is sent successfully.";
 else
   echo "Your Mail is not sent. Try Again.";
?>
```

Output :

Your Mail is sent successfully.

Summary:

- Using mail() method one can send various types of mails such as standards, html mail.
- The mail() method opens the SMTP socket, attempts to send the mail, closes the socket thus is a secure option.
- mail() method should not be used for bulk mailing as it is not very cost-efficient.

- The mail() method only checks for parameter or network failure, thus a success in the mail() method doesn't guarantee that the intended person will receive the mail.

2.4 Forms: Creating, Handling, Validation of Forms, PHP Filters, JSON Parsing

Form Creating

HTML forms are used to send the user information to the server and returns the result back to the browser. For example, if you want to get the details of visitors to your website, and send them good thoughts, you can collect the user information by means of form processing. Then, the information can be validated either at the client-side or on the server-side. The final result is sent to the client through the respective web browser. To create a HTML form, **form** tag should be used.

Attributes of Form Tag:

Attribute	Description
name or id	It specifies the name of the form and is used to identify individual forms.
Action	It specifies the location to which the form data has to be sent when the form is submitted.
method	It specifies the HTTP method that is to be used when the form is submitted. The possible values are **get** and **post**. If **get** method is used, the form data are visible to the users in the url. Default HTTP method is **get**.
encType	It specifies the encryption type for the form data when the form is submitted.
Novalidate	It implies the server not to verify the form data when the form is submitted.

Controls used in forms: Form processing contains a set of controls through which the client and server can communicate and share information. The controls used in forms are:

- **Textbox:** Textbox allows the user to provide single-line input, which can be used for getting values such as names, search menu and etc.
- **Textarea:** Textarea allows the user to provide multi-line input, which can be used for getting values such as an address, message etc.
- **DropDown:** Dropdown or combobox allows the user to provide select a value from a list of values.
- **Radio Buttons:** Radio buttons allow the user to select only one option from the given set of options.
- **CheckBox:** Checkbox allows the user to select multiple options from the set of given options.
- **Buttons:** Buttons are the clickable controls that can be used to submit the form.

All the form controls given above is designed by using the **input** tag based on the **type** attribute of the tag. In the below script, when the form is submitted, no event handling mechanism is done. Event handling refers to the process done while the form is submitted. These event handling mechanisms can be done by using javaScript or

PHP. However, JavaScript provides only client-side validation. Hence, we can use PHP for form processing.

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple Form Processing</title>
</head>
<body>
  <form id="form1" method="post">
    FirstName:
    <input type="text" name="firstname" required/>
    <br>
    <br>
    LastName
    <input type="text" name="lastname" required/>
    <br>
    <br>
    Address
    <input type="text" name="address" required/>
    <br>
    <br>
    Email Address:
    <input type="email" name="emailaddress" required/>
    <br>
    <br>
    Password:
    <input type="password" name="password" required/>
    <br>
    <br>
    <input type="submit" value="Submit”/>
  </form>
</body>
</html>
```

Form Handling

Form validation is done to ensure that the user has provided the relevant information. Basic validation can be done using HTML elements. For example, in the above script, the email address text box is having a type value as “email”, which prevents the user from entering the incorrect value for an email. Every form field in the above script is followed by a required attribute, which will intimate the user not to leave any field empty before submitting the form. PHP methods and arrays used in form processing are:

- **isset():** This function is used to determine whether the variable or a form control is having a value or not.

- **$_GET[]:** It is used the retrieve the information from the form control through the parameters sent in the URL. It takes the attribute given in the url as the parameter.
- **$_POST[]:** It is used the retrieve the information from the form control through the HTTP POST method. IT takes name attribute of corresponding form control as the parameter.
- **$_REQUEST[]:** It is used to retrieve an information while using a database.

Form Processing using PHP: Above HTML script is rewritten using the above mentioned functions and array. The rewritten script validates all the form fields and if there are no errors, it displays the received information in a tabular form.

Example:

```
<?php
if (isset($_POST['submit']))
{
  if ((!isset($_POST['firstname'])) || (!isset($_POST['lastname'])) ||
    (!isset($_POST['address'])) || (!isset($_POST['emailaddress'])) ||
    (!isset($_POST['password'])) || (!isset($_POST['gender'])))
  {
    $error = "*" . "Please fill all the required fields";
  }
  else
  {
    $firstname = $_POST['firstname'];
    $lastname = $_POST['lastname'];
    $address = $_POST['address'];
    $emailaddress = $_POST['emailaddress'];
    $password = $_POST['password'];
    $gender = $_POST['gender'];
  }
}
?>
<html>
<head>
  <title>Simple Form Processing</title>
</head>
<body>
  <h1>Form Processing using PHP</h1>
  <fieldset>
    <form id="form1" method="post" action="form.php">
      <?php
        if (isset($_POST['submit']))
        {
          if (isset($error))
          {
             echo "<p style='color:red;'>"
             . $error . "</p>";
          }
        }
        ?>
        FirstName:
        <input type="text" name="firstname"/>
         <span style="color:red;">*</span>
        <br>
        <br>
        Last Name:
        <input type="text" name="lastname"/>
          <span style="color:red;">*</span>
        <br>
        <br>
        Address:
        <input type="text" name="address"/>
          <span style="color:red;">*</span>
        <br>
        <br>
        Email:
        <input type="email" name="emailaddress"/>
          <span style="color:red;">*</span>
        <br>
        <br>
        Password:
        <input type="password" name="password"/>
          <span style="color:red;">*</span>
        <br>
        <br>
        Gender:
        <input type="radio"
            value="Male"
            name="gender"> Male
        <input type="radio"
            value="Female"
            name="gender">Female
        <br>
        <br>
        <input type="submit" value="Submit" name="submit" />
    </form>
  </fieldset>
  <?php
    if(isset($_POST['submit']))
    {
      if(!isset($error))
```

```
      {
          echo"<h1>INPUT RECEIVED</h1><br>";
          echo "<table border='1'>";
          echo "<thead>";
          echo "<th>Parameter</th>";
          echo "<th>Value</th>";
          echo "</thead>";
          echo "<tr>";
          echo "<td>First Name</td>";
          echo "<td>".$firstname."</td>";
          echo "</tr>";
          echo "<tr>";
          echo "<td>Last Name</td>";
          echo "<td>".$lastname."</td>";
          echo "</tr>";
          echo "<tr>";
          echo "<td>Address</td>";
          echo "<td>".$address."</td>";
          echo "</tr>";
          echo "<tr>";
          echo "<td>Email Address</td>";
          echo "<td>" .$emailaddress."</td>";
          echo "</tr>";
          echo "<tr>";
          echo "<td>Password</td>";
          echo "<td>".$password."</td>";
          echo "</tr>";
          echo "<tr>";
          echo "<td>Gender</td>";
          echo "<td>".$gender."</td>";
          echo "</tr>";
          echo "</table>";
    }
   }
  ?>
</body>
</html>
```

Note: When the PHP and HTML are coded in a single file, the file should be saved as PHP. In the form, the value for the action parameter should be a file name.

Form Validation

Data validation is an integral part of web development, especially while working with forms where the user first enters their personal data and then sends that to the database. Data sent in an invalid format can cause DBMS security problems. Hackers often use SQL injections to insert malicious SQL commands into the database. SQL injections can even destroy the database once inserted. An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things: Empty String, Validate String, Validate Numbers, Validate Email, Validate URL. Input length etc.

PHP Filters

PHP Filter is an extension that filters the data by either sanitizing or validating it. It plays a crucial role in security of a website, especially useful when the data originates from unknown or foreign sources, like user supplied input. For example data from a HTML form.

There are mainly two types of filters which are listed below:

- Validation: is used to validate or check if the data meets certain qualifications or not. For example, passing in FILTER_VALIDATE_URL will determine if the data is a valid url, but it will not change the existing data by itself.
- Sanitization: unlike validation, sanitization will sanitize data so as to ensure that no undesired characters by removing or altering the data. For example passing in FILTER_SANITIZE_EMAIL will remove all the characters that are inappropriate for an email address to contain. That said, it does not validate the data.

Example 1: PHP program to validate URL using FILTER_VALIDATE_URL filter.

```
<?php
// PHP program to validate URL
// Declare variable and initialize it to URL
$url = "https://www.geeksforgeeks.org";
  // Use filter function to validate URL
if (filter_var($url, FILTER_VALIDATE_URL)) {
   echo("valid URL");
}
else {
   echo("Invalid URL");
}
?>
```

Example 2: PHP program to validate email using FILTER_VALIDATE_EMAIL filter.

```
<?php
// PHP program to validate email
// Declare variable and initialize it to email
$email = "xyz@gmail.com";
// Use filter function to validate email
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
   echo "Valid Email";
}
else {
  echo "Invalid Email";
}?>
```

Filter Functions: The filter function is used to filter the data coming from insecure source.

- filter_var(): Filters a specific variable
- filter_var_array():Filters multiple variable i.e. array of variable
- filter_has_var(): Check if the variable of specific input type exists or not
- filter_id():helps to get filter id of the specified filter name
- filter_list():Returns a list of supported filter name in the form of array.
- filter_input():Gets an external variable and filters it if set to do so.
- filter_input_array():same as filter_input() but here Gets multiple variables i.e. array of variable and filters them if set to do so.

Predefined Filter Constants: There are many predefined filter constants which are listed below:

- Validate filter constants:
  - FILTER_VALIDATE_BOOLEAN: Validates a boolean
  - FILTER_VALIDATE_INT: Validates an integer
  - FILTER_VALIDATE_FLOAT: Validates a float
  - FILTER_VALIDATE_REGEXP: Validates a regular expression
  - FILTER_VALIDATE_IP: Validates an IP address
  - FILTER_VALIDATE_EMAIL: Validates an e-mail address
  - FILTER_VALIDATE_URL: Validates an URL
- Sanitize filter constants:
  - FILTER_SANITIZE_EMAIL: Removes all illegal characters from an e-mail address
  - FILTER_SANITIZE_ENCODED: Removes/Encodes special characters
  - FILTER_SANITIZE_MAGIC_QUOTES: Apply addslashes() function
  - FILTER_SANITIZE_NUMBER_FLOAT: Remove all characters, except digits, +- and optionally
  - FILTER_SANITIZE_NUMBER_INT: Removes all characters except digits and + –
  - FILTER_SANITIZE_SPECIAL_CHARS: Removes special characters
  - FILTER_SANITIZE_FULL_SPECIAL_CHARS Encoding quotes can be disabled by using FILTER_FLAG_NO_ENCODE_QUOTES.
  - FILTER_SANITIZE_STRING : Removes tags/special characters from a string

- FILTER_SANITIZE_STRIPPED : Alias of FILTER_SANITIZE_STRING
- FILTER_SANITIZE_URL: Removes all illegal character from s URL

```
<?php
  if(isset($_POST['email'])) {
    echofilter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
    echo"<br/><br/>";
  }
  if(isset($_POST['homepage'])) {
    echofilter_var($_POST['homepage'], FILTER_SANITIZE_URL);
    echo"<br/><br/>";
  }
?>
<form name="form1"method="post"action="form-sanitize.php">
  Email Address: <br/>
  <input type="text"name="email"value="<?php echo $_POST['email']; ?>"size="50"/><br/><br/>
  Home Page: <br/>
  <input type="text"name="homepage"value="<?php echo $_POST['homepage']; ?>"size="50"/><br/>
  <br/>
  <input type="submit"/>
</form>
```

- Other filter constants:
  - FILTER_UNSAFE_RAW :Do nothing, optionally strip/encode special characters
  - FILTER_CALLBACK :Call a user-defined function to filter data

Example

```
<?php
function convertSpace($string)
 {
 return str_replace(" ", "_", $string);
 }
$string = "Peter is a great guy!";
echo filter_var($string, FILTER_CALLBACK,
array("options"=>"convertSpace"));
?>
```

The output of the code will be:

Peter_is_a_great_guy!

```
<?php
$string="Peter is a great guy!";
echo filter_var($string, FILTER_CALLBACK,
array("options"=>"strtoupper"));
?>
```

The output of the code will be:

PETER IS A GREAT GUY!

JSON Parsing

PHP is a server-side scripting language used to process the data. JSON stands for JavaScript object notation. JSON data is written as name/value pairs.

Syntax:

```
{
“Data”:[{
“key”:”value”,
“key”:value,
“key n “:”value”
},
. . .
. . .
{
“key”:”value”,
“key”:value,
“key n “:”value”
}]
}
```

Advantages:

- JSON does not use an end tag.
- JSON is a shorter format.
- JSON is quicker to read and write.
- JSON can use arrays.

Approach: Create a JSON file and save it as data.json. We have taken *student* data in the file. The contents are as follows.

```
{
  "customers": [
   {
    "name": "Andrew",
    "email": "andrew@something.com",
    "age": 62,
    "countries": [
     {
      "name": "Italy",
      "year": 1983
     },
     {
      "name": "Canada",
      "year": 1998
     },
      {
        "name": "Germany",
        "year": 2003
      }
    ]
  },
  {
    "name": "Sajal",
    "email": "sajal@something.com",
    "age": 65,
    "countries": [
      {
        "name": "Belgium",
        "year": 1994
      },
      {
        "name": "Hungary",
        "year": 2001
      },
      {
        "name": "Chile",
        "year": 2013
      }
    ]
  },
  {
    "name": "Adam",
    "email": "adam@something.com",
    "age": 72,
    "countries": [
      {
        "name": "France",
        "year": 1988
      },
      {
        "name": "Brazil",
        "year": 1998
      },
      {
        "name": "Poland",
        "year": 2002
      }
    ]
  },
  {
    "name": "Monty",
      "email": "monty@something.com",
      "age": 77,
      "countries": [
       {
         "name": "Spain",
         "year": 1982
       },
       {
         "name": "Australia",
         "year": 1996
       },
       {
         "name": "Germany",
         "year": 1987
       }
      ]
    }
  ]
}
```

file_get_contents() function

This function is used to read the file into PHP code.

Syntax:

file_get_contents(path, file_name)

- *file_name* is the name of the file and *path* is the location to be checked.
- Use **json_decode()** function to decode to JSON file into array to display it.

It is used to convert the JSON into an array.

Syntax:

```
json_decode($json_object, true)
```

Parameters

- $json_object is the file object to be read.

Example

The following is the PHP code to parse JSON file.

```
<?php
$people_json = file_get_contents('data.json');
$decoded_json = json_decode($people_json, true);
$customers = $decoded_json['customers'];
foreach($customers as $customer) {
  $name = $customer['name'];
  $countries = $customer['countries'];
  foreach($countries as $country) {
      echo $name.' visited '.$country['name'].' in '.$country['year'].'.'.'<br>';
  }}?>
```

Output:

```
Andrew visited Italy in 1983.
Andrew visited Canada in 1998.
Andrew visited Germany in 2003.
Sajal visited Belgium in 1994.
Sajal visited Hungary in 2001.
Sajal visited Chile in 2013.
Adam visited France in 1988.
Adam visited Brazil in 1998.
Adam visited Poland in 2002.
Monty visited Spain in 1982.
Monty visited Australia in 1996.
Monty visited Germany in 1987.
```

2.5 Classes and Objects in PHP

Object-oriented programming is a programming model organized around **Object rather than the actions** and **data rather than logic**. Basic terminology of OOP to revise are as follows:

- **Class –** This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.
- **Object –** An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable –** These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member function –** These are the function defined inside a class and are used to access object data.
- **Inheritance –** When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent class –** A class that is inherited from by another class. This is also called a base class or super class.
- **Child Class –** A class that inherits from another class. This is also called a subclass or derived class.
- **Polymorphism –** This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it take different number of arguments and can do different task.

- **Overloading –** a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
- **Data Abstraction –** Any representation of data in which the implementation details are hidden (abstracted).
- **Encapsulation –** refers to a concept where we encapsulate all the data and member functions together to form an object.
- **Constructor –** refers to a special type of function which will be called automatically whenever there is an object formation from a class.
- **Destructor –** refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

Defining PHP Classes

The general form for defining a new class in PHP is as follows –

```
<?php
  class phpClass {
    var $var1;
    var $var2 = "constant string";
    function myfunc ($arg1, $arg2) {
      [..]
    }
    [..]
  }?>
```

Here is the description of each line –

- The special form class, followed by the name of the class that you want to define.
- A set of braces enclosing any number of variable declarations and function definitions.
- Variable declarations start with the special form var, which is followed by a conventional $ variable name; they may also have an initial assignment to a constant value.
- Function definitions look much like standalone PHP functions but are local to the class and will be used to set and access object data.

Example

```
<?php
  class Books {
    /* Member variables */
    var $price;
    var $title;
    /* Member functions */
    function setPrice($par){
      $this->price = $par;
    }
    function getPrice(){
      echo $this->price ."<br/>";
    }
    function setTitle($par){
      $this->title = $par;
    }
    function getTitle(){
      echo $this->title ." <br/>";
    }
  }
?>
```

The variable $this is a special variable and it refers to the same object ie. itself.

Creating Objects in PHP

Once you defined your class, then you can create as many objects as you like of that class type. Following is an example of how to create object using new operator.

```
$physics = new Books;
$maths = new Books;
$chemistry = new Books;
```

Here we have created three objects and these objects are independent of each other and they will have their existence separately. Next we will see how to access member function and process member variables.

Calling Member Functions

After creating your objects, you will be able to call member functions related to that object. One member function will be able to process member variable of related object only.

Following example shows how to set title and prices for the three books by calling member functions.

```
$physics->setTitle( "Physics for High School" );
$chemistry->setTitle( "Advanced Chemistry" );
$maths->setTitle( "Algebra" );
$physics->setPrice( 10 );
$chemistry->setPrice( 15 );
$maths->setPrice( 7 );
```

Now you call another member functions to get the values set by in above example.

```
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();
$physics->getPrice();
$chemistry->getPrice();
```

```
$maths->getPrice();
```

This will produce the following result –

```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

Constructor Functions

Constructor Functions are special type of functions which are called automatically whenever an object is created. So we take full advantage of this behaviour, by initializing many things through constructor functions.

PHP provides a special function called **__construct()** to define a constructor. You can pass as many as arguments you like into the constructor function.

Following example will create one constructor for Books class and it will initialize price and title for the book at the time of object creation.

```
function __construct( $par1, $par2 ) {
$this->title = $par1;
$this->price = $par2;
}
```

Now we don't need to call set function separately to set price and title. We can initialize these two member variables at the time of object creation only. Check following example below –

```
$physics = new Books( "Physics for High School", 10 );
$maths = new Books ( "Advanced Chemistry", 15 );
$chemistry = new Books ("Algebra", 7 );
/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();
$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

This will produce the following result –

```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

Destructor

Like a constructor function you can define a destructor function using function __destructor(). You can release all the resources with-in a destructor.

```
<?php
class MyClass
{
// Constructor
public function __construct()
        {
        echo 'The class "' . __CLASS__ . '" was initiated!<br>';
         }
 // Destructor
public function __destruct()
        {
        echo 'The class "' . __CLASS__ . '" was destroyed.<br>';
        }
}
// Create a new object
$obj = new MyClass;
// Destroy the object
unset($obj);
// Output a message at the end of the file
echo "The end of the file is reached.";
?>
```

Output

```
The class "MyClass" was initiated!
The class "MyClass" was destroyed.
The end of the file is reached.
```

2.6 Regular Expressions and Exception Handling

Regular Expression

Regular expressions commonly known as a regex (regexes) are a sequence of characters describing a special search pattern in the form of text string. They are basically used in programming world algorithms for matching some loosely defined patterns to achieve some relevant tasks. Some times regexes are understood as a mini programming language with a pattern notation which allows the users to parse text strings. The exact sequence of characters are unpredictable beforehand, so the regex helps in fetching the required strings based on a pattern definition.

Regular Expression is a compact way of describing a string pattern that matches a particular amount of text. As you know, PHP is an open-source language commonly used for website creation, it provides regular expression functions as an important tool. Like PHP, many other programming languages have their own implementation of regular expressions. This is the same with other applications also, which have their own support of regexes having various syntaxes. Many available modern languages and tools apply regexes on very large files and strings. Let us look into some of the advantages and uses of regular expressions in our applications.

Advantages and uses of Regular expressions:

- Regular expressions help in validation of text strings which are of programmer's interest.
- It offers a powerful tool for analyzing, searching a pattern and modifying the text data.
- It helps in searching specific string pattern and extracting matching results in a flexible manner.
- It helps in parsing text files looking for a defined sequence of characters for further analysis or data manipulation.
- With the help of in-built regexes functions, easy and simple solutions are provided for identifying patterns.
- It effectively saves a lot of development time, which are in search of specific string pattern.
- It helps in important user information validations like email address, phone numbers and IP address.
- It helps in highlighting special keywords in a file based on search result or input.
- It helps in identifying specific template tags and replacing those data with the actual data as per the requirement.
- Regexes are very useful for creation of HTML template system recognizing tags.
- Regexes are mostly used for browser detection, spam filtration, checking password strength and form validations.

The following table shows some regular expressions and the corresponding string which matches the regular expression pattern.

Regular Expression	Matches
Geeks	The string *"Geeks"*
^geeks	The string which starts with *"geeks"*
geeks$	The string which have *"geeks"* at the end.
^geeks$	The string where *"geeks"* is alone on a string.
[abc]	a, b, or c
[a-z]	Any lowercase letter
[^A-Z]	Any letter which is NOT a uppercase letter
(gif|png)	Either *"gif"* or *"png"*
[a-z]+	One or more lowercase letters
^[a-zA-Z0-9]{1, }$	Any word with at least one number or one letter
([ax])([by])	ab, ay, xb, xy
[^A-Za-z0-9]	Any symbol other than a letter or other than number
([A-Z]{3}|[0-9]{5})	Matches three letters or five numbers

Note: Complex search patterns can be created by applying some basic regular expression rules. Even many arithmetic operators like +, ^, – are used by regular expressions for creating little complex patterns.

Operators in Regular Expression:

Operator	Description
^	It denotes the start of string.
$	It denotes the end of string.
.	It denotes almost any single character.
()	It denotes a group of expressions.
[]	It finds a range of characters for example [xyz] means x, y or z .
[^]	It finds the items which are not in range for example [^abc] means NOT a, b or c.
– (dash)	It finds for character range within the given item range for example [a-z] means a through z.
| (pipe)	It is the logical OR for example x | y means x OR y.
?	It denotes zero or one of preceding character or item range.
*	It denotes zero or more of preceding character or item range.
+	It denotes one or more of preceding character or item range.
{n}	It denotes exactly n times of preceding character or item range for example n{2}.
{n, }	It denotes atleast n times of preceding character or item range for example n{2, }.
{n, m}	It denotes atleast n but not more than m times for example n{2, 4} means 2 to 4 of n.
\	It denotes the escape character.

Special Character Classes in Regular Expressions:

Special Character	Meaning
\n	It denotes a new line.
\r	It denotes a carriage return.
\t	It denotes a tab.
\v	It denotes a vertical tab.
\f	It denotes a form feed.
\xxx	It denotes octal character xxx.
\xhh	It denotes hex character hh.

Shorthand Character Sets: Let us look into some shorthand character sets available.

Shorthand	Meaning
\s	Matches space characters like space, newline or tab.
\d	Matches any digit from 0 to 9.
\w	Matches word characters including all lower and upper case letters, digits and underscore.

Predefined functions or Regex library: Let us look into the quick cheat sheet of pre-defined functions for regular expressions in PHP. PHP provides the programmers to many useful functions to work with regular expressions.

The below listed in-built functions are case-sensitive.

Function	Definition
preg_match()	This function searches for a specific pattern against some string. It returns true if pattern exists and false otherwise.
preg_match_all()	This function searches for all the occurrences of string pattern against the string. This function is very useful for search and replace.
ereg_replace()	This function searches for specific string pattern and replace the original string with

	the replacement string, if found.
eregi_replace()	The function behaves like ereg_replace() provided the search for pattern is not case sensitive.
preg_replace()	This function behaves like ereg_replace() function provided the regular expressions can be used in the pattern and replacement strings.
preg_split()	The function behaves like the PHP split() function. It splits the string by regular expressions as its parameters.
preg_grep()	This function searches all elements which matches the regular expression pattern and returns the output array.
preg_quote()	This function takes string and quotes in front of every character which matches the regular expression.
ereg()	This function searches for a string which is specified by a pattern and returns true if found, otherwise returns false.
eregi()	This function behaves like ereg() function provided the search is not case sensitive.

Note:

- By default, regular expressions are case sensitive.
- There is a difference between strings inside single quotes and strings inside double quotes in PHP. The former are treated literally, whereas for the strings inside double-quotes means the content of the variable is printed instead of just printing their names.

- **preg_match () function**

This function searches string for pattern, returns true if pattern exists, otherwise returns false. Usually search starts from beginning of subject string. The optional parameter offset is used to specify the position from where to start the search.

Syntax:

