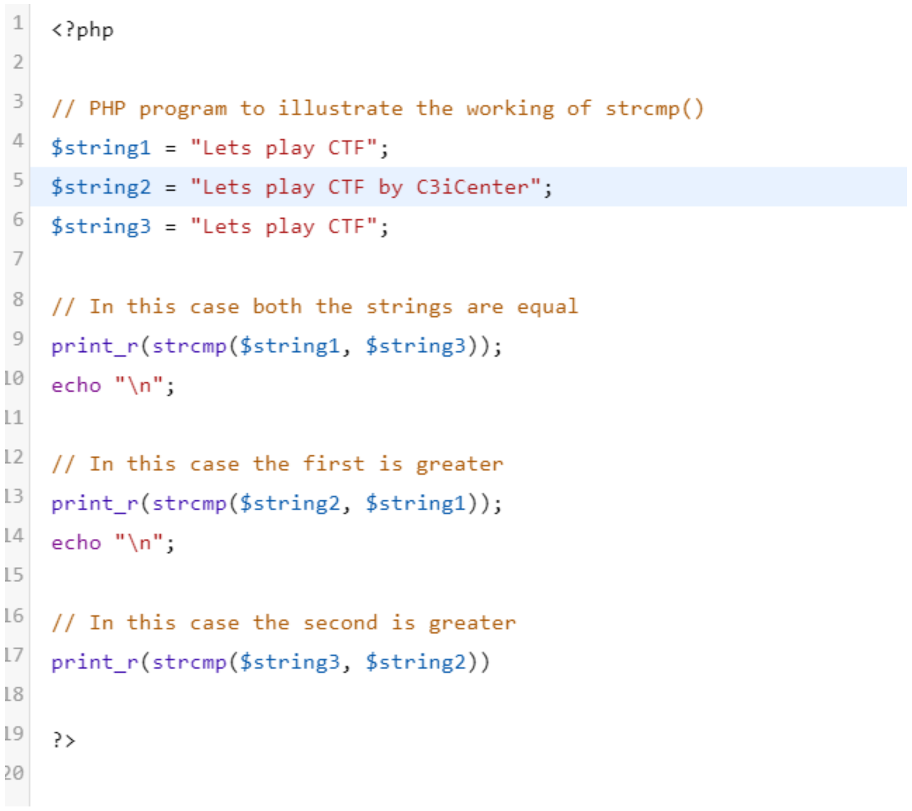
Unit-IV
String Manipulation and Regular Expression
Lecture 4: Comparing Strings, Searching & Replacing Strings

Comparing Strings:

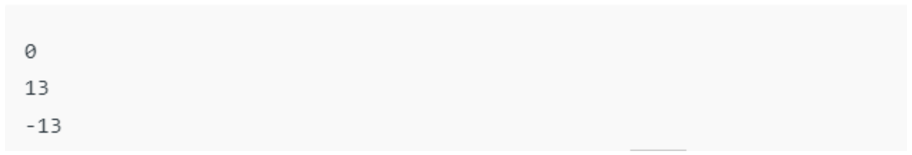
- **strcmp():** compares two strings.
  - It is case-sensitive i.e. capital and small cases will be treated differently, during comparison.
  - This function compares two strings and tells us that whether the first string is greater or smaller than the second string or equals to the second string.
  - **Syntax:**
    strcmp($string1,$string2)

The function returns a random integer value depending on the condition of match, which is given by:

- returns 0 if the strings are equal.
- returns a negative value (<0), if $string2 is greater than $string1.
- returns a positive value(>0), if $string1 is greater than $string2.



```php
<?php

// PHP program to illustrate the working of strcmp()
$string1 = "Lets play CTF";
$string2 = "Lets play CTF by C3iCenter";
$string3 = "Lets play CTF";

// In this case both the strings are equal
print_r(strcmp($string1, $string3));
echo "\n";

// In this case the first is greater
print_r(strcmp($string2, $string1));
echo "\n";

// In this case the second is greater
print_r(strcmp($string3, $string2))

?>
```



```
0
13
-13
```