```
int preg_match( $pattern, $input, $matches, $flags, $offset )
```

Parameters:

- **pattern:** This parameter holds the pattern to search for, as a string.
- **input:** This parameter holds the input string.
- **matches:** If matches exists then it contains results of search. The $matches[0] will contain the text that matched full pattern, $matches[1] will contain the text that matched the first captured parenthesized subpattern, and so on.
- **flags:** The flags can be following flags:
  - **PREG_OFFSET_CAPTURE:** If this flag is passed, for every match the append string offset will be returned.
  - **PREG_UNMATCHED_AS_NULL:** If this flag is passed, subpatterns which are not matched reports as NULL; otherwise they reports as empty string.

- **offset:** Usually, search starts from the beginning of input string. This optional parameter offset is used to specify the place from where to start the search (in bytes).

Return value: It returns true if pattern exists, otherwise false.

Example : Validating String with alphabets

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
  $ErrMsg = "Only alphabets and whitespace are allowed.";
       echo $ErrMsg;
} else {
  echo $name;
}
```

Example : Validating numeric

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
  $ErrMsg = "Only numeric value is allowed.";
  echo $ErrMsg;
} else {
  echo $mobileno;
}
```

Example: Validating Email

```
$email = $_POST ["Email"];
$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) ){
  $ErrMsg = "Email is not valid.";
      echo $ErrMsg;
} else {
  echo "Your valid email address is: " .$email;
}
```

Example: Validate URL

```
$websiteURL = $_POST["website"];
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$websiteURL)) {
 $websiteErr = "URL is not valid";
 echo $websiteErr;
} else {
  echo "Website URL is: " .$websiteURL;
}
```

- **preg_match_all () function**

The preg_match_all() function searches for all the matches to a regular expression in a string.

Unlike the preg_match() function that stops searching when it finds the first match, the preg_match_all() function continues searching for the next matches till the end of the string.

Syntax

```
preg_match_all($pattern, $subject, &$matches = null, $flags = 0, $offset = 0):
int|false|null
```

Parameters:

- **$pattern** is a string that specifies the pattern to search.
- **$subject** is the input string to match the pattern.
- **$matches** is a multi-dimensional array that contains all the matches.
- $flags is a combination of the flags PREG_PATTERN_ORDER, PREG_SET_ORDER, PREG_OFFSET_CAPTURE, and PREG_UNMATCHED_AS_NULL. By default, the $flags is PREG_PATTERN_ORDER if you skip it.
- $offset is an integer that specifies the position from which the search starts. By default, the preg_match_all() function starts searching from the beginning of the string.

The preg_match_all() function returns a number that specifies the number of full pattern matches. If there is no match, the preg_match_all() function returns zero. In case of failure, it returns false.

1) Using the PHP preg_match_all() function to match numbers in a string example

```
<?php
$pattern = '/\d+/';
$str = 'PHP 1.0 released in 1995';
if (preg_match_all($pattern, $str, $matches)) {
   print_r($matches);
}
```

Output:

```
Array
(
   [0] => Array
      (
         [0] => 1
         [1] => 0
         [2] => 1995
      )
)
```

2) Using the preg_match_all() function with flags parameters

```
<?php
$pattern = '/\b([a-zA-Z])\w+\b/';
$str = 'Alice, Bob, Peter';
if (preg_match_all($pattern, $str, $matches)) {
   print_r($matches);
}
```

Output:

```
Array
(
   [0] => Array
      (
         [0] => Alice
         [1] => Bob
```

```
        [2] => Peter
    )
  [1] => Array
    (
        [0] => A
        [1] => B
        [2] => P
    )
)
```

The $matches array contains the full pattern matches in the first element and the capturing groups in the second element. It returns the same result as if you use the PREG_PATTERN_ORDER flag.

If you want to group each set of matches in an array element, you can use the PREG_SET_ORDER flag.

```
<?php

$pattern = '/\b([a-zA-Z])\w+\b/';
$str = 'Alice, Bob, Peter';

if    (preg_match_all($pattern,    $str,    $matches,
PREG_SET_ORDER)) {
  print_r($matches);
}
```

```
Output:
Array
(
  [0] => Array
    (
        [0] => Alice
        [1] => A
    )
  [1] => Array
    (
        [0] => Bob
        [1] => B
    )
  [2] => Array
    (
        [0] => Peter
        [1] => P
    )
)
```

The $flags can also be:

- PREG_OFFSET_CAPTURE returns the offset of the match together with the matched string.
- PREG_UNMATCHED_AS_NULL returns NULL instead of an empty string if no match is found for the subpatterns a.k.a capturing groups.

```
<?php

$pattern = '/\b([a-zA-Z])\w+\b/';
$str = 'Alice, Bob, Peter';
```

```
Output:
Array
(
  [0] => Array
```

```
if (preg_match_all($pattern, $str, $matches,
PREG_SET_ORDER | PREG_OFFSET_CAPTURE)) {
    print_r($matches);
}
```

```
        (
            [0] => Array
                (
                    [0] => Alice
                    [1] => 0
                )
            [1] => Array
                (
                    [0] => A
                    [1] => 0
                )
        )
    [1] => Array
        (
            [0] => Array
                (
                    [0] => Bob
                    [1] => 7
                )
            [1] => Array
                (
                    [0] => B
                    [1] => 7
                )
        )
    [2] => Array
        (
            [0] => Array
                (
                    [0] => Peter
                    [1] => 12
                )
            [1] => Array
                (
                    [0] => P
                    [1] => 12
                )
        )
)
```

preg_replace() funation

The preg_replace() function searches for matches and replaces them with a pattern.

Syntax:

```
preg_replace($pattern, $replacement, $subject, $limit = -1, &$count = null): string
```

Parameters:

- $pattern is a regular expression to match.
- $replacement is a string to replace. It may contain the backreferences.
- $subject is a string to search and replace.
- $limit is the maximum possible replacement for the pattern. By default, it is -1, which is unlimited.
- $count stores the number of replacements.

If the $subject matches the $pattern, the preg_replace() function replaces the match with the $replacement and returns it.

1) Using the PHP preg_replace() function to change the date format

The following example uses the preg_replace() function to change the date string from dd-mm-yyyy to mm-dd-yyyy format:

```
<?php
$pattern = '/(\d{2})-(\d{2})-(\d{4})/';
$replacement = '\2-\1-\3';
$str = '25-12-2021';
echo preg_replace($pattern, $replacement, $str);
?>
```

```
Output:
12-25-2021
```

The following regular expression matches the date in the dd-mm-yyyy format:

```
/(\d{2})-(\d{2})-(\d{4})/
```

It consists of three capturing groups for day (\d{2}), month (\d{2}), and year (\d{4}).

The replacement string contains three backreferences \2 for the second capturing group which is month, \1 for the first capturing group which is day, and \3 for the third capturing group which is the year.

Therefore, the preg_replace() changes the format of the date from dd-mm-yyyy to mm-dd-yyyy.

2) Using the PHP preg_replace() function to remove the excessive whitespace

```
<?php
$str = 'PHP   is    awesome';
echo preg_replace('/\s\s+/', ' ', $str);
?>
```

```
Output:

PHP is awesome
```

The pattern /\s\s+/ matches one or more spaces.

PHP preg_replace() function with arrays

The preg_replace() function also accepts the $pattern, $replacement, and $subject as arrays:

preg_replace(string|array $pattern, string|array $replacement, string|array $subject, int $limit = -1, int &$count = null): string|array|null

If both $pattern and $replacement are arrays and $subject is a string, the preg_replace() function will replace each pattern in the $pattern array with the replacement in the $replacement array.

If the $replacement array has fewer elements than the $pattern array, the preg_replace() function wil replace extra patterns with an empty string.

If the $subject is an array, the preg_replace() function searches and replaces each string in the array and returns the result as an array.

Example:

```
<?php
  $date = 'May 29, 2020';
  $pattern = '/(\w+) (\d+), (\d+)/i';
  $replacement = '${1} 5,$3';

  //display the result returned by preg_replace
  echo preg_replace($pattern, $replacement, $date);
?>
```

Output:

```
May 5, 2020
```

Unit – 3 PHP Interaction with Database

(MySQL)

For developing database driven web applications PHP is often used in conjunction with MySQL. You require some basic knowledge of PHP for trying out these commands.

3.1 Create Database, Create Table, Table Handling Insert, Update, Delete Operations

3.2 Querying database using PHP: Select, Insert, Update, Delete, Where, order by

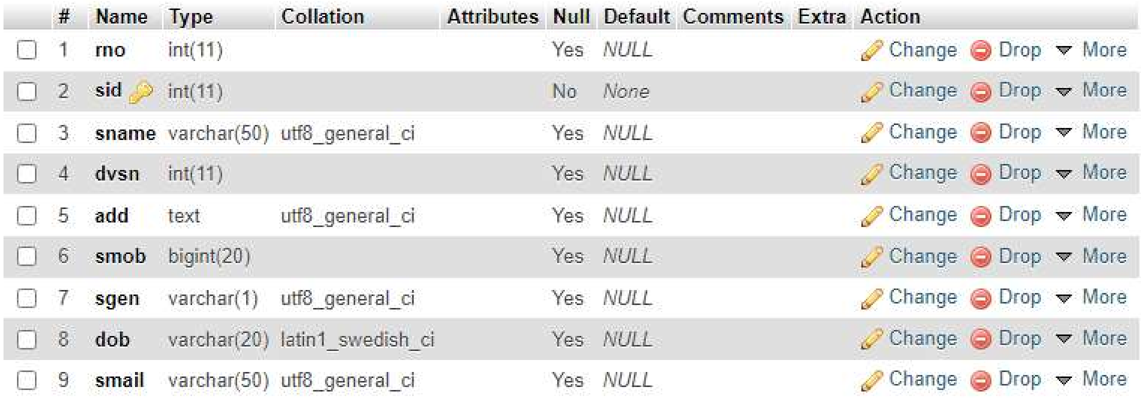
3.3 Processing Search query in background using AJAX

AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.

On most of the sites, we can see there is one search bar on the site where we can type in search content and get a search result with page refreshing. A live search is a search feature that shows the result instantly while you start typing some characters in the search input box. You can see the search results like the Google Search Autocomplete, Facebook, and Twitter. They have amazing live search functionality. It makes it simpler for the clients to discover what they are looking for. It delivers the result without page refresh. This functionality can be done using Ajax with jQuery.

With the assistance of jQuery we can utilize Ajax http dom work, with the assistance of this capacity, it searches for information on the server side and sends back outcome to the front end website page without a page refresh. These are the step-by-step process to live search data using Ajax -

Step – 1 Creation of DB table



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	rno	int(11)			Yes	NULL			Change Drop More
☐	2	sid 🔑	int(11)			No	None			Change Drop More
☐	3	sname	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More
☐	4	dvsn	int(11)			Yes	NULL			Change Drop More
☐	5	add	text	utf8_general_ci		Yes	NULL			Change Drop More
☐	6	smob	bigint(20)			Yes	NULL			Change Drop More
☐	7	sgen	varchar(1)	utf8_general_ci		Yes	NULL			Change Drop More
☐	8	dob	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop More
☐	9	smail	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More

Step – 2 Script for DB connection

```
<?php
$con=mysqli_connect("localhost","root","XXXXX","ty2022");
if (!($con))
        echo "Database Server Could not connected..".mysqli_error($con);?>
```

Step – 3 PHP Interface file

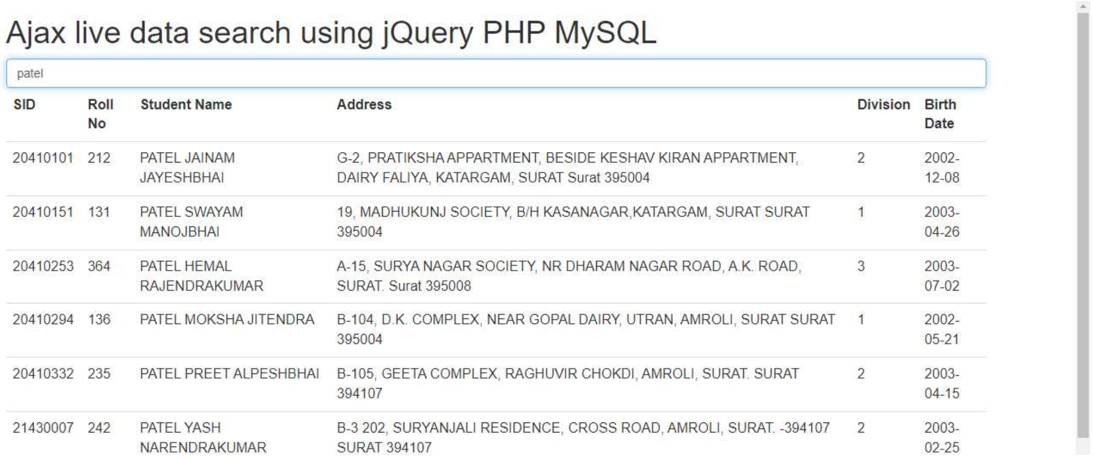
```
<?php
  include("config.inc");
?>
<html>
<head>
  <title>Ajax live data search using jQuery, PHP, MySQL</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"
rel="stylesheet">
  <script>
  $(document).ready(function(){
    load_data();
    function load_data(query)
    {
      $.ajax({
      url:"search.php",
      method:"POST",
      data:{query:query},
      success:function(data)
      {
        $('#result').html(data);
      }
      });
    }
    $('#search').keyup(function(){
    var search = $(this).val();
    if(search != '')
    {
      load_data(search);
    }
    else
    {
      load_data();
    }
    });
  });
  </script>
</head>
<body>
```

```
<div class="container-fluid">
<div class="content-wrapper">
  <div class="container">
    <h1>Ajax live data search using jQuery PHP MySQL</h1>
    <div class="row">
    <div class="col-xs-12">
      <input type="text" name="search" id="search" placeholder="Search" class="form-control" />
      <!-- Following block will get filled with the data as of keyup event executes on every key  stroke--
>
      <div id="result">

      </div>
    </div>
    </div>
  </div>
</div>
</div>
</body>
</html>
```

Step – 4 PHP Backend file for performing Search

```
<?php
require ('config.inc');
$return = '';
if(isset($_POST["query"]))
  {
  $search = mysqli_real_escape_string($con, $_POST["query"]);
  $query = "SELECT * FROM ty2022  WHERE sid  LIKE '%".$search."%' OR sname LIKE '%".$search."%';";
  }
else
  {
  $query = "SELECT * FROM ty2022;";
  }
$result = mysqli_query($con, $query);
if(mysqli_num_rows($result) > 0)
{
  $return .='
  <div class="table-responsive">
  <table class="table table bordered">
  <tr>
    <th>SID</th>
    <th>Roll No</th>
    <th>Student Name</th>
    <th>Address</th>
    <th>Division</th>
    <th>Birth Date</th>
  </tr>';
  while($ty = mysqli_fetch_row($result))
  {
    $return .= '
    <tr>
    <td>'.$ty[1].'</td>
    <td>'.$ty[0].'</td>
    <td>'.$ty[2].'</td>
    <td>'.$ty[4].'</td>
    <td>'.$ty[3].'</td>
    <td>'.$ty[7].'</td>
    </tr>';
  }
  echo $return;
  }
else
{
  echo 'No results found for given SID or Student Names!';
}
?>
```

Output



Ajax live data search using jQuery PHP MySQL

patel

SID	Roll No	Student Name	Address	Division	Birth Date
20410101	212	PATEL JAINAM JAYESHBHAI	G-2, PRATIKSHA APPARTMENT, BESIDE KESHAV KIRAN APPARTMENT, DAIRY FALIYA, KATARGAM, SURAT Surat 395004	2	2002-12-08
20410151	131	PATEL SWAYAM MANOJBHAI	19, MADHUKUNJ SOCIETY, B/H KASANAGAR,KATARGAM, SURAT SURAT 395004	1	2003-04-26
20410253	364	PATEL HEMAL RAJENDRAKUMAR	A-15, SURYA NAGAR SOCIETY, NR DHARAM NAGAR ROAD, A.K. ROAD, SURAT. Surat 395008	3	2003-07-02
20410294	136	PATEL MOKSHA JITENDRA	B-104, D.K. COMPLEX, NEAR GOPAL DAIRY, UTRAN, AMROLI, SURAT SURAT 395004	1	2002-05-21
20410332	235	PATEL PREET ALPESHBHAI	B-105, GEETA COMPLEX, RAGHUVIR CHOKDI, AMROLI, SURAT. SURAT 394107	2	2003-04-15
21430007	242	PATEL YASH NARENDRAKUMAR	B-3 202, SURYANJALI RESIDENCE, CROSS ROAD, AMROLI, SURAT. -394107 SURAT 394107	2	2003-02-25

Unit – 4 PHP – Python Integration

PHP and Python are two of the most popular and influential programming languages. They both have their strengths and weaknesses and share many standard software development capabilities. Although **Rasmus Lerdorf** developed **PHP** as a web language, and **Guido van Rossum** created **Python** as a general-purpose programming language, technological advances have made both excellent web development candidates. As with any technology decision, choosing the best one for your next web development project ultimately comes down to your requirements and resources. Assessing each language and comparing their strengths across several core elements can help you make the correct choice.

Development Environment (PHP < Python)

Language Complexity (PHP > Python)

Extendibility (PHP > Python)

Security (PHP > Python)

Scalability and Adaptability (=)

Documentation and Community Support(=)

4.1 Executing Python script using PHP

PHP (which is a recursive acronym for PHP Hypertext Preprocessor) is one of the most widely used web development technologies in the world. PHP code used for developing websites and web applications can be embedded directly along with the rest of the HTML markup. This, along with a rich number of libraries to perform system tasks, or to integrate with other application, makes PHP the go-to language for the Internet.

While there are plenty of libraries (or modules) in PHP, there can be cases when a module for a certain task is either missing or is not properly implemented; while the same module is available in Python.

Apart from this, for certain scripting tasks the user might find Python more suitable, while he finds PHP suitable for the rest of his codebase.

4.1.1 Calling Python Script using echo(), escapeshellcmd(), shell_exec() methods

escapeshellcmd()

escapeshellcmd() escapes any characters in a string that might be used to trick a shell command into executing arbitrary commands. This function should be used to make sure that any data coming from user input is escaped before this data is passed to the exec() or system() functions, or to the backtick operator.

The command line is a dangerous place for unescaped characters. Never pass unmodified user input to one of PHP's shell-execution functions. Always escape the appropriate characters in the command and the arguments. This is crucial. It is unusual to execute command lines that are coming from web forms and not something we recommend lightly. However, sometimes you need to run an external program, so escaping commands and arguments is useful.

Following characters are preceded by a backslash:

#&;`|*?~<>^()[]{}$\, \x0A and \xFF. ' and " are escaped only if they are not paired. In Windows, all these characters plus % and ! are replaced by a space instead.

Syntax

string **escapeshellcmd** (string command)

Parameter

Command- The command that will be escaped.

Return Value

The function/method returns the string after escaping the described chraters as backtick operators

Example

```
<?php
$command = substr("\\%!** Hello World",0);
$escaped_command = escapeshellcmd($command);
 echo ($escaped_command);
?>
```

Output

```
^\^%^!^*^* Hello World
```

shell_exec ()

The shell_exec() function is an inbuilt function in PHP which is used to execute the commands via shell and return the complete output as a string. The shell_exec is an alias for the backtick operator, for those used to unix. If the command fails return NULL and the values are not reliable for error checking.

Syntax:

string shell_exec($cmd)

Parameters:

This function accepts single parameter *$cmd* which is used to hold the command that will be executed.

Return Value:

This function returns the executed command or NULL if an error occurred.

Note:

This function is disabled when PHP is running in safe mode.

Example:

```
<?php
// shell_exec test
if (shell_exec('mkdir test'))
```

```
   echo "test directory created in present working directory";
else
   echo "shell_exec failed..!";
//resulting output is found in present working directory if there is no test directory present
//once the test directory is created on second execution teh command could not executed
?>
```

```
<?php
$output = shell_exec('dir');
echo $output;
?>
```

Executing Python Script Using PHP

Using above 2 methods along with echo within PHP, one can execute the Python Script from the web server like Apache. We have to create 2 files for the same purpose. One file will consist the python script where as the other will consisting of above methods under the PHP script to execute the mentioned python file.

Check the example below:

Example

test.py

```
#!c:/python310 python
print("Welcome to Python Zone")
```

test.php

```
<?php
$command = escapeshellcmd('python test.py');
$output = shell_exec($command);
echo $output;
?>
```

Output

```
Welcome to Python Zone
```

Explanation

test.py: in this file the first line mentions from where the python executable file can locate so that the python code can get execute there after. Whereas the second line is just a print statement for displaying a string.

test.php: in php file the first line is using the escapeshellcmd function accepts the command for executing the python file test.py. The second line extract the generated output of first command in a PHP variable. And third is just used to show the resulted variable on the browser by just passing the URL of PHP file to execute it.

4.2 Executing PHP Script using Python

4.2.1 Sub-process Module in Python:

The subprocess module present in Python(both 2.x and 3.x) is used to run new applications or programs through Python code by creating new processes. It also helps to obtain the input/output/error pipes as well as the exit codes of various commands.

A running program is called a **process**. Each process has its own system state, which includes memory, lists of open files, a program counter that keeps track of the instruction being executed, and a call stack used to hold the local variables of functions.

Normally, a process executes statements one after the other in a single sequence of control flow, which is sometimes called the main thread of the process. At any given time, the program is only doing one thing.

A program can create new processes using library functions such as those found in the os or subprocess modules such as **os.fork()**, **subprocess.Popen()**, etc. However, these processes, known as **subprocesses**, run as completely independent entities-each with their own private system state and main thread of execution.

Because a subprocess is independent, it executes concurrently with the original process. That is, the process that created the subprocess can go on to work on other things while the subprocess carries out its own work behind the scenes.

Subprocess Module

The subprocess module allows us to:

- spawn new processes
- connect to their input/output/error pipes
- obtain their return codes

4.2.1.1Methods: Check-call(), check-output(), declode(), Popen(), Communicate(), split()

Popen() function

The Underlying process creation and management in sub-process module is handled by the popen class. It offers a lot of flexibility so that developers are able to handle the less common cases not covered by the convenience functions.The P in the name of the Popen() function stands for process.

If you have multiple instances of an application open, each of those instances is a separate process of the same program.

For example, if you open multiple windows of your web browser at the same time, each of those windows is a different process of the web browser program

Syntax:

```
class subprocess.Popen(args, bufsize=-
1, executable=None, stdin=None, stdout=None, stderr=None, preexec_fn=None, close_fds=True, shell=F
alse, cwd=None, env=None, universal_newlines=None, startupinfo=None, creationflags=0, restore_signa
ls=True, start_new_session=False, pass_fds=(), *, group=None, extra_groups=None, user=None, umask=
- 1, encoding=None, errors=None, text=None, pipesize=- 1)
```

But, the general syntax to use subprocess.Popen

```
subprocess.Popen(cmd,shell=True/False,stdout=subprocess.PIPE,stderr=subprocess.PIPE)
```

In this syntax we are storing the command output (stdout) and command error (stderr) in the same variable

Example1:

```
import subprocess
proc = subprocess.Popen("php demo1.php", shell=True, stdout=subprocess.PIPE)
script_response = proc.stdout.read()
print(script_response)
```

Let's assume that demo1.php has 1 line code <?php echo "hello world"; ?>. it will return hello world in it's output.

Example:

```
#!/usr/bin/env python3
import subprocess
# Define command as string
cmd = 'ls -ltr'
# Use shell to execute the command and store it in sp variable
sp = subprocess.Popen(cmd,shell=True)
# Store the return code in rc variable
rc=sp.wait()
# Print the content of sp variable
print(sp)
```

call() function

Subprocess in Python has a call() method that is used to initiate a program.

The subprocess module provides another function that makes process spawning a safer operation than using Popen().

The subprocess.call() function waits for the called command/program to finish reading the output.

Syntax:

```
subprocess.call(args, *, stdin=None, stdout=None, stderr=None, shell=False, cwd=None, timeout=None,
**other_popen_kwargs)
```

It supports the same arguments as the Popen() constructor, such as shell, executable, and cwd, but this time, your script will wait for the program to complete and populate the return code without the need to communicate().

You can fetch the exit status using Popen.returncode

To suppress stdout or stderr, supply a value of subprocess. DEVNULL which indicates that the special file os.devnull will be used.

Example:

```
import subprocess
# if the script don't need output.
subprocess.call("php demo1.php")
```

Example:

```
import subprocess

subprocess.call(["ls", "-l"])
```

check_call() function

This is another function which is part of subprocess module which can run command with arguments. check_call will wait for command execution to complete.

check_call() will wait for command execution to complete.

If the execution is successful then the function will return zero then return, otherwise raise CalledProcessError.

The CalledProcessError object will have the return code in the returncode attribute.

Syntax:

```
subprocess.check_call(args, *, stdin=None, stdout=None, stderr=None, shell=False)
```

Parameters

args: It is the command that you want to execute. You can pass multiple commands by separating them with a semicolon (;)

stdin: This refers to the standard input stream's value passed as (os.pipe())

stdout: It is the standard output stream's obtained value

stderr: This handles any errors that occurred from the standard error stream

shell: It is the boolean parameter that executes the program in a new shell if kept true

Example:

```
import subprocess
s = subprocess.check_call("gcc HelloWorld.c -o out1;./out1", shell = True)
print(", return code", s)
```

Note: Do not use stdout=PIPE or stderr=PIPE with call and check_call function as that can deadlock based on the child process output volume. Use Popen with the communicate() method when you need pipes.

check_output() function

The save process output or stdout allows you to store the output of a code directly in a string with the help of the check_output function.

The subprocess.check_output() is similar to subprocess.run(check=True)

By default, this function will return the data as encoded bytes so you can use text=True or universal_newlines=True to get string value as output

This function will run command with arguments and return its output.

If the return code was non-zero it raises a CalledProcessError.

The CalledProcessError object will have the return code in the returncode attribute and any output in the output attribute.

Syntax:

```
subprocess.check_output(args, *, stdin=None, stderr=None, shell=False, universal_newlines=False
                                        OR
subprocess.check_output(args, *, stdin=None, stderr=None, shell=False, cwd=None, encoding=None, errors=None, universal_newlines=None, timeout=None, text=None, **other_popen_kwargs)
```

Parameters:

args The command to be executed. Several commands can be passed as a string by separated by “;”.

stdinValue of standard input stream to be passed as pipe(os.pipe()).

stdoutValue of output obtained from standard output stream.

stderrValue of error obtained(if any) from standard error stream.

shellboolean parameter.If True the commands get executed through a new shell environment.

universal_newlines Boolean parameter.If true files containing stdout and stderr are opened in universal newline mode.

Example:

```
import subprocess
s = subprocess.check_output(["echo", "Hello World!"])
print(s)
```

Output:

```
b’Hello World’!
```

communicate() function

In subprocess,Popen() can interact with the three channels and redirect each stream to an external file, or to a special value called **PIPE**. An additional method, called communicate(), is used to read from the stdout and write on the stdin.

The spawned processes can communicate with the operating system in three channels:

- Standard input (stdin)
- Standard output (stdout)
- Standard error (stderr)

The communicate() method can take input from the user and return both the standard output and the standard error.

Syntax:

```
Popen.communicate(input=None, timeout=None)
```

returns a tuple (stdout_data, stderr_data).

Example:

```
from subprocess import Popen, PIPE
process = Popen(['cat', 'example.py'], stdout=PIPE, stderr=PIPE)
stdout, stderr = process.communicate()
print(stdout)
```

Output:

```
b ‘ ‘
… program finished with exit code 0
Press enter to exit console
```

In the above code, the process.communicate() is the primary call that reads all the process’s inputs and outputs. The “stdout” handles the process output, and the “stderr” is for handling any sort of error and is written only if any such error is thrown.

split() function

Sometime, for example when you use **subprocess** you might want to run an external program avoiding the invocation of the shell. For that you need to have the external command in pieces, not in a single long string.

You could use the regular split method of strings with space as the delimiter, but that would not give you the right results if some of the pieces contain spaces.

For example if this is the command line:

```
'./bin/application --source /some/directory --target /other/dir --verbose -d --name "Foo Bar"'
```

The solution is to use the split method of the shlex module.In this example we use both. The str.split() and the shlex.split()

Example:

```
import shlex
```

```
cmd = './bin/application --source /some/directory --target /other/dir --verbose -d --name "Foo Bar"'
print(cmd.split(' '))
print(shlex.split(cmd))
```

Output:

```
['./bin/application', '--source', '/some/directory', '--target', '/other/dir', '--verbose', '-d', '--name',
'"Foo', 'Bar"']
['./bin/application', '--source', '/some/directory', '--target', '/other/dir', '--verbose', '-d', '--name', 'Foo
Bar']
```

decode() function

There are a couple of methods you can use to convert the byte into string format for subprocess.Popen output:

- decode("utf-8") (It Return a string decoded from the given bytes. Default encoding is 'utf-8')
- universal_newlines

In **decode**, errors may be given to set a different error-handling scheme. The default for errors is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace' and any other name registered via codes.register.error().

Syntax:

```
bytes.decode(encoding='utf-8', errors='strict')
bytearray.decode(encoding='utf-8', errors='strict')
```

Example:

```
s = subprocess.check_call("gcc HelloWorld.c -o out1;./out1", shell = True)
print(", return code", s)
print(s.decode("utf-8"))
```

here, it decode "s" to a normal string

Which subprocess module function should I use?

The functions run(), call(), check_call(), and check_output() are wrappers around the Popen class.

Using Popen directly gives more control over how the command is run, and how its input and output streams are processed.

Wait for the command to complete

Use subprocess.call or subprocess.run to run the command described by args. Wait for command to complete, then return the returncode attribute.

Use subprocess.Popen with wait() to wait for the command to complete

4.2.2 OS Module in Python:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

4.2.2.1write(), read(), close(), mkdir(), makedirs(), path, exists(), isfile(), join()

A file descriptor is small integer value that corresponds to a file or other input/output resource, such as a pipe or network socket. A File descriptor is an abstract indicator of a resource and act as handle to perform various lower level I/O operations like read, write, send etc.

For Example: Standard input is usually file descriptor with value 0, standard output is usually file descriptor with value 1 and standard error is usually file descriptor with value 2.

Further files opened by the current process will get the value 3, 4, 5 an so on.

Write()

os.write() method in Python is used to write a byte / string to the given file descriptor.

Note: os.write() method is intended for low-level operation and should be applied to a file descriptor as returned by os.open() or os.pipe() method.

Syntax:

```
os.write(fd, str)
```

Parameter:
fd: The file descriptor representing the target file.

str: A bytes-like object to be written in the file.

Return Type:

This method returns an integer value which represents the number of bytes actually written.

Example:

```
import os
path = "/home / phppython / myworks.txt"
fd = os.open(path, os.O_RDWR)
s = "myworks: A Computer science portal for works."
line = str.encode(s)
numBytes = os.write(fd, line)
print("Number of bytes written:", numBytes)
os.close(fd)
```

Output:

```
Number of bytes written: 51
```

Read()

os.read() method in Python is used to read at most n bytes from the file associated with the given file descriptor.

If the end of the file has been reached while reading bytes from the given file descriptor, os.read() method will return an empty bytes object for all bytes left to be read.

Note: os.read() method is intended for low-level operation and should be applied to a file descriptor as returned by os.open() or os.pipe() method.

Syntax:

```
os.read(fd, n)
```

Parameter:

fd: A file descriptor representing the file to be read.
n: An integer value denoting the number of bytes to be read from the file associated with the given file descriptor fd.

Return Type:

This method returns a bytestring which represents the bytes read from the file associated with the file descriptor fd.

Consider the below text as the content of the file named mywork.txt.

Example:

```
import os
path = "/home / phppython / myworks.txt"
fd = os.open(path, os.O_RDONLY)
# Number of bytes to be read
n = 51
readBytes = os.read(fd, n)
print(readBytes)
os.close(fd)
```

Output:

```
myworks: A Computer science portal for works'
```

close()

os.close() method in Python is used to close the given file descriptor, so that it no longer refers to any file or other resource and may be reused.
Syntax:

```
os.close(fd)
```

Parameter:
fd: A file descriptor, which is to be closed.

Return Type:

This method does not return any value

Example:

```
import os
path = "/ home / phppython / myworks.txt"
fd = os.open(path, os.O_WRONLY)
s = "myworks: A computer science portal for works"
# Convert string to bytes object
line = str.encode(s)
os.write(fd, line)
os.close(fd)
print("File descriptor closed successfully")
```

Output:

```
        File descriptor closed successfully
```

os.popen()/write()/read()/close() examples in short

os.popen() function opens a file or from the command specified, and it returns a file object which is connected to a pipe.

Example

```
import os
fd = "python.txt"
# popen() is similar to open()
file = open(fd, 'w')
file.write("This is awesome")
file.close()
file = open(fd, 'r')
text = file.read()
print(text)
# popen() provides gateway and accesses the file directly
file = os.popen(fd, 'w')
file.write("This is awesome")
# File not closed, shown in next function.
```

Output:

```
This is awesome
```

os.close()

This function closes the associated file with descriptor fr.

Example

```
import os
fr = "Python1.txt"
file = open(fr, 'r')
text = file.read()
print(text)
os.close(file)
```

Output:

```
Traceback (most recent call last):
  File "main.py", line 3, in
    file = open(fr, 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'Python1.txt'
```

Directory Handling

There are different methods available in the OS module for creating a directory. These are –

os.mkdir() and os.makedirs()

os.mkdir()

os.mkdir() method in Python is used to create a directory named path with the specified numeric mode. This method raises FileExistsError if the directory to be created already exists.

Example:

```
import os
os.mkdir("d:\\newdir")
```

It will create the new directory to the path in the string argument of the function in the D drive.

os.makedirs()

os.makedirs() method in Python is used to create a directory recursively. That means while making leaf directory if any intermediate-level directory is missing, os.makedirs() method will create them all.

Example:

```
#import the os module
from os import listdir, makedirs
#print the list of files and directories in current directory
print("Before adding directory")
print(listdir())
#create a directory
makedirs("./Educative/Leaf", 0o755)
print("After adding directory")
#print the list of files and directories in current directory
print(listdir())
print("Display inner directory")
print(listdir("./Educative"))
```

output:

```
Before adding directory
['__ed_script.sh', '__ed_stderr.txt', 'output', '__ed_mem.txt', 'main.py', '__ed_stdout.txt']
```

```
After adding directory
['Educative', '__ed_script.sh', '__ed_stderr.txt', 'output', '__ed_mem.txt', 'main.py', '__ed_stdout.txt']
Display inner directory
['Leaf']
```

In the code above,

- we import the os module which comes with methods like makedirs mkdir, listdir, etc.
- we create directories with the names Educative and Leaf in the current directory ./, and we provide mode as 0o755. This will first create the Educative directory and then create Leaf inside it.
- In the output, we can check that the directories Educative and Leaf are created.

exists(path)

Test whether a path exists. Returns False for broken symbolic links.

Example:

```
import os
imaginary = os.path.exists('path/to/file.html')
real = os.path.exists('c:/python/osmodule/main.py')
print(imaginary)
print(real)
```

output:

```
False
True
```

isfile() and isdir()

It Checks to see if the path is a file or if the path is a directory.

Example:

```
import os
contents = os.listdir()
for item in contents:
  if os.path.isdir(item):
    print(item + " is a directory")
  elif os.path.isfile(item):
    print(item + " is a file")
```

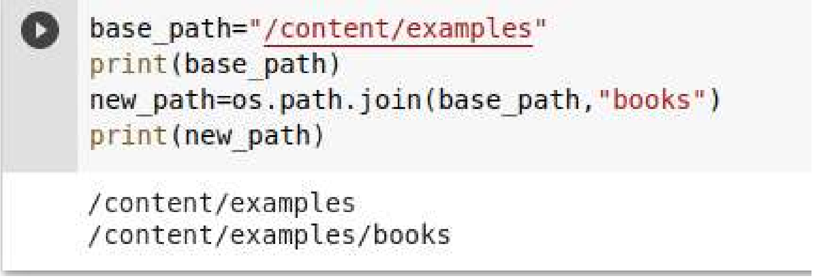
Output:

```
.idea is a directory
main.py is a file
renamed_file.py is a file
```

os.path.join() :

This method joins various path components with exactly one directory separator ("/") following each non-empty part except for the last path component. If the last path component is empty then a directory separator ("/") is put at the end. This method returns a string with the concatenated path.

Example1:



```
base_path="/content/examples"
print(base_path)
new_path=os.path.join(base_path,"books")
print(new_path)
```

```
/content/examples
/content/examples/books
```

Example2:

```
import os
good_vals = os.environ.get('homedrive')
joined = os.path.join(good_vals, '/index.html')
print(joined)
```

4.2.2.2isdir(), listdir(), walk(), chdir()

isdir()

OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. os.path module is sub module of OS module in Python used for common path name manipulation.

os.path.isdir() method in Python is used to check whether the specified path is an existing directory or not. This method follows symbolic link, that means if the specified path is a symbolic link pointing to a directory then the method will return True.

Syntax:

```
os.path.isdir(path)
```

Parameter:
path: A path-like object representing a file system path.

Return Type: This method returns a Boolean value of class bool. This method returns True if specified path is an existing directory, otherwise returns False.

Example:

```
# Python program to explain os.path.isdir() method
 # importing os.path module
import os.path
# Path
```

```
path = '/home/User/Documents/file.txt'
# Check whether the
# specified path is an
# existing directory or not
isdir = os.path.isdir(path)
print(isdir)
# Path
path = '/home/User/Documents/'
# Check whether the
# specified path is an
# existing directory or not
isdir = os.path.isdir(path)
print(isdir)
```

Output:

```
False
True
```

listdir()

This method in Python is used to get the list of all files and directories in the specified directory. If we don't specify any directory, then the list of files and directories in the current working directory will be returned.

Example:

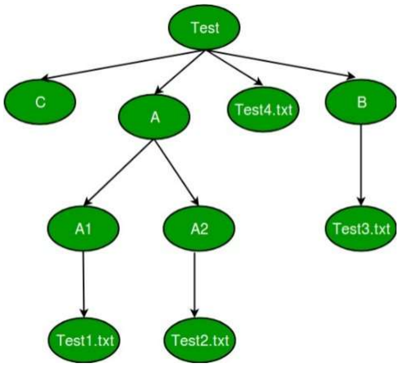
```
# Python program to explain os.listdir() method
# importing os module
import os
# Get the list of all files and directories
# in the root directory
path = "/"
dir_list = os.listdir(path)
print("Files and directories in '", path, "' :")
# print the list
print(dir_list)
```

Output:

```
Files and directories in ' / ' :
['sys', 'run', 'tmp', 'boot', 'mnt', 'dev', 'proc', 'var', 'bin', 'lib64', 'usr',
'lib', 'srv', 'home', 'etc', 'opt', 'sbin', 'media']
```

walk()

Suppose we have given below file structure in our system and we want to traverse all its branches completely from top to bottom?



How does os.walk() work in python ?

OS.walk() generate the file names in a directory tree by walking the tree either top-down or bottom-up. For each directory in the tree rooted at directory top (including top itself), it yields a 3-tuple (dirpath, dirnames, filenames).

root: Prints out directories only from what you specified.

dirs: Prints out sub-directories from root.

files: Prints out all files from root and directories.

Example1:

```
# Driver function
import os
if __name__ == "__main__":
   for (root,dirs,files) in os.walk('Test', topdown=True):
      print (root)
      print (dirs)
      print (files)
      print ('--------------------------------')
```

Example2:

```
import os
for root, dirs, files in os.walk('c:\python\osmodule'):
   for name in files:
      print('file: ' + os.path.join(root, name))
   for name in dirs:
      print('dir: ' + os.path.join(root, name))
```

output

```
file: C:\python\osmodule\main.py
file: C:\python\osmodule\renamed_file.py
dir: C:\python\osmodule\.idea
file: C:\python\osmodule\.idea\.gitignore
file: C:\python\osmodule\.idea\misc.xml
file: C:\python\osmodule\.idea\modules.xml
file: C:\python\osmodule\.idea\osmodule.iml
file: C:\python\osmodule\.idea\workspace.xml
```

```
dir: C:\python\osmodule\.idea\inspectionProfiles
file: C:\python\osmodule\.idea\inspectionProfiles\profiles_settings.xml
file: C:\python\osmodule\.idea\inspectionProfiles\Project_Default.xml
```

chdir()

The os module provides the chdir() function to change the current working directory.

Example:

```
import os
os.chdir("d:\\")
```

Output:

```
d:\\
```

Path:

The os.path module is a very extensively used module that is handy when processing files from different places in the system. It is used for different purposes such as for merging, normalizing and retrieving path names in python . All of these functions accept either only bytes or only string objects as their parameters. Its results are specific to the OS on which it is being run.

import os.path vs import os

os.path works strangely actually. It looks like os packaged with a submodule path but actually, os is a normal module which operate with sys.module to support os.path. Let us list what happens behind the scenes:

When Python starts, it loads many modules into sys.module.

os module is also loaded when Python starts. It assigns its path to the os specific module attribute.

It injects sys.modules['os.path'] = path so that you're able to do import os.path as though it was a submodule.

os.path module contains some useful functions on pathnames. The path parameters are either strings or bytes . The result is an object of the same type, if a path or file name is returned. As there are different versions of operating system so there are several versions of this module in the standard library. **Following are some functions of OS Path module.**

1. os.path.basename(path): This function gives us the last part of the path which may be a folder or a file name. The difference in how the path is mentioned in Windows and Linux in terms of the backslash and the forward slash.

Example1:

```
import os
# In windows
fldr = os.path.basename("C:\Users\xyz\Documents\My Web Sites")
print(fldr)
```

```
file = os.path.basename("C:\Users\xyz\Documents\My Web Sites\intro.html")
print(file)
# In nix*
fldr = os.path.basename("/Documents/MyWebSites")
print(fldr)
file = os.path.basename("/Documents/MyWebSites/music.txt")
print(file)
Running the above code gives us the following result −
```

Output

```
My Web Sites
intro.html
MyWebSites
```

2. os.path.dirname(path) : This function gives us the directory name where the folder or file is located.

Example

```
import os
# In windows
DIR = os.path.dirname("C:\Users\xyz\Documents\My Web Sites")
print(DIR)
# In nix*
DIR = os.path.dirname("/Documents/MyWebSites")
print(DIR)
Running the above code gives us the following result −
```

Output

```
C:\Users\xyz\Documents
/Documents
```

3. os.path.isabs(path) : It specifies whether the path is absolute or not. In Unix system absolute path means path begins with the slash('/') and in Windows that it begins with a (back)slash after chopping off a potential drive letter.

Example

```
# isabs function
import os
out = os.path.isabs("/baz/foo")
print(out)
```

Output:

```
True
```

4. os.path.isdir(path): This function specifies whether the path is existing directory or not.

Example

```
# isdir function
import os
out = os.path.isdir("C:\\Users")
print(out)
```

Output:

```
True
```

5. os.path.isfile(path) : Sometimes we may need to check if the complete path given, represents a folder or a file. If the file does not exist then it will give False as the output. If the file exists then the output is True.

Example

```
print(IS_FILE)
IS_FILE = os.path.isfile("C:\Users\xyz\Documents\My Web Sites\intro.html")
print(IS_FILE)
# In nix*
IS_FILE = os.path.isfile("/Documents/MyWebSites")
print(IS_FILE)
IS_FILE = os.path.isfile("/Documents/MyWebSites/music.txt")
print(IS_FILE)
```

Running the above code gives us the following result –

Output

```
False
True
False
True
```

6. os.path.normcase(path): This function normalizes the case of the pathname specified. In Unix and Mac OS X system it returns the pathname as it is . But in Windows it converts the path to lowercase and forward slashes to backslashes.

Example

```
# normcase function in windows
import os
out = os.path.normcase("/BAz")
print(out)
```

Output:

```
'\\baz'
```

7. os.path.normpath(path) : This is an interesting function which will normalize the given path by eliminating extra slashes or changing the backslash to forward slash depending on which OS it is. Which means so that A//B, A/B/, A/./B and A/foo/../B all become A/B. On Windows, it converts forward slashes to backward slashes.

Example

```
import os
# Windows path
NORM_PATH = os.path.normpath("C:/Users/Pradeep/Documents/My Web Sites")
print(NORM_PATH)
# Unix Path
NORM_PATH = os.path.normpath("/home/ubuuser//Documents/")
print(NORM_PATH)
```

Running the above code gives us the following result −

Output

```
# Running in Windows
C:\Users\Pradeep\Documents\My Web Sites
\home\ubuuser\Documents
# Running in Linux
C:/Users/Pradeep/Documents/My Web Sites
/home/ubuuser/Documents
```

Example2:

```
# normpath function in Unix
import os
out = os.path.normpath("foo/./bar")
print(out)
```

Output:

```
 'foo/bar'
```

8. os.path.getsize(): In this method, python will give us the size of the file in bytes. To use this method we need to pass the name of the file as a parameter.

Example

```
import os #importing os module
size = os.path.getsize("filename")
print("Size of the file is", size," bytes.")
```

Output:

```
Size of the file is 192 bytes.
```

Extra functions:

split(p)

Split a pathname. Return tuple (head, tail) where tail is everything after the final slash. Either part may be empty.