- **strnatcmp() function:** This function compares two strings using a natural order algorithm and returns a positive integer, negative or zero.
    - This function is case-sensitive.
    - Natural ordering means this functions compares the strings that orders alphanumeric strings in the way a human being would.
    - **Syntax:**
      strnatcmp ($string1, $string2)
    - In a natural algorithm, the number 2 is less than the number 10 whereas in computer sorting, 10 is considered to be less than 2 as the first number in "10" is less than 2.
    - e.g. strcmp() would order the string "2" as greater than the string "12" because it is lexicographically greater but strnatcmp() would do it the other way around.

- **strcasecmp():** This function is also used to compare two given strings.
    - It is case-insensitive.
    - This function is similar to strncasecmp(), the only difference is that the strncasecmp() provides the provision to specify the number of characters to be used from each string for the comparison.
    - **Syntax: strcasecmp($string1, $string2)**

Matching and replacing substrings with string functions:

There are some built-in functions in php that are used to check if a particular substring is present in a larger string.

- **strtstr() :** It searches for the first occurrence of a string inside another string and displays the portion of the latter starting from the first occurrence of the former in the latter.
    - The function is case-sensitive.
    - **Syntax: strstr(str, search, before)**
      where, str → string to search
      search → the string to search for
      before → a Boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrences of the search parameter.

      e.g.
      ```
      <?php
      echo strstr("ctf Center!", "c");
      ?>
      ```

      O/P:
      ```
      tf Center
      ```

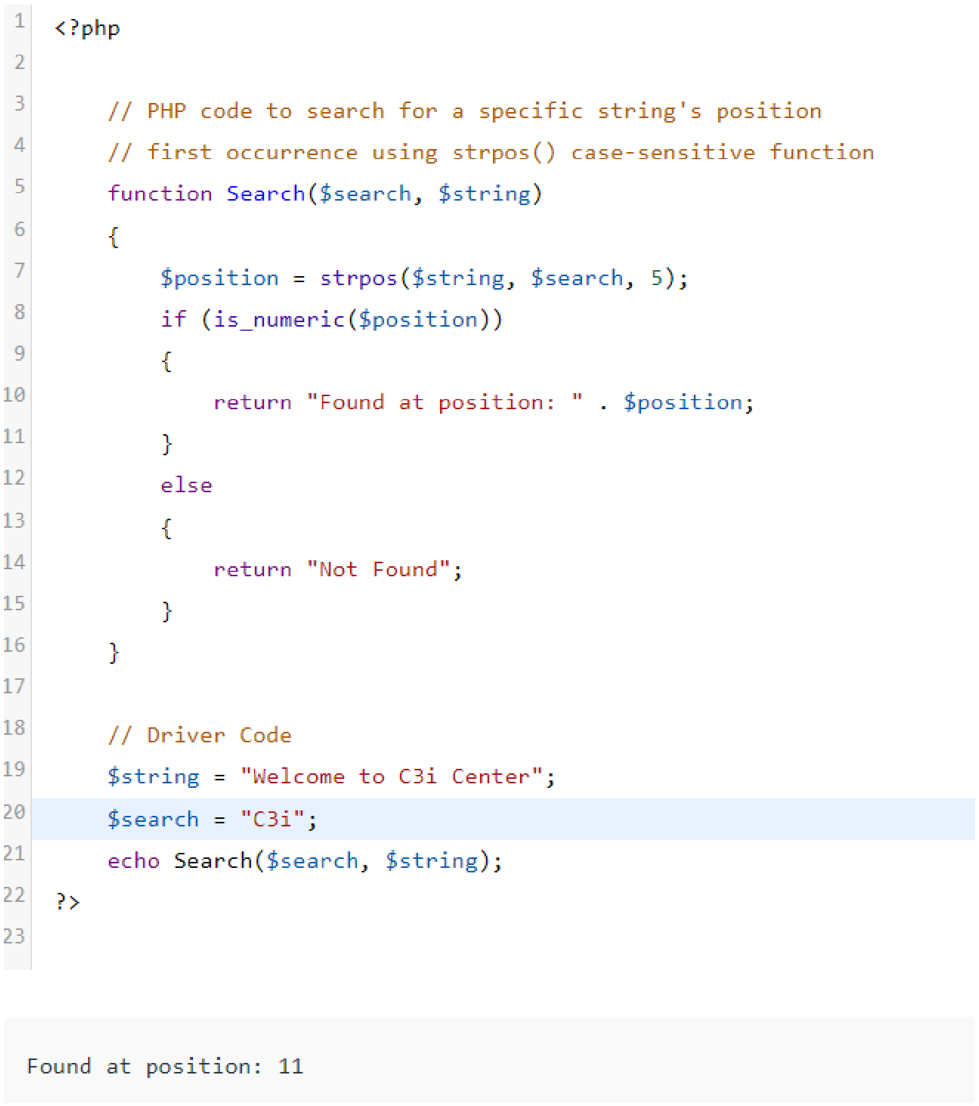
- **strchr() :** is used to find the first appearance of a string inside another sting.
    - **It is the pseudonym of strstr() function** i.e. alias of strstr().
    - **Syntax: strchr($originalStr,$searchStr,$before_search)**