```
import os
split = os.path.split('path/to/file.html')
print(split)
```

output:

```
 ('path/to', 'file.html')
```

getcwd()

It returns the current working directory(CWD) of the file.

```
import os
print(os.getcwd())
```

rmdir()

The rmdir() function removes the specified directory with an absolute or related path. First, we have to change the current working directory and remove the folder.

Example

```
import os
# It will throw a Permission error; that's why we have to change the current working directory.
os.rmdir("d:\\newdir")
os.chdir("..")
os.rmdir("newdir")
```

rename()

A file or directory can be renamed by using the function os.rename(). A user can rename the file if it has privilege to change the file.

Example

```
import os
fd = "python.txt"
os.rename(fd,'Python1.txt')
os.rename(fd,'Python1.txt')
```

Output:

```
Traceback (most recent call last):
  File "main.py", line 3, in
    os.rename(fd,'Python1.txt')
FileNotFoundError: [Errno 2] No such file or directory: 'python.txt' -> 'Python1.txt'
```

error()

The os.error() function defines the OS level errors. It raises OSError in case of invalid or inaccessible file names and path etc.

Example

```
import os
try:
    # If file does not exist,
    # then it throw an IOError
    filename = 'Python.txt'
    f = open(filename, 'rU')
    text = f.read()
    f.close()
# The Control jumps directly to here if
# any lines throws IOError.
except IOError:
    # print(os.error) will <class 'OSError'>
    print('Problem reading: ' + filename)
```

Output:

```
Problem reading: Python.txt
```

All functions in the python os module raise the OSError (or subclasses thereof) when invalid or inaccessible file names and paths, or other arguments that have the correct type, but are not accepted by the operating system are encountered.

Unit – 5 Python Web Framework: Flask

Flask is one of the most popular web application frameworks written in Python. It is a micro-framework designed for an easy and quick start. Extending with tools and libraries adds more functionality to Flask for more complex projects. For installation of Flask one must have already installed Python 2.7 or 3.5 and newer.

5.1 Installation of Flask and Environment Setup

Step 1: Install Virtual Environment

Install Flask in a virtual environment to avoid problems with conflicting libraries. Check Python version before starting:

- Python 3 comes with a virtual environment module called *venv* preinstalled. **If you have Python 3 installed, skip to Step 2.**
- Python 2 users must have to install the *virtualenv* module. **If you have Python 2, follow the instructions given below for windows.**

Install virtualenv on Windows

- 1. Open the command line with administrator privileges.
- 2. Use **pip** to install *virtualenv* on Windows:

```
py -2 -m pip install virtualenv
```

hint: https://phoenixnap.com/kb/install-pip-windows

Step 2: Create an Environment

1. Make a separate directory for your project:

```
mkdir <project name>
```

2. Move into the directory:

```
cd <project name>
```

3. Within the directory, create the virtual environment for Flask. When you create the environment, a new folder appears in your project directory with the environment's name.

Create an Environment in Windows

- **For Python 3:**

Create and name a virtual environment in Python 3 with:

```
py -m venv <name of environment>
```

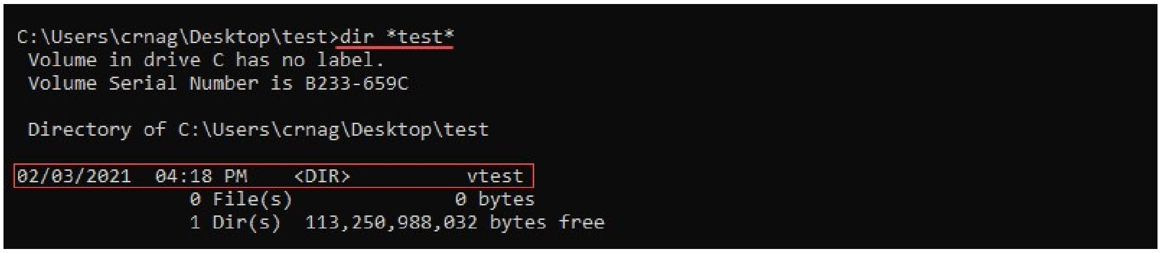
- **For Python 2:**

For Python 2, create the virtual environment with the *virtualenv* module:

```
py -m virtualenv <name of environment>
```

List the folder structure using the **dir** command:

dir *<project name>*



```
C:\Users\crnag\Desktop\test>dir *test*
 Volume in drive C has no label.
 Volume Serial Number is B233-659C

 Directory of C:\Users\crnag\Desktop\test

02/03/2021  04:18 PM    <DIR>          vtest
               0 File(s)              0 bytes
               1 Dir(s)  113,250,988,032 bytes free
```

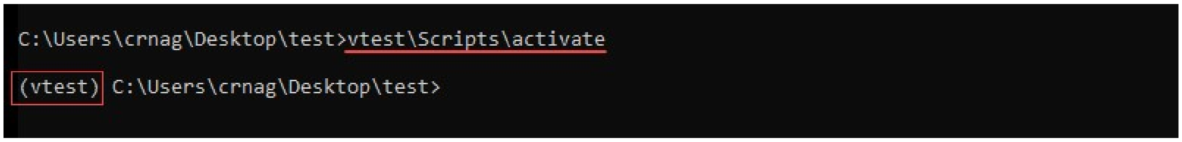
The project directory shows the newly created environment:

Step 3: Activate the Environment

Activate the virtual environment before installing Flask. The name of the activated environment shows up in the CLI after activation.

Activate the Environment on Windows

For Windows, activate the virtual environment with:



```
C:\Users\crnag\Desktop\test>vtest\Scripts\activate

(vtest) C:\Users\crnag\Desktop\test>
```

<name of environment>\Scripts\activate

Step 4: Install Flask

Install Flask within the activated environment using **pip**:

pip install Flask

Flask is installed automatically with all the dependencies.

Step 5: Test the Development Environment

1. Create a simple Flask application to test the newly created development environment.

2. Make a file in the Flask project folder called *hello.py*.

3. Edit the file using a text editor and add the following code to make an application that prints "*Hello world!*":

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def helloworld():
   return 'Hello world!'
```

4. Save the file and close.

5. Using the console, navigate to the project folder using the **cd** command.

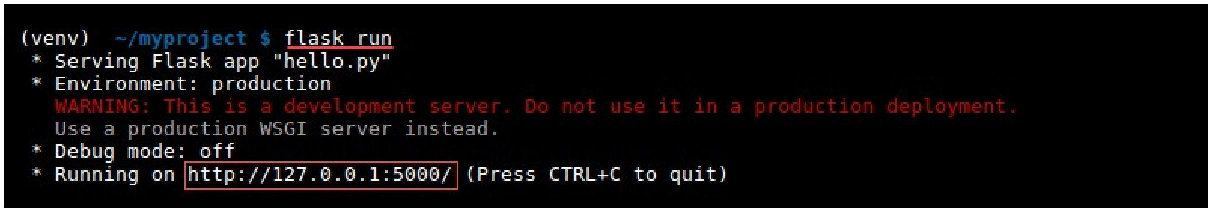
6. Set the *FLASK_APP* environment variable.

- **For Windows:**

setx FLASK_APP "hello.py"

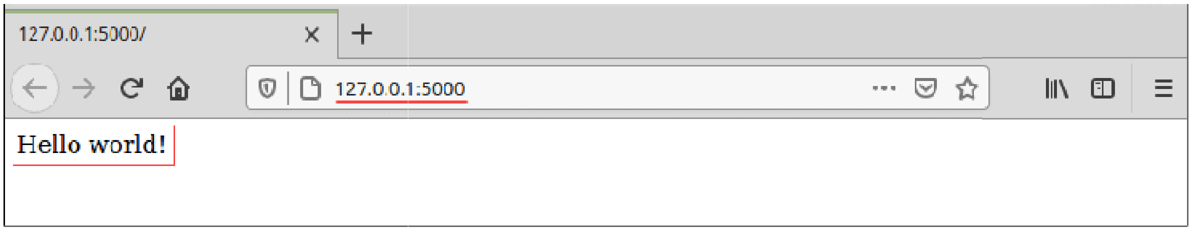
7. Run the Flask application with:

flask run



```
(venv)  ~/myproject $ flask run
 * Serving Flask app "hello.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

The output prints out a confirmation message and the address.



Hello world!

8. Copy and paste the address into the browser to see the project running:

To View Video: https://www.youtube.com/watch?v=QjtW-wnXlUY

5.1.1 Web Server Gateway Interface

WSGI refers to Web Server Gateway Interface. WSGI plays a vital role at the time when you deploy your Django or Flask application.

WSGI is a specification that describes the communication between **web servers and Python web applications or frameworks**. It explains how a web server communicates with python web applications/frameworks and how web applications/frameworks can be chained for processing a request.

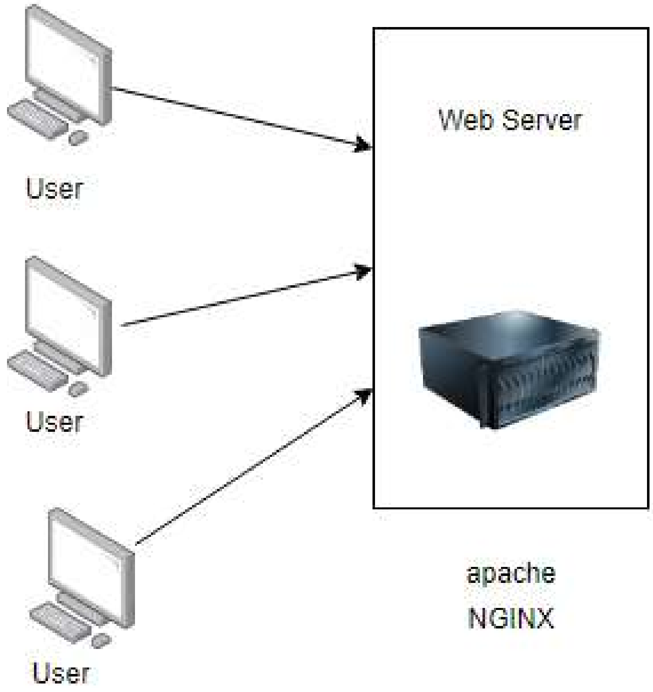
How does WSGI work?

Now, let's have a look at how WSGI work. So, to obtain a clear understanding of WSGI, let us assume a case scenario where you have a web application developed in Django or Flask application as shown in the figure

Python application
Django
Flask

The above figure represents your Django/Flask application.

Since a web application is deployed in the web server. The figure below represents the web server that obtains requests from various users.

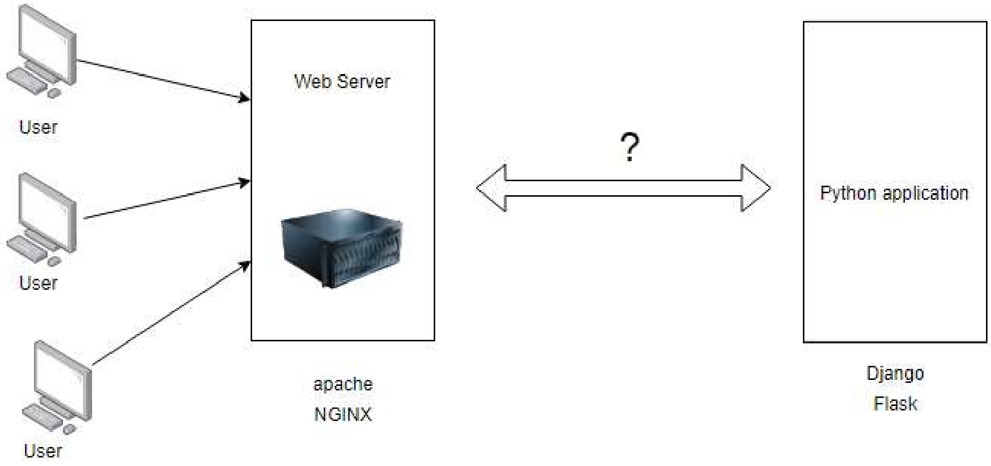


The web server which obtains requests from the users.

The above web server can be apache, NGINX, etc. server which is responsible for handling various static files and caching purposes. Furthermore, you can also use the server as a load balancer if you are willing to scale multiple applications.

How can a web server interact with the Python application?

The figure representing problem in the interaction between Python application and a web server.



So, now a problem arises as a web server has to interact with a Python application.

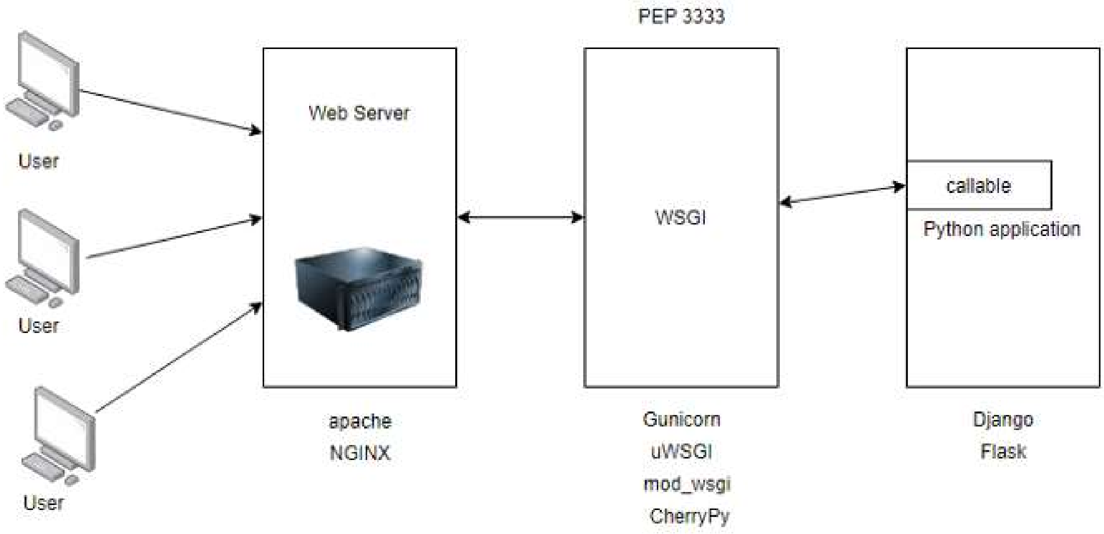
Hence, a mediator is required for carrying out the interaction between the web servers and the Python application. So, the standard for carrying out communication between the web server and Python application is WSGI(Web Server Gateway Interface).

Now, web server is able to send requests or communicate with WSGI containers. Likewise, Python application provides a 'callable' object which contains certain functionalities that are invoked by WSGI application which are defined as per the PEP 3333 standard. Hence, there are multiple WSGI containers available such as Gunicorn, uWSGI, etc.

The figure below represents the communication that is carried out between web server, WSGI, and Python application.

Communication between user, web server, WSGI, and Python application.

There are multiple WSGI containers that are available today. Hence, a WSGI container is required to be installed in the project so that a web server can communicate to a WSGI container which further communicates to the Python application and provides the response back accordingly. Finally, when the web server obtains the response, it is sent back to the web browser/users.



Why use the WSGI rather than directly pointing the web server to the Django or Flask application?

If you directly point your web server to your application, it reduces the **flexibility** of your application. Since your web server now directly points to your web application, you are unable to swap out web stack components. Now, let's have a look at an example to make you clear about the applicability of WSGI. For instance, today you have decided to deploy your application using Gunicorn but after some years you decide to switch from Gunicorn to mod_wsgi. Now, in this case, you can easily switch to mod_wsgi without making any changes in the application or framework that implements WSGI. Hence, WSGI provides flexibility to your application.

Another reason for using WSGI is due to its **scalability**. Once your application is live, up and running there can be thousands of requests in your application. Hence, WSGI is capable of serving thousands of requests at a time. As we know, the WSGI server is responsible for handling the requests from the web server and takes decision for carrying out the communication of those requests to an application framework's process. Here, we can divide the responsibilities among the servers for scaling web traffic.

5.1.2 Web template engine (Jinja2)

Jinja2 is a modern day templating language for Python developers. It was made after Django's template. It is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.

It includes:

- Template inheritance and inclusion.
- Define and import macros within templates.
- HTML templates can use autoescaping to prevent XSS from untrusted user input.
- A sandboxed environment can safely render untrusted templates.
- Async support for generating templates that automatically handle sync and async functions without extra syntax.
- I18N support with Babel.
- Templates are compiled to optimized Python code just-in-time and cached, or can be compiled ahead-of-time.
- Exceptions point to the correct line in templates to make debugging easier.
- Extensible filters, tests, functions, and even syntax.

Jinja's philosophy is that while application logic belongs in Python if possible, it shouldn't make the template designer's job difficult by restricting functionality too much.

5.1.3 Creating the Flask Class object

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

class **flask.Flask(*import_name*, *static_path=None*, *static_url_path=None*, *static_folder='static'*, *template_folder='templates'*, *instance_path=None*, *instance_relative_config=False*)**

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an *__init__.py* file inside) or a standard module (just a *.py* file).

Usually you create a **Flask** instance in your main module or in the *__init__.py* file of your package like this:

```
from flask import Flask
app = Flask(__name__)
```

The idea of the first parameter is to give Flask an idea what belongs to your application. This name is used to find resources on the file system, can be used by extensions to improve debugging information and a lot more.

So it's important what you provide there. If you are using a single module, *__name__* is always the correct value. If you however are using a package, it's usually recommended to hardcode the name of your package there.

For example if your application is defined in *yourapplication/app.py* you should create it with one of the two versions below:

```
app = Flask('yourapplication')
app = Flask(__name__.split('.')[0])
```

Why is that? The application will work even with *__name__*, thanks to how resources are looked up. However it will make debugging more painful. Certain extensions can make assumptions based on the import name of your application. For example the Flask-SQLAlchemy extension will look for the code in your application that triggered an SQL query in debug mode. If the import name is not properly set up, that debugging information is lost. (For example it would only pick up SQL queries in *yourapplication.app* and not *yourapplication.views.frontend*)

New in version 0.7: The *static_url_path*, *static_folder*, and *template_folder* parameters were added.

New in version 0.8: The *instance_path* and *instance_relative_config* parameters were added.

Parameters:	- **import_name** – the name of the application package - **static_url_path** – can be used to specify a different path for the static files on the web. Defaults to the name of the *static_folder* folder. - **static_folder** – the folder with static files that should be served at *static_url_path*. Defaults to the 'static' folder in the root path of the application. - **template_folder** – the folder that contains the templates that should be used by the application. Defaults

	to 'templates' folder in the root path of the application. **instance_path** – An alternative instance path for the application. By default the folder 'instance' next to the package or module is assumed to be the instance path. **instance_relative_config** – if set to *True* relative filenames for loading the config are assumed to be relative to the instance path instead of the application root.

5.1.4 Creating and hosting first basic Flask App.

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for beginners to programming, since you can build a web application quickly using only a single Python file.

5.1.4.1route(), run(), add_url()_rule()

App routing is used to map the specific URL with the associated function that is intended to perform some task.

In our first application, the URL ('/') is associated with the home function that returns a particular string displayed on the web page.

In other words, we can say that if we visit the particular URL mapped to some particular function, the output of that function is rendered on the browser's screen.

Example

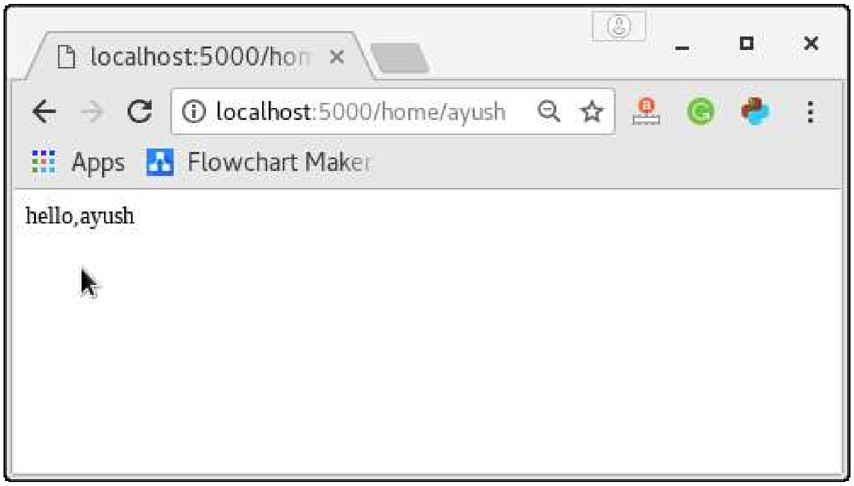
```
from flask import Flask
app = Flask(__name__)

@app.route('/home')
def home():
  return "hello, welcome to our website";

if __name__ =="__main__":
  app.run(debug = True)
```

Flask facilitates us to add the variable part to the URL by using the section. We can reuse the variable by adding that as a parameter into the view function. Consider the following example.

Example

```
from flask import Flask
app = Flask(__name__)
@app.route('/home/<name>')
def home(name):
  return "hello,"+name;
if __name__ =="__main__":
  app.run(debug = True)
```

It will produce the following result on the web browser.

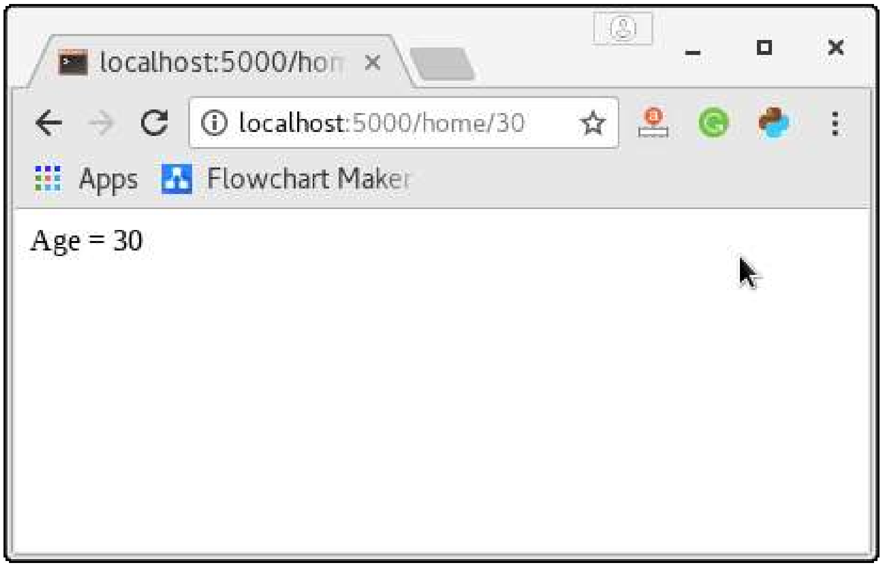


hello,ayush

The converter can also be used in the URL to map the specified variable to the particular data type. For example, we can provide the integers or float like age or salary respectively.

Example

```
from flask import Flask
app = Flask(__name__)
@app.route('/home/<int:age>')
def home(age):
  return "Age = %d"%age;
if __name__ =="__main__":
  app.run(debug = True)
```



Age = 30

The following converters are used to convert the default string type to the associated data type.

string: default

int: used to convert the string to the integer

float: used to convert the string to the float.

path: It can accept the slashes given in the URL.

add_url_rule() function

There is one more approach to perform routing for the flask web application that can be done by using the add_url() function of the Flask class. The syntax to use this function is given below.

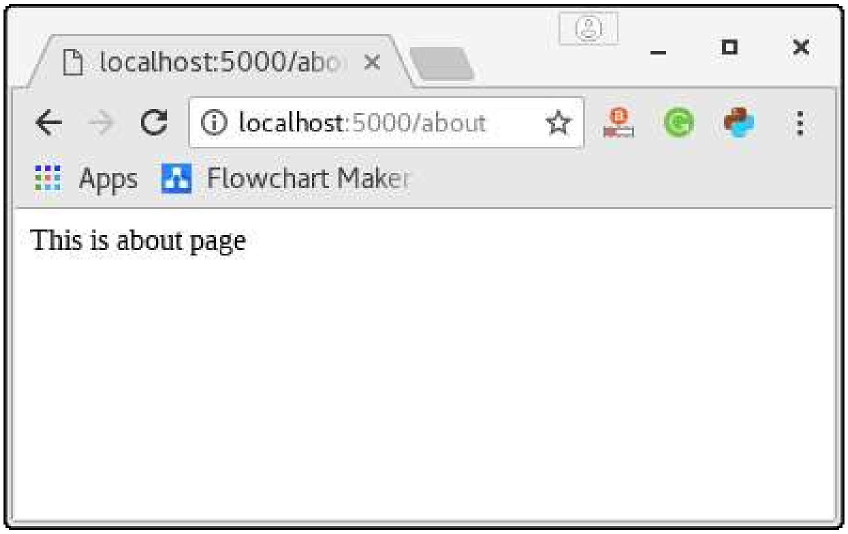
```
add_url_rule(<url rule>, <endpoint>, <view function>)
```

This function is mainly used in the case if the view function is not given and we need to connect a view function to an endpoint externally by using this function.

Consider the following example.

Example

```
from flask import Flask
app = Flask(__name__)
def about():
  return "This is about page";
app.add_url_rule("/about","about",about)
if __name__ =="__main__":
  app.run(debug = True)
```



This is about page

5.2 URL building and its advantages

5.2.1 url_for() function

The url_for() function is used to build a URL to the specific function dynamically. The first argument is the name of the specified function, and then we can pass any number of keyword argument corresponding to the variable part of the URL.

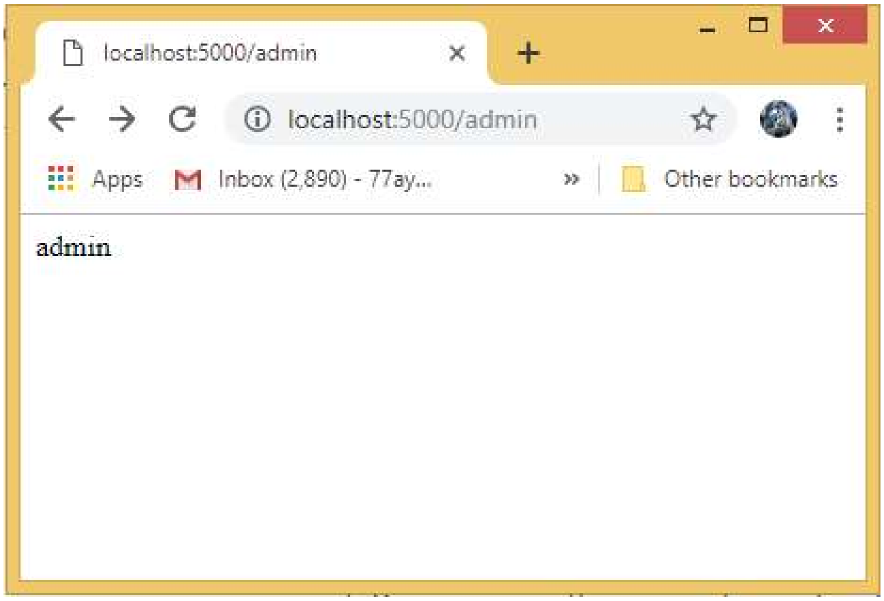
This function is useful in the sense that we can avoid hard-coding the URLs into the templates by dynamically building them using this function.

Consider the following python flask script.

Example

```
from flask import *

app = Flask(__name__)
 @app.route('/admin')
def admin():
  return 'admin'

@app.route('/librarion')
def librarion():
  return 'librarion'
@app.route('/student')
def student():
  return 'student'

@app.route('/user/<name>')
def user(name):
  if name == 'admin':
    return redirect(url_for('admin'))
  if name == 'librarion':
    return redirect(url_for('librarion'))
  if name == 'student':
    return redirect(url_for('student'))
if __name__ =='__main__':
  app.run(debug = True)
```

The above script simulates the library management system which can be used by the three types of users, i.e., admin, librarian, and student. There is a specific function named user() which recognizes the user the redirect the user to the exact function which contains the implementation for this particular function.



admin

For example, the URL http://localhost:5000/user/admin is redirected to the URL http://localhost:5000/admin, the URL localhost:5000/user/librarion, is redirected to the URL http://localhost:5000/librarion, the URL http://localhost:5000/user/student is redirected to the URL http://localhost/student.

Benefits of the Dynamic URL Building

- It avoids hard coding of the URLs.
- We can change the URLs dynamically instead of remembering the manually changed hard-coded URLs.
- URL building handles the escaping of special characters and Unicode data transparently.
- The generated paths are always absolute, avoiding unexpected behavior of relative paths in browsers.
- If your application is placed outside the URL root, for example, in /myapplication instead of /, url_for() properly handles that for you.

5.3 Flask HTTP methods:

HTTP is the hypertext transfer protocol which is considered as the foundation of the data transfer in the world wide web. All web frameworks including flask need to provide several HTTP methods for data communication.

The methods are given in the following table.

SN	Method	Description
1	GET	It is the most common method which can be used to send data in the unencrypted form to the server.
2	HEAD	It is similar to the GET but used without the response body.
3	POST	It is used to send the form data to the server. The server does not cache the data transmitted using the post method.
4	PUT	It is used to replace all the current representation of the target resource with the uploaded content.
5	DELETE	It is used to delete all the current representation of the target resource specified in the URL.

We can specify which HTTP method to be used to handle the requests in the route() function of the Flask class. By default, the requests are handled by the GET() method.

POST Method

To handle the POST requests at the server, let us first create a form to get some data at the client side from the user, and we will try to access this data on the server by using the POST request.

login.html

```
<html>
  <body>
    <form action = "http://localhost:5000/login" method = "post">
```

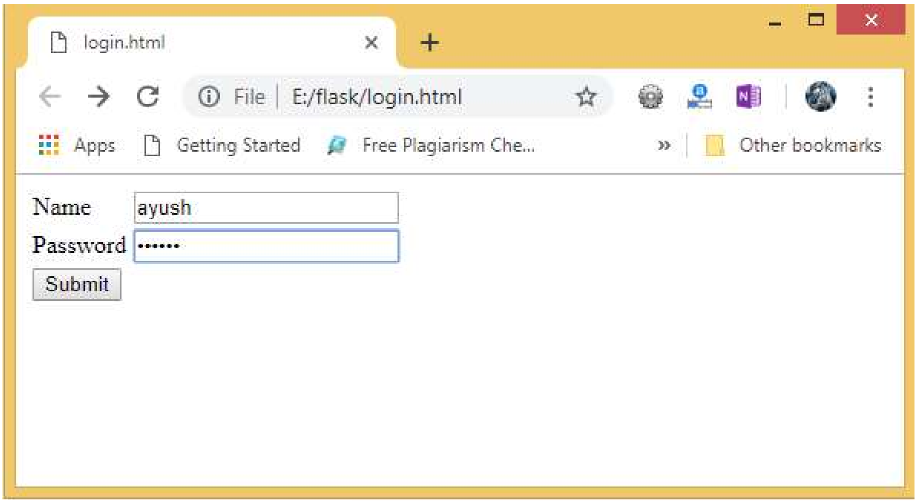
```
     <table>
    <tr><td>Name</td>
    <td><input type ="text" name ="uname"></td></tr>
    <tr><td>Password</td>
    <td><input type ="password" name ="pass"></td></tr>
    <tr><td><input type = "submit"></td></tr>
  </table>
   </form>
  </body>
</html>
```

Now, Enter the following code into the script named post_example.py.

post_example.py

```
from flask import *
app = Flask(__name__)

@app.route('/login',methods = ['POST'])
def login():
    uname=request.form['uname']
    passwrd=request.form['pass']
    if uname=="ayush" and passwrd=="google":
       return "Welcome %s" %uname
if __name__ == '__main__':
  app.run(debug = True)
```

Now, start the development server by running the script using python post_exmple.py and open login.html on the web browser as shown in the following image.



Name ayush
Password ••••••

Give the required input and click Submit, we will get the following result.

Hence, the form data is sent to the development server by using the post method.

GET Method

Let's consider the same example for the Get method. However, there are some changes in the data retrieval syntax on the server side. First, create a form as login.html.

login.html

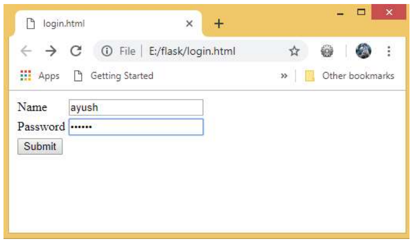
```
<html>
  <body>
    <form action = "http://localhost:5000/login" method = "get">
      <table>
     <tr><td>Name</td>
     <td><input type ="text" name ="uname"></td></tr>
     <tr><td>Password</td>
     <td><input type ="password" name ="pass"></td></tr>
     <tr><td><input type = "submit"></td></tr>
   </table>
    </form>
  </body>
</html>
```

Now, create the following python script as get_example.py.

get_example.py

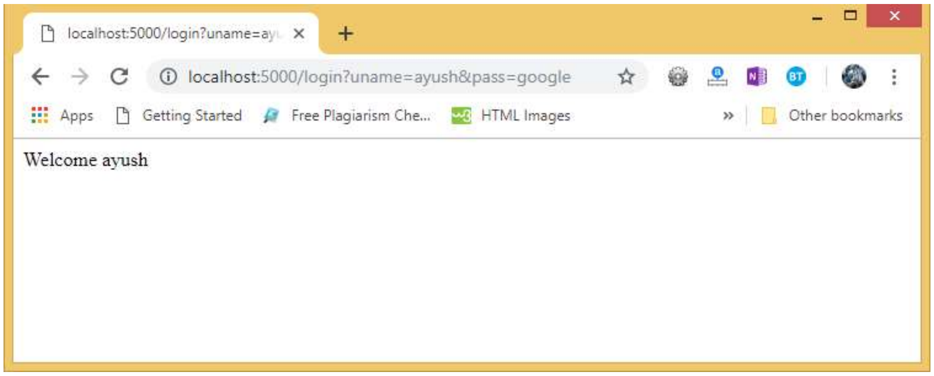
```
from flask import *
app = Flask(__name__)

@app.route('/login',methods = ['GET'])
def login():
     uname=request.args.get('uname')
     passwrd=request.args.get('pass')
     if uname=="ayush" and passwrd=="google":
        return "Welcome %s" %uname

if __name__ == '__main__':
  app.run(debug = True)
```

Now, open the HTML file, login.html on the web browser and give the required input.



Name ayush
Password ••••••
Submit

Now, click the submit button.


localhost:5000/login?uname=ayu
localhost:5000/login?uname=ayush&pass=google
Apps
Getting Started
Free Plagiarism Che...
HTML Images
Other bookmarks

Welcome ayush

As we can check the result. The data sent using the get() method is retrieved on the development server.

The data is obtained by using the following line of code.

```
uname = request.args.get('uname')
```

Here, the args is a dictionary object which contains the list of pairs of form parameter and its corresponding value.

In the above image, we can also check the URL which also contains the data sent with the request to the server. This is an important difference between the GET requests and the POST requests as the data sent to the server is not shown in the URL on the browser in the POST requests.

HEAD method

HEAD is a request method supported by HTTP used by the World Wide Web. The HEAD method asks for a response identical to that of a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

How to make HEAD request through Python Requests

Python’s requests module provides in-built method called **head()** for making a HEAD request to a specified URI.

Syntax

```
requests.head(url, params={key: value}, args)
```

Example

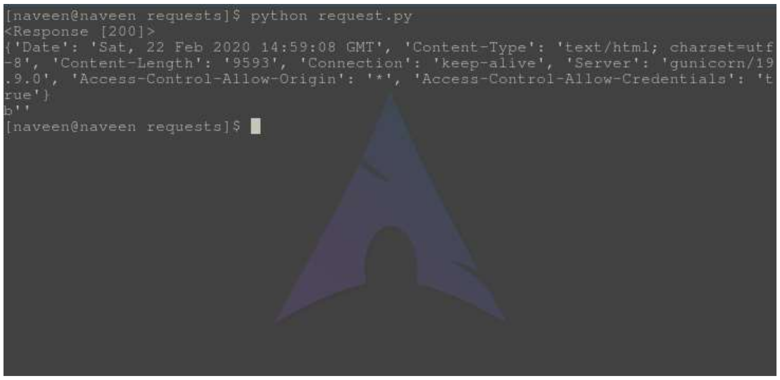
```
import requests
# Making a HEAD request
r = requests.head('https://httpbin.org/', data ={'key':'value'})
# check status code for response received
# success code - 200
print(r)
```

```
# print headers of request
print(r.headers)
# checking if request contains any content
print(r.content)
```

save this file as request.py and through terminal run,

python request.py

Output



```
[naveen@naveen requests]$ python request.py
<Response [200]>
{'Date': 'Sat, 22 Feb 2020 14:59:08 GMT', 'Content-Type': 'text/html; charset=utf
-8', 'Content-Length': '9593', 'Connection': 'keep-alive', 'Server': 'gunicorn/19
.9.0', 'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Credentials': 't
rue'}
b''
[naveen@naveen requests]$ 
```

DELETE Method

DELETE is a request method supported by HTTP used by the World Wide Web. The DELETE method deletes the specified resource. As with a PUT request, you need to specify a particular resource for this operation. A successful response SHOULD be 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has not yet been enacted, or 204 (No Content) if the action has been enacted but the response does not include an entity.

How to make DELETE request through Python Requests

Python's requests module provides in-built method called **delete()** for making a DELETE request to a specified URI.

Syntax

```
requests.delete(url, params={key: value}, args)
```

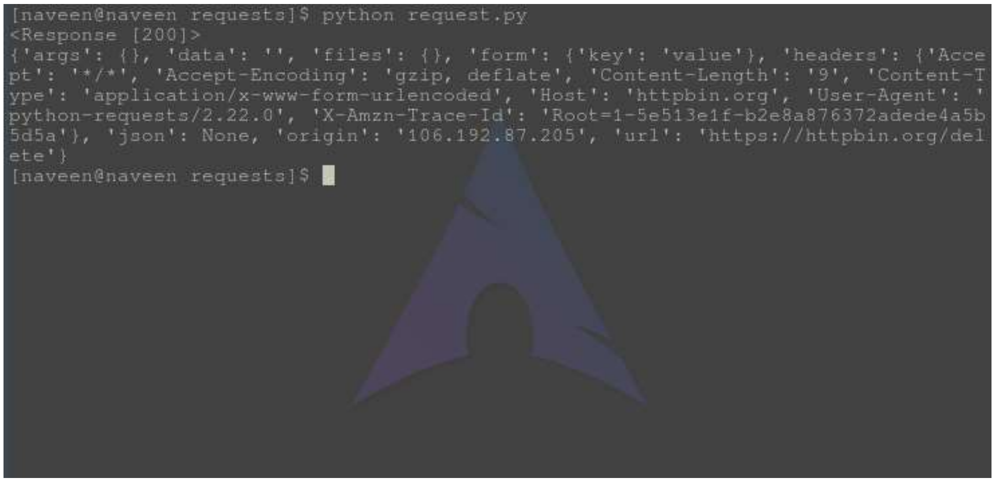
Example

```
import requests
# Making a DELETE request
r = requests.delete('https://httpbin.org / delete', data ={'key':'value'})
# check status code for response received
# success code - 200
print(r)
# print content of request
print(r.json())
```

save this file as request.py and through terminal run,

python request.py

Output



```
[naveen@naveen requests]$ python request.py
<Response [200]>
{'args': {}, 'data': '', 'files': {}, 'form': {'key': 'value'}, 'headers': {'Acce
pt': '*/*', 'Accept-Encoding': 'gzip, deflate', 'Content-Length': '9', 'Content-T
ype': 'application/x-www-form-urlencoded', 'Host': 'httpbin.org', 'User-Agent': '
python-requests/2.22.0', 'X-Amzn-Trace-Id': 'Root=1-5e513e1f-b2e8a876372adede4a5b
5d5a'}, 'json': None, 'origin': '106.192.87.205', 'url': 'https://httpbin.org/del
ete'}
[naveen@naveen requests]$
```

PUT Method

PUT is a request method supported by HTTP used by the World Wide Web. The PUT method requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified and if the URI does not point to an existing resource, then the server can create the resource with that URI.

How to make PUT request through Python Requests

Python's requests module provides in-built method called **put()** for making a PUT request to a specified URI.

Syntax

```
requests.put(url, params={key: value}, args)
```

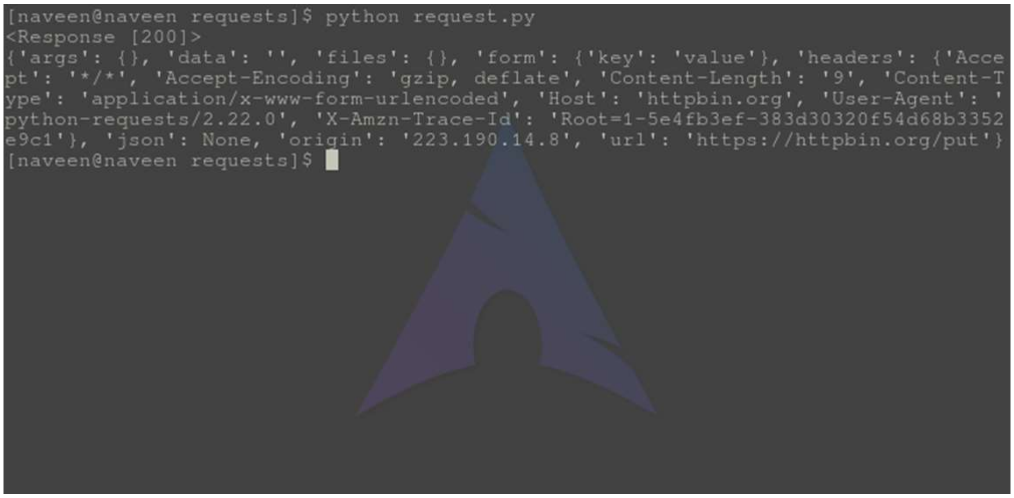
Example

```
import requests
# Making a PUT request
r = requests.put('https://httpbin.org / put', data ={'key':'value'})
# check status code for response received
# success code - 200
print(r)
# print content of request
print(r.content)
```

save this file as request.py and through terminal run,

python request.py

Output –