- **stristr():** this function is identical to strstr(). The only difference is that it is not case sensitive.

- **strrchr():** This function takes two arguments a string and a character. This function searches the given character in the given string and returns the portion of string starting from the last occurrence of the given character in that string.
    - **Syntax: strrchr($string,$key)**

Find the position of a substring:

- **strpos():** This function helps us to find the position of the first occurrence of a string in another string. This returns an integer value of the position of the first occurrence of the string.
    - This function is case-sensitive i.e. it treats upper-case and lower-case characters differently.
    - **Syntax: strops(original_str, search_str, start_pos);**
        - original_str: this is mandatory parameter that refers to the original string in which we need to search the occurrence of the required string.
        - search_str: this is a mandatory parameter that refers to the string that we need to search.
        - start_pos → is an optional parameter that refers to the position of the string from where the search must begin.
    - This function returns an integer value that represents the original_str where the string search_str first occurs.



```php
<?php

    // PHP code to search for a specific string's position
    // first occurrence using strpos() case-sensitive function
    function Search($search, $string)
    {
        $position = strpos($string, $search, 5);
        if (is_numeric($position))
        {
            return "Found at position: " . $position;
        }
        else
        {
            return "Not Found";
        }
    }

    // Driver Code
    $string = "Welcome to C3i Center";
    $search = "C3i";
    echo Search($search, $string);
?>

```

```
Found at position: 11
```

Replacing substrings:

- **str_replace():** Is used to replace all the occurrences of the search string or array of search strings by replacement string or array of replacement strings in the given string or array respectively.
    - **Syntax:**
      str_replace ($searchVal, $replaceVal, $subjectVal, $count)
    - This function accepts 4 parameters out of which 3 are mandatory and 1 is optional.
        - $searchVal: this parameter can be of both string and array types. This parameter specifies the string to be searched and replaced.
        - $replaceVal: this parameter can be of both string and aray types. This parameter specifies the string with which we want to replace the $searchVal string.

        - $subjectVal: this parameter can be of both string and array types. This parameter specifies the string or array of strings which we want to search for $searchVal and replace with $replaceVal.
        - $count: this parameter is optional and if passed, its value will be set to the total number of replacement operations performed on the string $subjectVal.
    - This function returns a string or an array based on the $subjectVal parameter with the replaced values.

- **substr_replace():** It is used to replace a part of a string with another string.
    - **The index in the original string from** which the replacement is to be performed needs to be passed as a parameter.
    - If desired, the length up to which the replacement is to be done can also be specified.
    - **Syntax: substr_replace($string,$replacement,$start,$length)**
        - The length value is optional and represents the point at which PHP will stop replacing. If you don't supply this value, the string will be replaced from start to the end of the string.
    - The string generated after replacement is returned. In case of an array of strings, the array is returned.