```
[naveen@naveen requests]$ python request.py
<Response [200]>
{'args': {}, 'data': '', 'files': {}, 'form': {'key': 'value'}, 'headers': {'Acce
pt': '*/*', 'Accept-Encoding': 'gzip, deflate', 'Content-Length': '9', 'Content-T
ype': 'application/x-www-form-urlencoded', 'Host': 'httpbin.org', 'User-Agent': '
python-requests/2.22.0', 'X-Amzn-Trace-Id': 'Root=1-5e4fb3ef-383d30320f54d68b3352
e9c1'}, 'json': None, 'origin': '223.190.14.8', 'url': 'https://httpbin.org/put'}
[naveen@naveen requests]$ 
```

Difference between PUT and POST methods

PUT	POST
PUT request is made to a particular resource. If the Request-URI refers to an already existing resource, an update operation will happen, otherwise create operation should happen if Request-URI is a valid resource URI (assuming client is allowed to determine resource identifier). **Example** PUT /article/{article-id}	POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. It essentially means that POST request-URI should be of a collection URI. **Example** POST /articles
PUT method is idempotent. So if you send retry a request multiple times, that should be equivalent to single request modification.	POST is NOT idempotent. So if you retry the request N times, you will end up having N resources with N different URIs created on server.
Use PUT when you want to modify a single resource which is already a part of resources collection. PUT overwrites the resource in its entirety. Use PATCH if request updates part of the resource.	Use POST when you want to add a child resource under resources collection.
Generally, in practice, always use PUT for UPDATE operations.	Always use POST for CREATE operations.

5.3.2 Dynamic Data Representation using Jinja2

Web 2.0 has introduced the concept of delivering user-centric interactive content. In the very early stage of the internet, the content delivered had static files. Fast forward to the current situation: Everything you consume is dynamically generated. The website content changes according to the user inputs and the logic for implementing such changes involves additional scripting via the server or client-side.

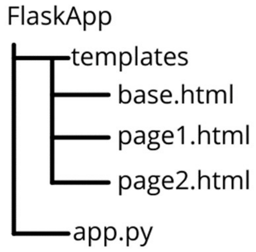
Web development using Python is one of the most preferred projects after Data Science and Machine Learning. Every Python learner is advised to adopt at least one web development framework using Python. Some of the most popular choices include Flask, Django, and FastAPI. One thing that is common among all these frameworks is the use of templating engines.

Templating engines allow us to pass the data from the server-side to HTML pages (also support any text file and XML, CSV, LaTeX, etc) whenever the user makes the request. This also allows us to build interactive websites with less effort plus the support for the Object-Oriented paradigm on the abstract level (inheritance) for web pages nails it.

Jinja is one such template engine that is used extensively in Flask and FastAPI. Django's Templating engine is also very much similar to Jinja with a few differences.Jinja templating comes pre-installed with Flask and as an optional requirement in FastAPI. The usage of the templating engine in both frameworks remains the same.

We are able to use templates in flask for storing the non-changing data, but we can also use them dynamically to show data using Jinja. Jinja is used to write python-like syntax in HTML files, which helps in using variables like functionality. In other words, we can make dynamic templates also.

File structure



5.3.2.1Jinja2 Delimiters

Jinga 2 template engine provides some delimiters which can be used in the HTML to make it capable of dynamic data representation. The template system provides some HTML syntax which are placeholders for variables and expressions that are replaced by their actual values when the template is rendered.

The jinga2 template engine provides the following delimiters to escape from the HTML.

- {% ... %} for statements
- {{ ... }} for expressions to print to the template output
- {# ... #} for the comments that are not included in the template output
- # ... ## for line statements

Example

Consider the following example where the variable part of the URL is shown in the HTML script using the {{ ... }} delimiter.

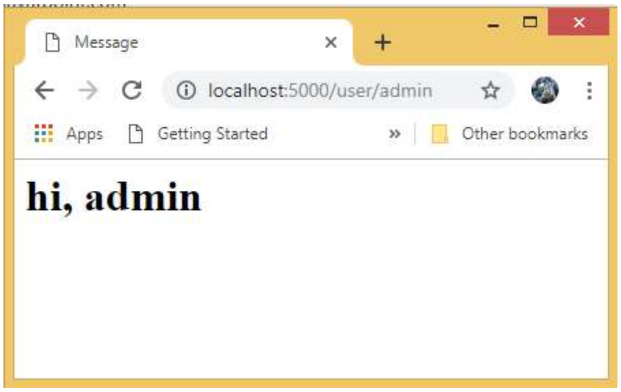
message.html

```
<html>
<head>
<title>Message</title>
</head>
<body>
<h1>hi, {{ name }}</h1>
</body>
</html>
```

script.py

```
from flask import *
app = Flask(__name__)

@app.route('/user/<uname>')
def message(uname):
    return render_template('message.html',name=uname)
if __name__ == '__main__':
  app.run(debug = True)
```

The variable part of the URL http://localhost:5000/user/admin is shown in the HTML script using the {{name}} placeholder.



hi, admin

Embedding Python statements in HTML

Due to the fact that HTML is a mark-up language and purely used for the designing purpose, sometimes, in the web applications, we may need to execute the statements for the general-purpose computations. For this purpose, Flask facilitates us the delimiter {%...%} which can be used to embed the simple python statements into the HTML.

Example

In the following example, we will print the table of a number specified in the URL, i.e., the URL http://localhost:5000/table/10 will print the table of 10 on the browser's window.

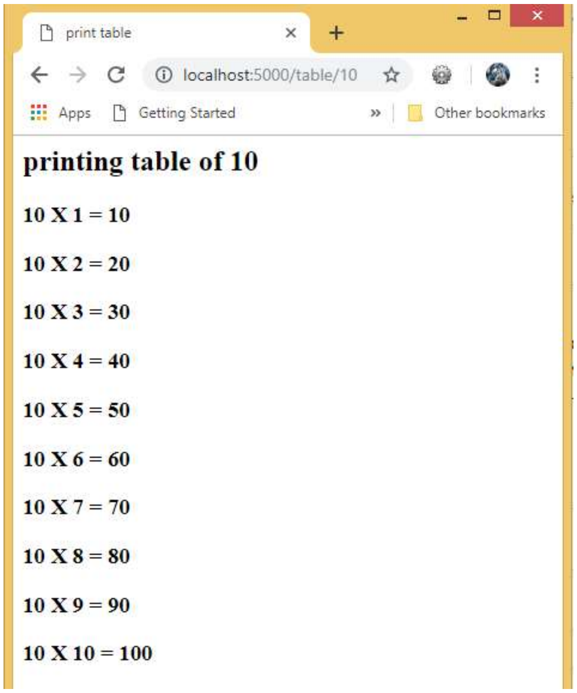
Here, we must notice that the for-loop statement is enclosed inside {%...%} delimiter, whereas, the loop variable and the number is enclosed inside {{ ... }} placeholders.

script.py

```
from flask import *
app = Flask(__name__)

@app.route('/table/<int:num>')
def table(num):
    return render_template('print-table.html',n=num)
if __name__ == '__main__':
  app.run(debug = True)
```

print-table.py

```
<html>
<head>
<title>print table</title>
</head>
<body>
<h2> printing table of {{n}}</h2>
{% for i  in range(1,11): %}
   <h3>{{n}} X {{i}} = {{n * i}} </h3>
{% endfor %}
</body>
</html>
```



printing table of 10

```
10 X 1 = 10
10 X 2 = 20
10 X 3 = 30
10 X 4 = 40
10 X 5 = 50
10 X 6 = 60
10 X 7 = 70
10 X 8 = 80
10 X 9 = 90
10 X 10 = 100
```

Referring Static files in HTML

The static files such as CSS or JavaScript file enhance the display of an HTML web page. A web server is configured to serve such files from the static folder in the package or the next to the module. The static files are available at the path **/static** of the application.

Example

In the following example, the flask script returns the HTML file, i.e., **message.html** which is styled using the stylesheet **style.css**. The flask template system finds the static CSS file under **static/css** directory. Hence the style.css is saved at the specified path.

script.py

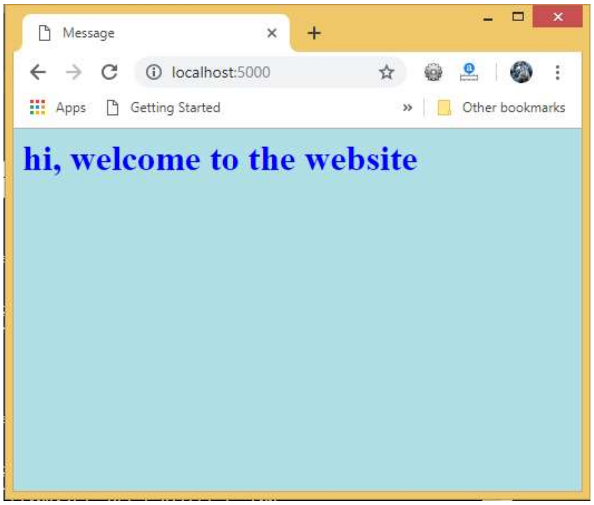
```
from flask import *
app = Flask(__name__)
@app.route('/')
def message():
    return render_template('message.html')
if __name__ == '__main__':
  app.run(debug = True)
```

message.html

```
<html>
<head>
  <title>Message</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
  <h1>hi, welcome to the website</h1>
</body>
</html>
```

style.css

```
body {
 background-color: powderblue;
}
h1 {
 color: blue;
}
p {
 color: red;
}
```



hi, welcome to the website

allinone.py

```
from flask import Flask, render_template_string, request

class CustomFlask(Flask):
   jinja_options = Flask.jinja_options.copy()
   jinja_options.update(dict(
      block_start_string='<%',
      block_end_string='%>',
      variable_start_string='%%',
      variable_end_string='%%',
      comment_start_string='<#',
      comment_end_string='#>',
   ))

app = CustomFlask(__name__)
app.config['DEBUG'] = True

@app.route("/")
def index():
   template = """
<# this is a jinja2 comment #>
<% block stuff %>
<h1>Jinja2</h1>
<% for i in range(5) %>
<p>Hello %% name %%!</p>
<% endfor %>

<h1>Mustache</h1>
<p>{{something}}</p>
      {{#items}}
       {{#first}}
<li><strong>{{name}}</strong></li>
```

```
        {{/first}}
        {{#link}}
<li><a href="{{url}}">{{name}}</a></li>
        {{/link}}
      {{/items}}
<% endblock %>
   """
   return render_template_string(template, name=request.values.get('name', Viral'))
if __name__ == "__main__":
   app.run(use_debugger=True, use_reloader=True)
```

5.3.2.2Embedding Python statement in HTML

As we have seen in the file structure in previous section, for working with the HTML files we need to consider the structure we have seen earlier. So we will keep all the html files within the same directory say "templates". Our data will be flew from the python to these html files.

Once we are able to start and run flask server, we can start sending inputs from the Python backend, and later on, we will see how inputs from web UI flow to the backend processing and render that result on the website.

The main idea of using a templating engine in web development is to render data dynamically in HTML files. To represent such data forms in HTML files, there are 3 types of syntaxes are available:

1. **Expressions:** These are the simple variable representation that is used for displaying any data type from the Python backend. It is similar to print statements in programming languages. The syntax is {{ ... }}
2. **Statements**: These syntaxes are used to implement control structures such as for loop, if-else, macros, blocks, imports, and much more things. The syntax is {% ... %}
3. **Comments**: One can also add Jinja comments to the template codes. These are represented as {# ... #} .

Let's try passing some values from the Python server to the Jinja templates. Simply create a new HTML file under the "templates" directory. In this HTML file, expression syntax has been used to print the value from the server to the HTML webpage. Here are the contents of the new.html file:

```
{{first_name}}, {{last_name}} :: {{article_title}}
```

The above content expects 3 parameters to be passed from the backend. To render this HTML template with actual values from the Python backend, you need to use the render_template method from the flask package. This method takes in the template name, followed by the values to be passed to the HTML template. The parameter's name should be the same as the name used in the HTML template. The following snippet shows this:

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/', methods=['GET', 'POST'])
def index():
   return render_template('new.html',
                     first_name=Viral',
                     last_name="Polishwala",
                     article_title="Jinja HTML ")
if __name__ == "__main__":
   app.run(debug=True)
```

5.3.2.3Static File reference in HTML

Web applications often require static files, such as javascript files or CSS files that support Web display. Typically, we configure the Web server and it provides us with this. But during development Flask development, Python parses all web requests. To solve this, these files are placed in the static folder, which will be available in the applications /static kind of folder.

Where to place static files

The URL of the special endpoint static is used to generate a static file. In your programs directory, create a new directory named static. In this directory you can place images, javascript files, css files and many other files that don’t need a Python backend.

About url_for() function

url_for() function in Flask allows us to create a absolute URL of resources. It takes two parameters:

1: Endpoint
2: Values

For example:

url_for('static', filename = 'name_of_file')

This will generate the path by using the file name present in the **static** folder.

Example

In the following example, the javascript function defined in the hello.js is called on the OnClick event of the HTML button in index.html, which is rendered on the “/“ URL of the Flask application.

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route("/")
def index():
  return render_template("index.html")
if __name__ == '__main__':
  app.run(debug = True)
```

Then index.html

```
<html>
  <head>
    <script type = "text/javascript"
      src = "{{ url_for('static', filename = 'hello.js') }}" ></script>
  </head>
  <body>
    <input type = "button" onclick = "sayHello()" value = "Say Hello" />
  </body>
</html>
```

Add a javascript file, hello.js

```
function sayHello() {
  alert("Hello World")
}
```

One more such example is given below: for index.html

```
<html>
   <head>
      <title>Static File</title>
      <link href="{{url_for('static', filename = 'style.css')}}" rel="stylesheet">
      <script src="{{url_for('static', filename = 'script.js')}}"></script>
   </head>
   <body>
      <h3>Hello World!</h3>
   </body>
</html>
```

5.4 Flask Request Object (Forms, args, files, redirect)

A Flask Request Object is an object that contains all of the data about a particular HTTP request. It contains information about the HTTP method, as well as any data that was sent in with that request, such as form fields or file uploads. It also has methods for accessing information in the body of an HTTP request.

How it is different from regular request?

- A Flask Request Object is a request object that has been extended with additional functionality. This makes it a more powerful and flexible request object.
- Flask Request Objects are great for making it easier to create RESTful APIs. They're also useful for handling data validation, or for creating custom methods on the fly.

Advantages

The advantages of using a Flask Request Object are:

- It can be used to fetch data from the database and pass it to templates.
- It is an easy way to write unit tests for your app.
- It can be used to handle cookies and sessions in your app.

Use of Request Object

A request object is a built-in object which is used to represent an HTTP request. It contains information such as the URL, headers, and data.

The request object is the first argument passed to the view function. It contains all of the information about what has been requested by the client. This includes, but is not limited to, the URL, headers, data, cookies and more. It can be used for many things such as authentication and authorization.

The flask object has a significant role to play because it is the request object that holds the data when the data is sent form an html form in the form of information until it reaches the server. The request object performs all the data processing, error handling, and other function too, while transferring the data to the server.

Or we can say that, In the client-server architecture, the request object contains all the data that is sent from the client-side to the server using HTTP methods like get, post. And all that data is temporarily stored in the request object till the server receives it.

SN	Attribute	Description
1	Methods	By which data is sent from the client-side. i.e. get or post.
2	File	It holds the info about the uploaded file.
3	Cookies	These are the temporary short files that hold the names and values of cookies, which helps to track user session on the client-side.
4	Args	Arguments are fetched from the URL address. It is part of the URL address that is declared in the URL address just after the '?'.
5	Form	It stores various key-value of form parameters.

5.4.1 Form request object, render_template() method, Form data handling

In the following example, the "URL" displays a web page ("customer1.html"), that contains a simple html form that is used to take the information as the input from the customer. The information filled by the user in the html form is sent to the http://localhost:5000/success using the **POST** method. The render_datafunction() function receives all information from the request object and displays it on the result_data.html page.

This application has three files, which are script2.py, customer.html, and result_data.html.

1. Script2.py

```
from flask import *
 app = Flask(__name__)
 @app.route('/')
 def customer():
```

```
return render_template('customer1.html')
@app.route('/success',methods = ['POST', 'GET'])
def render_datafunction():
if request.method == 'POST':
result = request.form
return render_template("result_data.html",result = result)
if __name__ == '__main__':
app.run(debug = True)
```

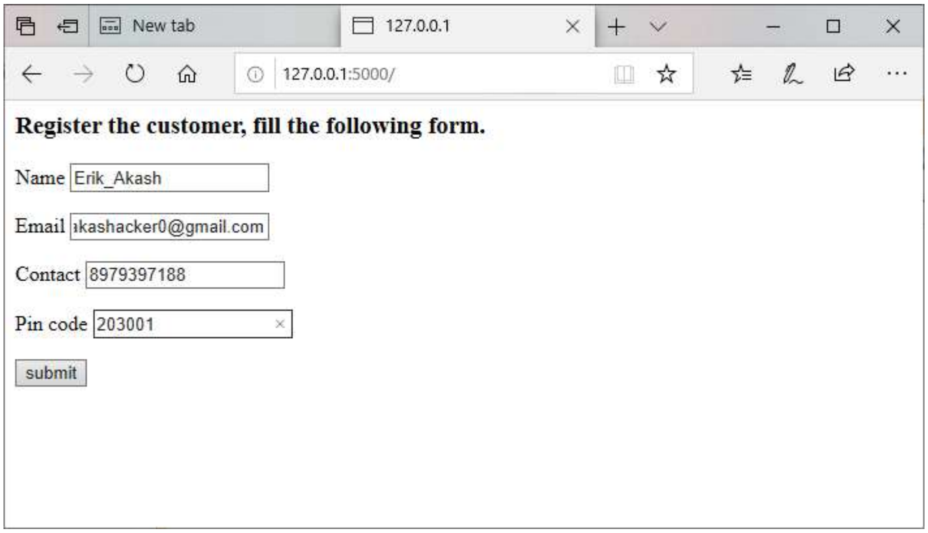
2. Customer1.html

```
<html>
 <body>
 <h3>Customer Registration</h3>
 <p>Fill this form.</p>
 <form action = "http://localhost:5000/success" method = "POST">
 <p>Name <input type = "text" name = "entername" /></p>
 <p>Email <input type = "email" name = "enteremail" /></p>
 <p>Contact <input type = "text" name = "entercontact" /></p>
 <p>Pin code <input type ="number" name = "pin" /></p>
 <p><input type = "submit" value = "submit" /></p>
</form>
 </body>
 </html>
```

3. Result_data.html

```
<!doctype html>
 <html>
 <body>
 <p><strong>Thanks for the registration. Confirm your details</strong></p>
 <table border = 1>
 {% for key, value in result.items() %}
 <tr>
 <th> {{ key }} </th>
 <td> {{ value }} </td>
 </tr>
 {% endfor %}
 </table>
 </body>
 </html>
```

Firstly you must run the script.py file by using the command **python script.py**. This will start the development server on the localhost:5000, which can be later accessed on the browser, as given below:



Register the customer, fill the following form.

Name Erik_Akash

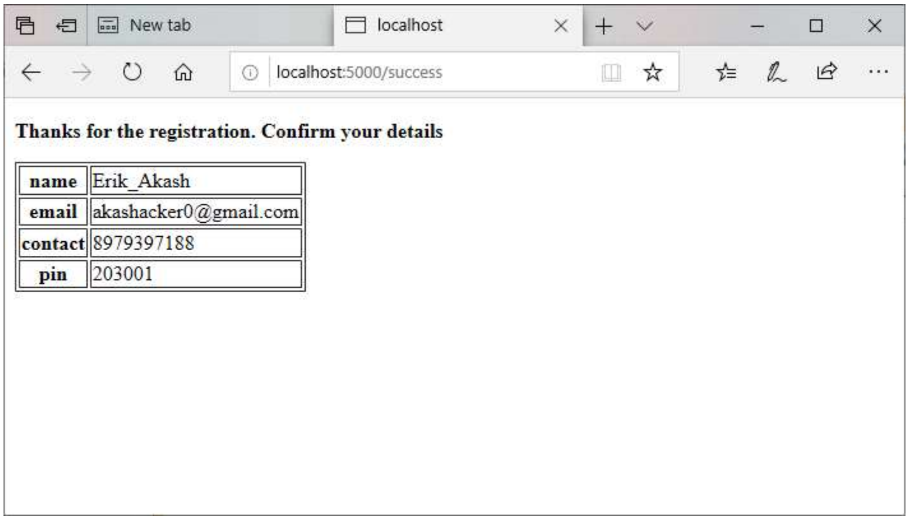
Email ıkashacker0@gmail.com

Contact 8979397188

Pin code 203001

submit

Now click on the submit button, and output is shown in the given screenshot.



Thanks for the registration. Confirm your details

name	Erik_Akash
email	akashacker0@gmail.com
contact	8979397188
pin	203001

flask.render_template(template_name_or_list, **context)

Renders a template from the template folder with the given context.

Parameters

- template_name_or_list (Union[str, jinja2.environment.Template, List[Union[str, jinja2.environment.Template]]]) – the name of the template to be rendered, or an iterable with template names the first one existing will be rendered
- context (Any) – the variables that should be available in the context of the template.

Return type

str

5.4.2 Flask Session, Creating Session, Session Variable, Session.pop()

In the session, the data is saved on the server. It can be determined as a time interval in which the client accesses the server until the user logs off. The data between them is stored in a temporary folder on the server. Each user is assigned a specific session ID. The Session object is a dictionary that contains the key-value pair of variables associated with the session. A SECRET_KEY is used to store encrypted data in the cookie.

The concept of session is very similar to that of a cookie. However, session data is stored on the server.

The session can be defined as the duration during which a user log in into the server and disconnects. The data used to track this session is stored in the temporary directory of the server.

Session data is stored on top of cookies, and the server cryptographically signs.

The syntax which is used to set the session variable to a specific value on the server are given below:-

For example:

```
Session[key]= value   // stores the session value
```

To remove a session variable, use the pop() method on the session object and mention the variable to be removed.

```
Session.pop(key, None)  // releases a session variable
```

Let's see a simple example to understand how we can set and get the session variable.

Session.py

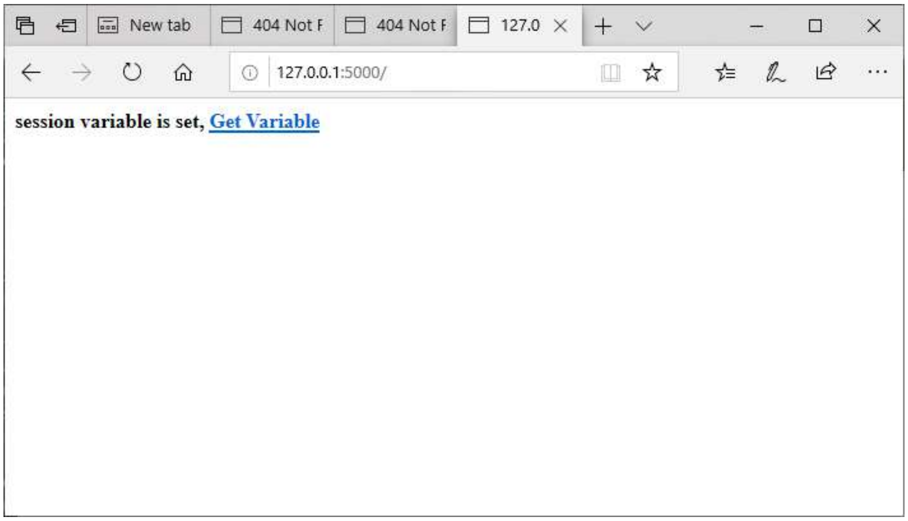
```
from flask import *
 app = Flask(__name__)
 app.secret_key = "abc"
 @app.route('/')
 def home():
 res = make_response("<h4>session variable is set, <a href='/get'>Get Variable</a></h4>")
 session['response']='session#1'
 return res;
 @app.route('/get')
 def getVariable():
 if 'response' in session:
 s = session['response'];
 return render_template('getsession.html',name = s)
 if __name__ == '__main__':
 app.run(debug = True)
```

getsession1.html

```
<html>
 <head>
```

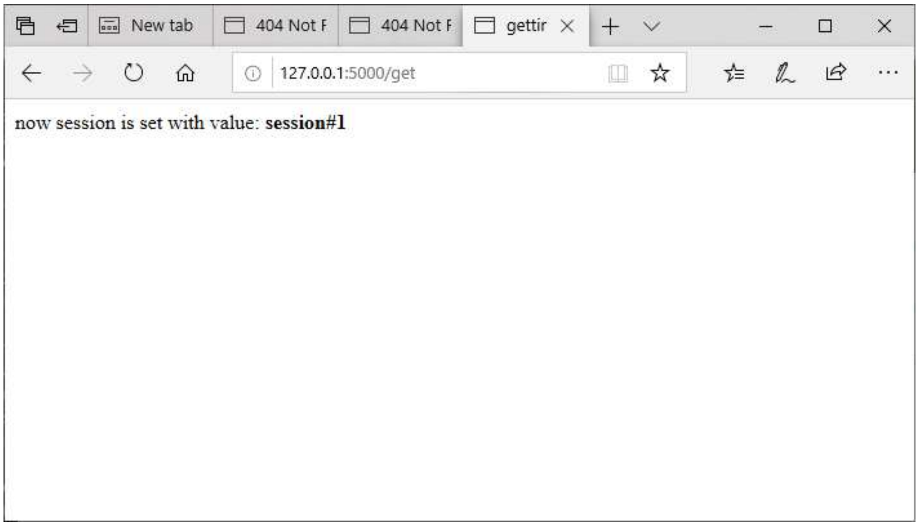
```
<title>getting the session</title>
</head>
<body>
<p>now  session is set with value: <strong>{{name}}</strong></p>
</body>
</html>
```

Now run the Session.py file



session variable is set, Get Variable

Now click on the get variable.



now session is set with value: **session#1**

Login Application in the flask using Session

Now let's create a login application in a flask where a home page is shown to the user as given below

Loginpage.py

```
from
flask import *
```

```
app = Flask(__name__)
app.secret_key = "ayush"
@app.route('/')
def home():
 return render_template("homepage2.html")
@app.route('/login')
def login():
 return render_template("loginpage3.html")
@app.route('/success',methods = ["POST"])
def success():
 if request.method == "POST":
  session['email']=request.form['email']
  return render_template('success3.html')
 @app.route('/logout')
 def logout():
 if 'email' in session:
 session.pop('email',None)
 return render_template('logoutpage2.html');
 else:
 return '<p>user already logged out</p>'
 @app.route('/profile')
 def profile():
  if 'email' in session:
  email = session['email']
  return             render_template('profile.html',name=email)
  else:
  return '<p>Please login first</p>'
  if __name__ == '__main__':
  app.run(debug = True)
```

homepage2.html

```
<html>
 <head>
 <title>home</title>
 </head>
 <body>
 <h3>Welcome to the website</h3>
 <a href = "/login">login</a><br>
 <a href = "/profile">view profile</a><br>
 <a href = "/logout">Log out</a><br>
 </body>
 </html>
```

Loginpage3.html

```
<html>
 <head>
  <title>login</title>
 </head>
 <body>
  <form method = "post" action = "http://localhost:5000/success">
      <table>
           <tr><td>Email</td><td><input type = 'email' name = 'email'></td></tr>
           <tr><td>Password</td><td><input type = 'password' name = 'pass'></td></tr>
           <tr><td><input type = "submit" value = "Submit"></td></tr>
        </table>
     </form>
 </body>
 </html>
```

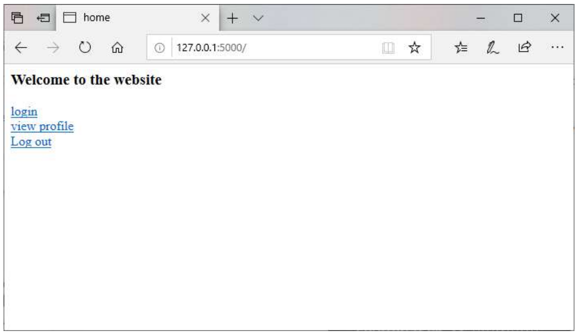
Success.html

```
<html>
 <head>
 <title>success</title>
 </head>
 <body>
   <h2>Login successful</h2>
     <a href="/profile">View Profile</a>
 </body>
 </html>
```

Logoutpage2.html

```
<html>
 <head>
  <title>logout</title>
 </head>
 <body>
 <p>logout successful, click <a href="/login">here</a> to login again</p>
 </body>
 </html>
```

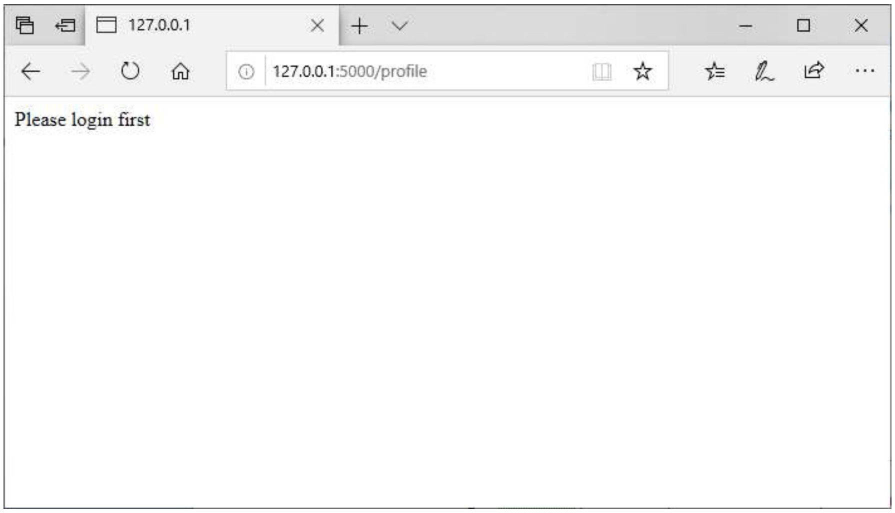
Now lets run the loginpage1.py file from cmd as shown below



Welcome to the website

login
view profile
Log out

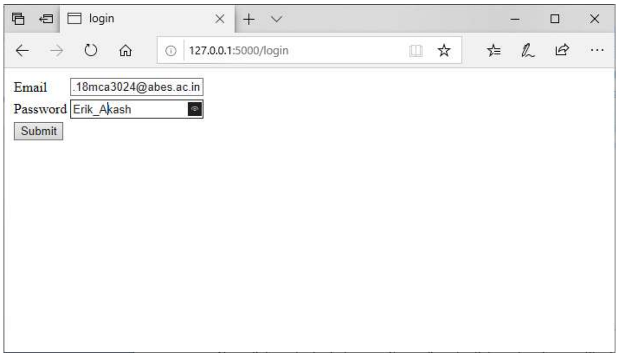
Now click on the login button, if you directly click on the view profile then it may show some warning as shown below



Please login first

Now click on the login button.

Now click on the submit button as shown.

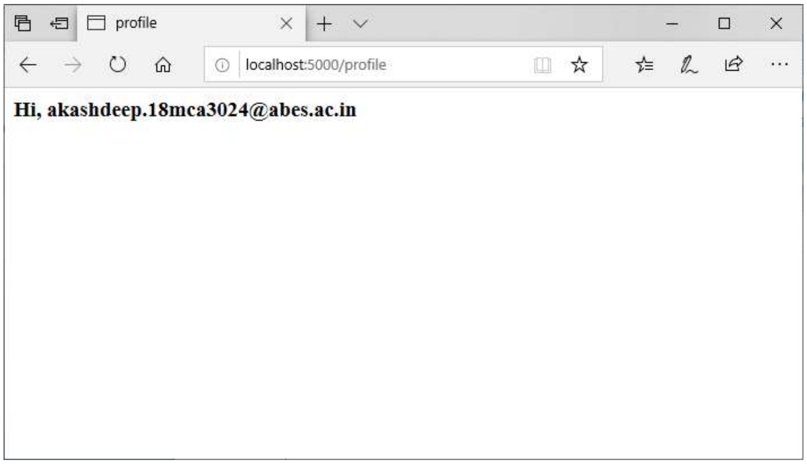


Email .18mca3024@abes.ac.in

Password Erik_Akash

Submit

Now click on the view profile button.



Hi, akashdeep.18mca3024@abes.ac.in

5.4.3 File Uploading: request.files(), object, save() method, saving file to specific folder

Uploading files using the Flask framework is a straightforward task. You need an HTML form with the enctype attribute and the URL manager, which retrieves the file and saves the object in the desired location. These files are temporarily stored on the server and then in the desired location.

Or we can say that **File Uploading** is the process of transmitting binary or standard files to the server. Flask makes it easy for us to upload files. All we need is to have an HTML form with a set of multipart / form-data encryption.

The server-side globe script retrieves the requested object file using the request — files [] object. Once the file is loaded correctly, it is saved in the desired location on the server.

The uploaded file is saved in the temporary directory of the server for a while before being saved in the desired location.

However, we can mention the path to the folder in which the file should be uploaded to the server, and the maximum size of the uploaded file should also be mentioned. This can be done in the configuration settings of the global object.

```
app.config['MAX_CONTENT-PATH'] = it is used to specify the maximum size of the file to be uploaded.

app.config['UPLOAD_FOLDER']      = it is used to  specify the location or the upload folder.
```

We can easily understand the file uploading with the help of an example

In this example, we will provide a file selector (file_upload_form.html) to the user where the user can select a file from the file system and send it to the server.

On the server-side, the file is retrieved using the request.files ['file'] object and saved in the server location.

Fileuploadform1.html

```
<html>
 <head>
 <title>upload</title>
 </head>
 <body>
 <form action = "/success4" method = "post" enctype="multipart/form-data">
 <input type="file" name="file" />
 <input type = "submit" value="Upload">
 </form>
 </body>
 </html>
```

Success4.html

```
<html>
 <head>
 <title>success</title>
```

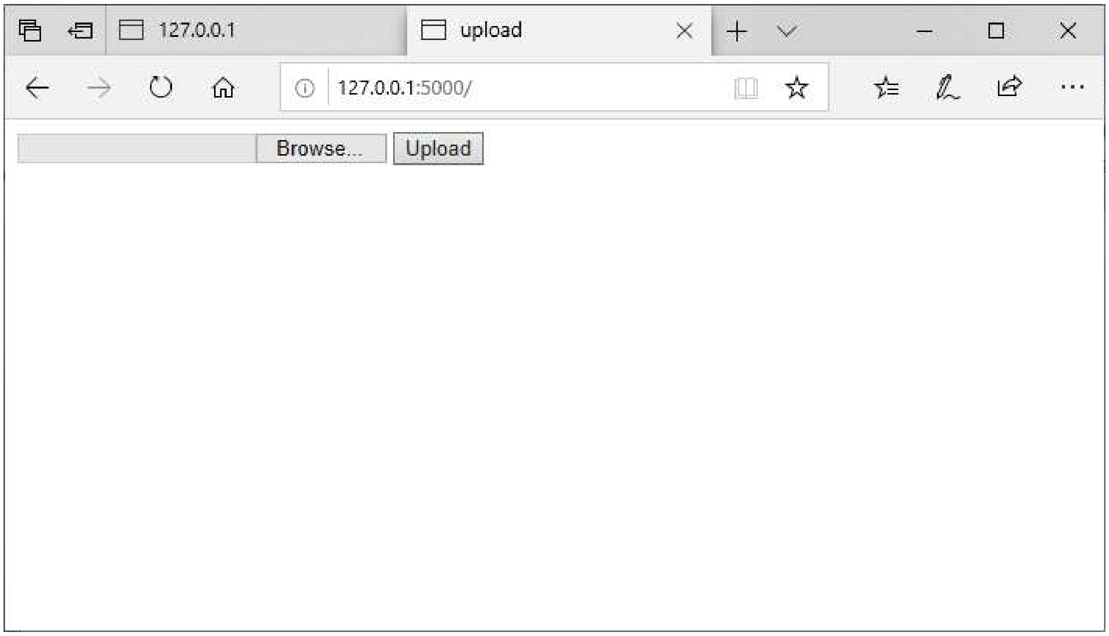
```
</head>
<body>
<p>File  has been uploaded successfully</p>
<p>File Name: {{name}}</p>
</body>
</html>
```

Uploladpagefile.py

```
from flask import *
 app = Flask(__name__)
 @app.route('/')
 def upload():
 return render_template("fileuploadform1.html")
 @app.route('/success', methods = ['POST'])
 def success():
 if request.method == 'POST':
 f = request.files['file']
 f.save(f.filename)
 return render_template("success4.html", name = f.filename)
 if __name__ == '__main__':
 app.run(debug = True)
```

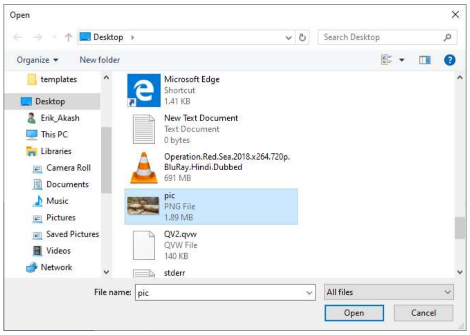
Now let's run the **uploadpagefile.py.**

An HTML form is displayed to the user.



Click on the **Browse** button so that we can explore the file system and search the appropriate file.

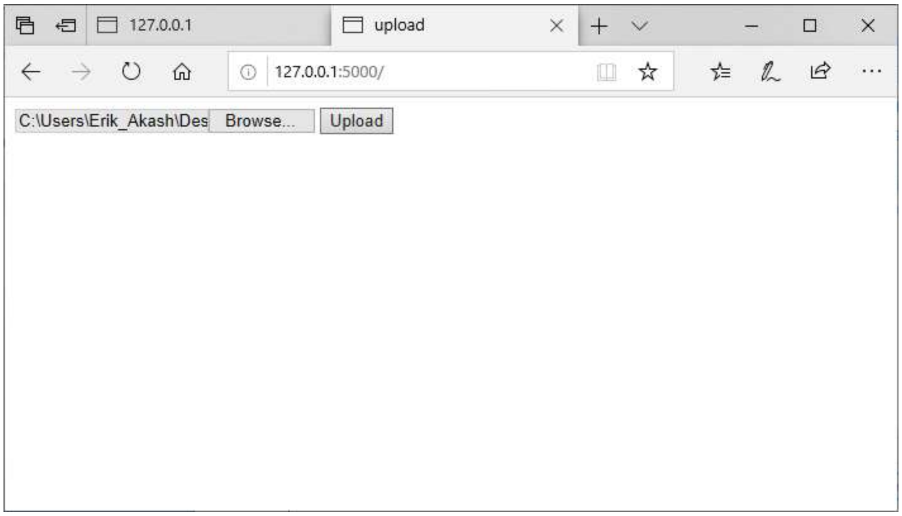
Now we have to choose a file, as shown below.



File name: pic

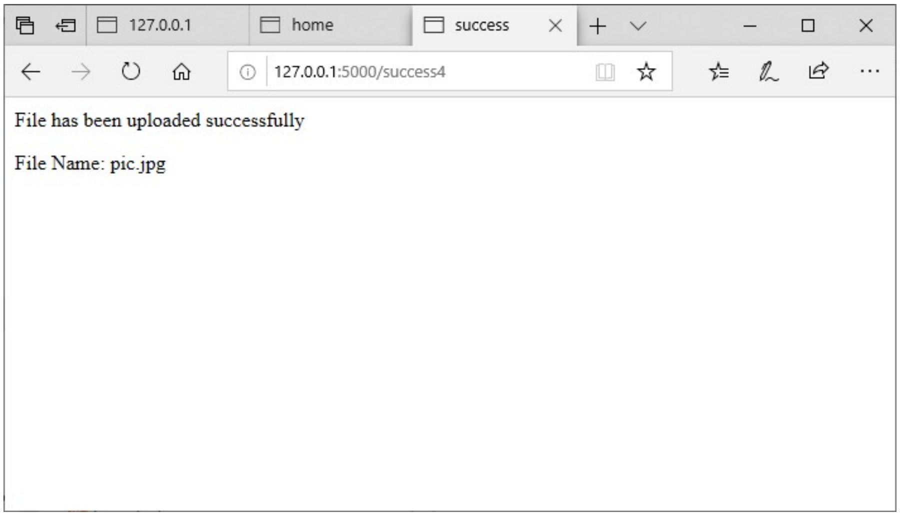
All files

Now double click on the file.



C:\Users\Erik_Akash\Des

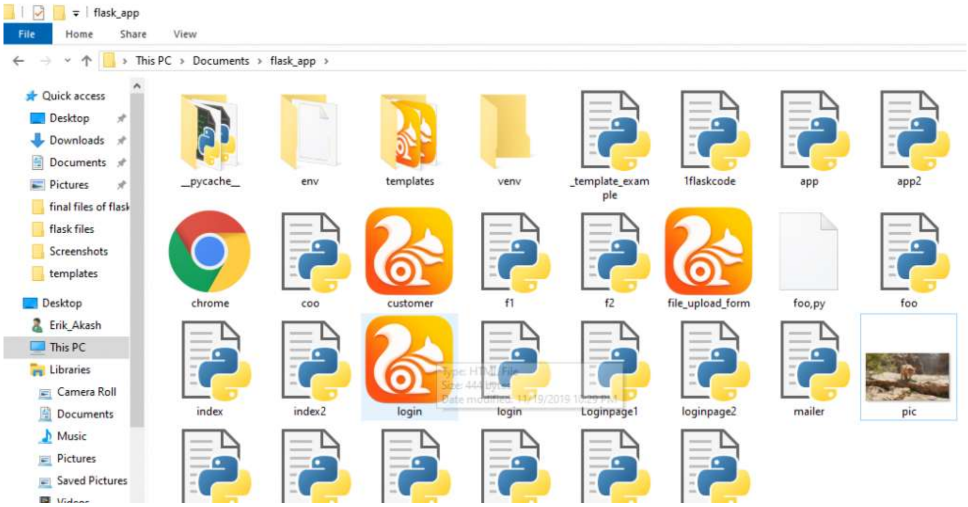
Here click on the **Upload** button. Then user get a message of successful file uploading as given below:-



File has been uploaded successfully

File Name: pic.jpg

We can confirm this by checking in the directory where upload.py is located, as shown in the image below.



5.4.4 Redirecting: redirect() method, location, status code and response

Redirect is also a function provided by the flask. It is called redirect() function. When this function is called, it returns a response object and also redirects the user to another location with the specific code.

The syntax of the redirect function is given below:

```
Flask.redirect(location, status code, response)
```

Here the **first parameter** is the location that is referred to as a target place where the user is going to be redirected.

The **second parameter** is the status code. It is sent to the header of the browser. The default code is 302, which stands for "found." And the third one is the response, and it is used to *instantiate the* response.

We can understand this with the help of an example.

First of all, we have to create three pages.

- Homepage=home3.html
- Login page =login3.html
- Python script=redirect.py

Both html file must be saved in the template folder that we made earlier flask_app/templates and .py file saved to the flask_app folder.

Home3.html

```
<html>
 <head>
 <title>home</title>
 </head>
 <body>
 <h3>Welcome</h3>
 <a href = "/login">login</a><br>
```

```
</html>
```

login3.html

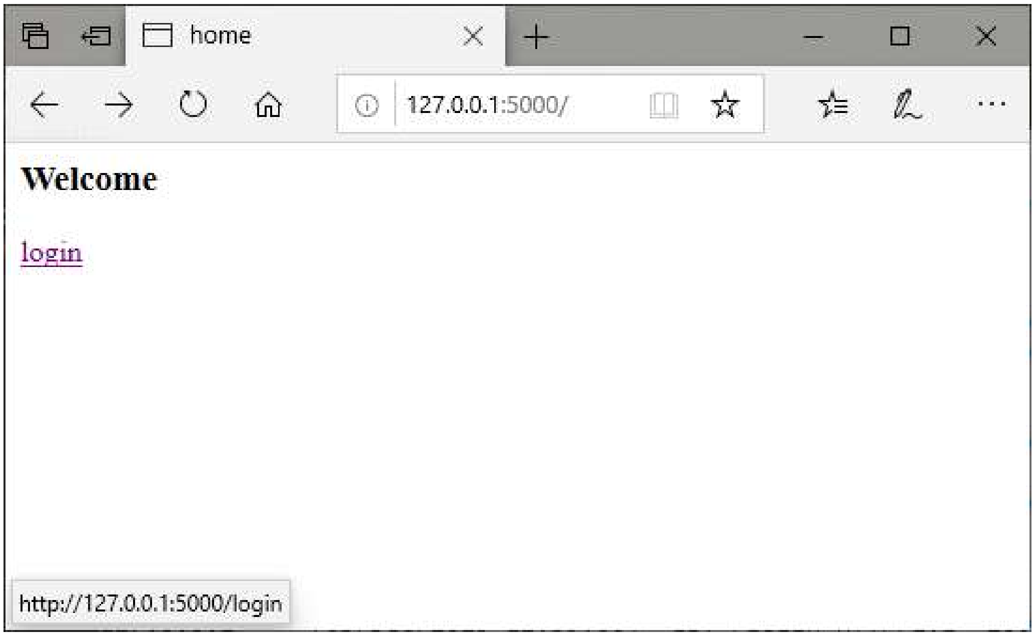
```
<html>
 <head>
  <title>login</title>
 </head>
 <body>
  <form method = "post" action = "http://localhost:5000/validate">
  <table>
  <tr><td>Email</td><td><input type = 'email' name = 'email'></td></tr>
  <tr><td>Password</td><td><input type = 'password' name = 'pass'></td></tr>
  <tr><td><input type = "submit" value = "Submit"></td></tr>
  </table>
  </form>
 </body>
 </html>
```

redirect.py

```
from
flask import *
 app = Flask(__name__)
 @app.route('/')
 def home_page ():
 return render_template("home3.html")
 @app.route('/login')
 def login_page():
 return render_template("login3.html");
 @app.route('/validate', methods = ["POST"])
 def validate():
 if request.method == 'POST' and request.form['pass'] == 'Akash':
 return redirect(url_for("success"))
 return redirect(url_for("login"))
 @app.route('/success')
 def success():
 return "logged in successfully"
 if __name__ == '__main__':
 app.run(debug = True)
```

Run the redirect.py file, and the output is given below.

Now click on the login.

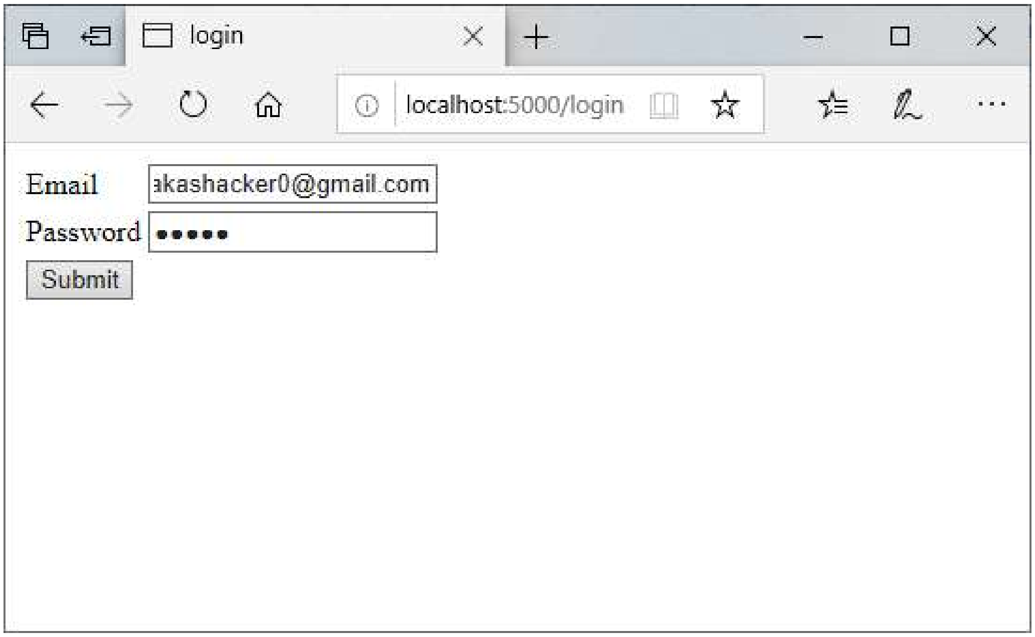


Welcome

login

http://127.0.0.1:5000/login

After clicking on the login button, a html form will appear. Fill this form and click on the Submit button.



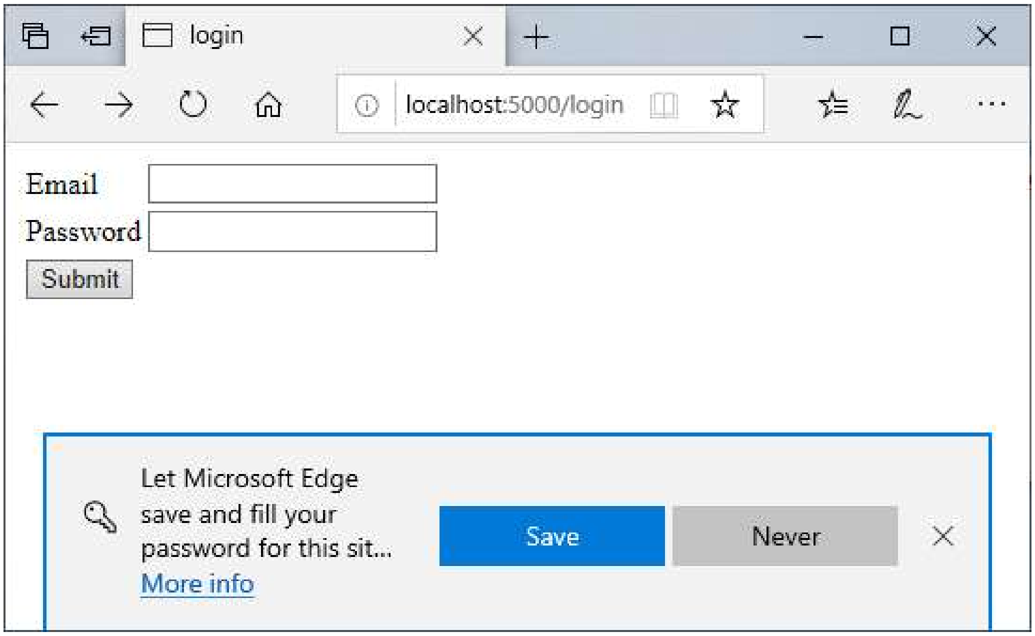
Email: akashacker0@gmail.com

Password: •••••

If the user enters the wrong details like an incorrect email id or password, then the page is going to redirect to the same page again and again until the user enters the correct details.

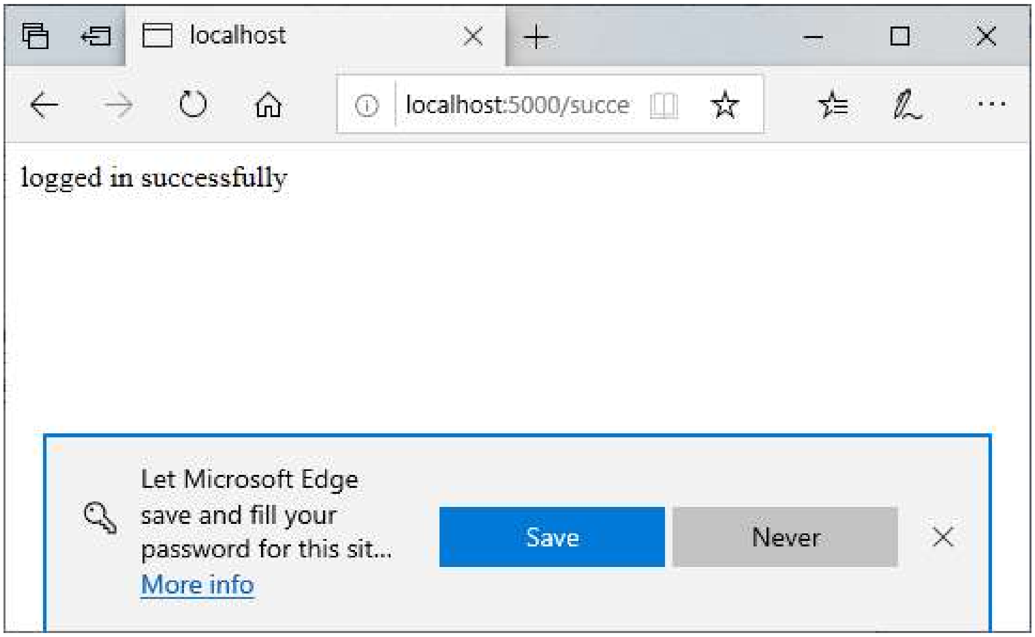
Example:

If the user enters an incorrect password, then it redirects the login page back again, and it will show notification, i.e., You have entered incorrect Email or password.



Let Microsoft Edge save and fill your password for this sit...

When the correct key is entered, the output is.



```
logged in successfully
```

Let Microsoft Edge save and fill your password for this sit...

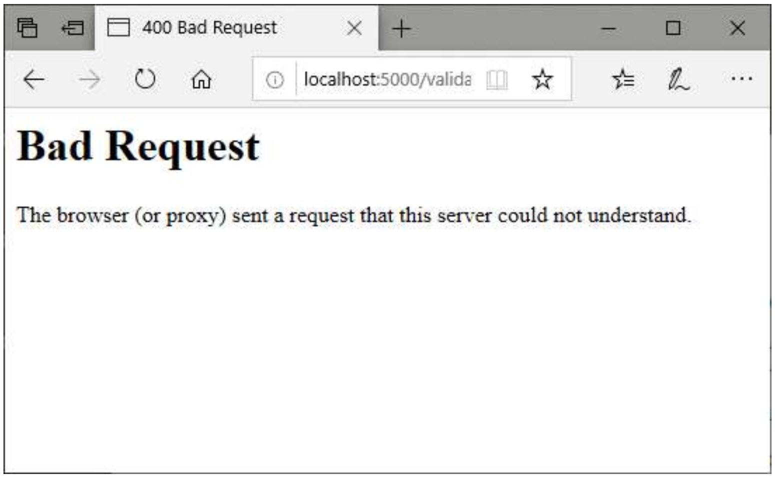
Abort function is a function of the flask. It is used in particular scenarios, in which some kind of errors occurred from the client-side in the request. It also provides a plain blank white error page for the user with basic information about the error.

Syntax of abort() function.

Flask.abort(code)

Some of the most common error codes depend on the particular errors are as follows.

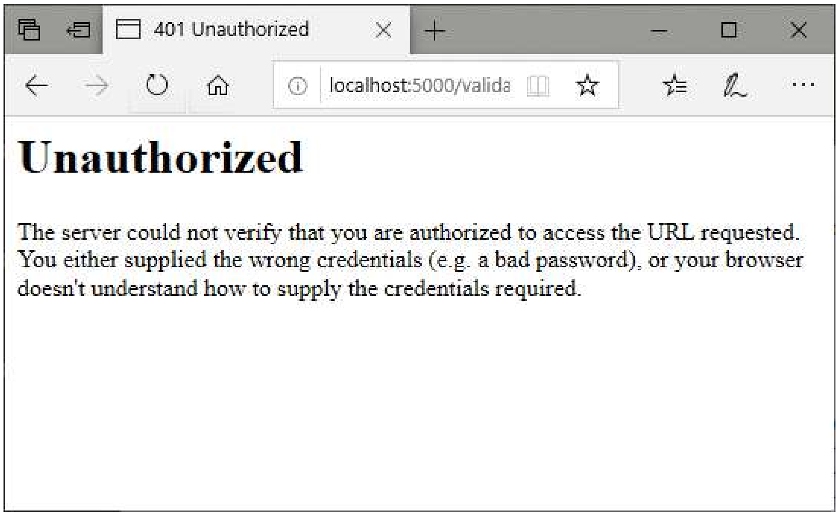
- 400



Bad Request

The browser (or proxy) sent a request that this server could not understand.

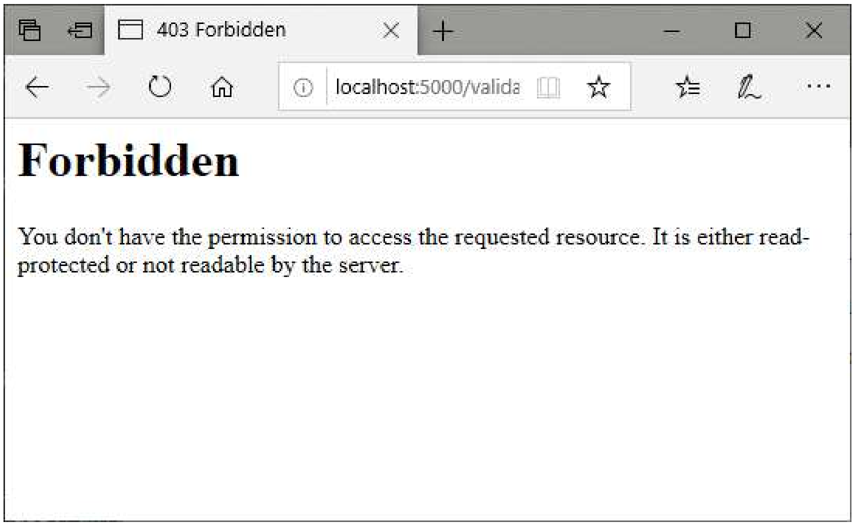
- 401



Unauthorized

The server could not verify that you are authorized to access the URL requested. You either supplied the wrong credentials (e.g. a bad password), or your browser doesn't understand how to supply the credentials required.

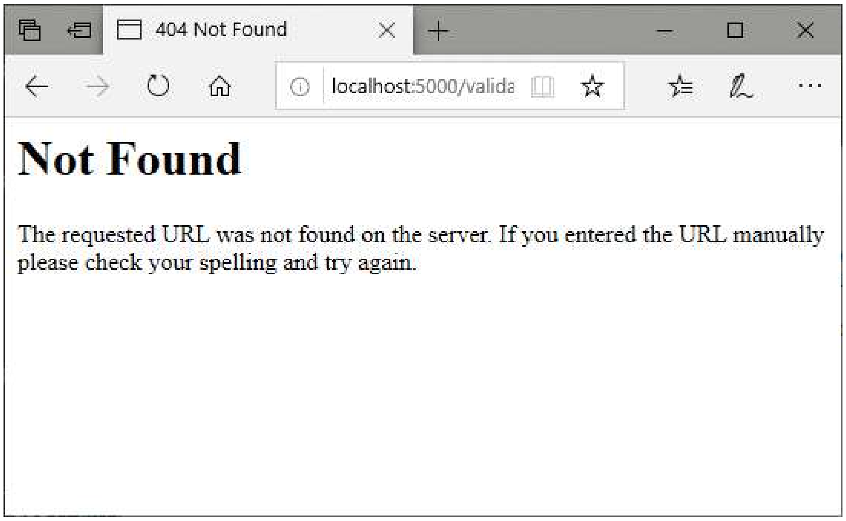
- 403



Forbidden

You don't have the permission to access the requested resource. It is either read-protected or not readable by the server.

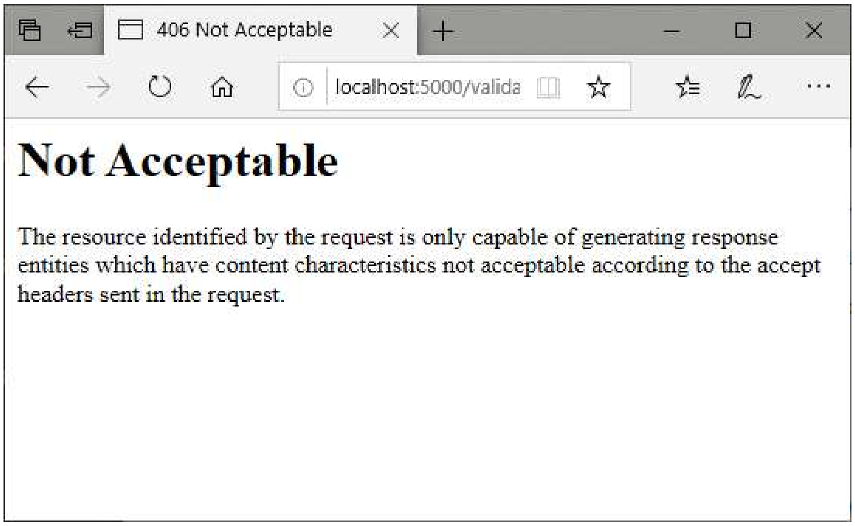
- 404



Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

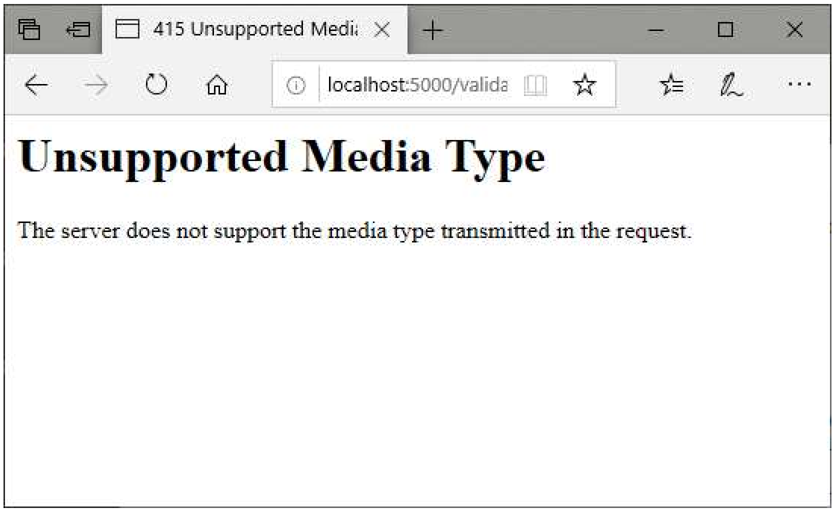
- 406



Not Acceptable

The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.

- 415



Unsupported Media Type

The server does not support the media type transmitted in the request.

We can understand the working of the abort() function with the help of an example.

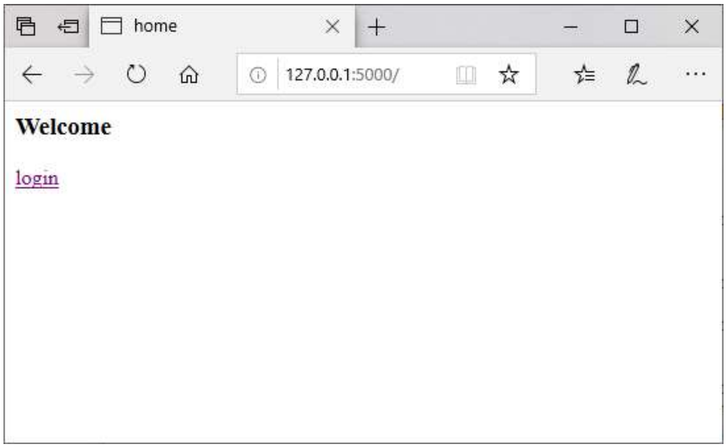
Note:- In this example, we use both the html templates and the same files that we used in the example of redirect() function, so we don't have to create new files.

Abort1.py

```
from flask import *
 app = Flask(__name__)
 @app.route('/')
 def home_page ():
  return render_template("home3.html")
 @app.route('/login')
 def login_page():
 return render_template("login3.html");
 @app.route('/validate', methods = ["POST"])
 def validate_code():
 if request.method == 'POST' and request.form['pass'] == 'Akash':
 return redirect(url_for("success"))
  else:
 abort(400)
 @app.route('/success')
 def success():
  return "logged in successfully"
  if __name__ == '__main__':
  app.run(debug = True)
```

Output

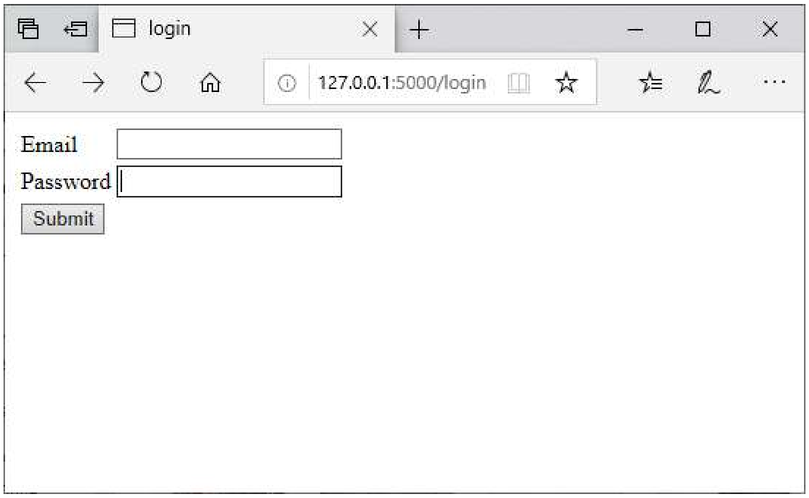
Click on the login.



Welcome

login

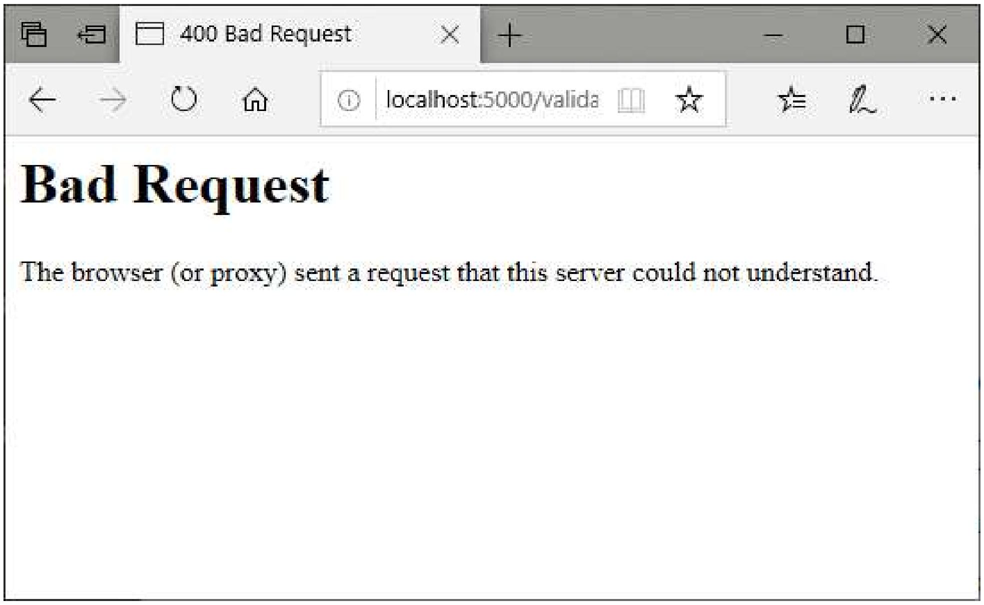
Fill the correct Email because the code has validation. Click on the Submit button to see how abort() function works.



Email

Password

Submit



Bad Request

The browser (or proxy) sent a request that this server could not understand.