Assignment no.	2
Aim	Write a Python program to compute following operations on String: a) To display word with the longest length b) To determines the frequency of occurrence of particular character in the string c) To check whether given string is palindrome or not d) To display index of first appearance of the substring e) To count the occurrences of each word in a given string (Do not use string built-in functions)
Objective	To understand the concept of Strings in Python To apply string manipulation operations
Outcome	After executing this assignment, Students will be able to perform all the tasks without using built in functions. Students will be familiar with String module Students will be in position to use string manipulation operations
OS/Programming tools used	(64-Bit) 64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit)

Theory related to assignment:

In this assignment We shall Learn Data type String.

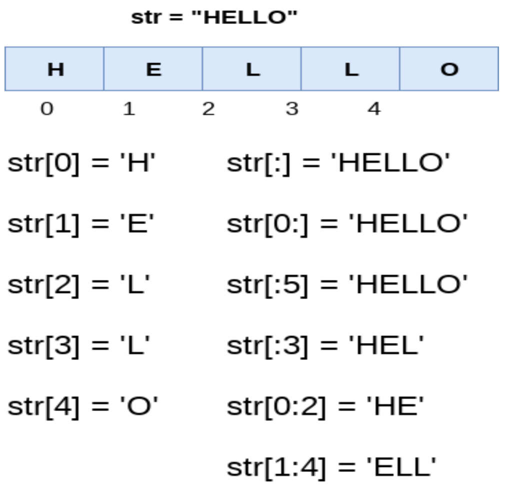
- Python treats strings as contiguous series of characters delimited by single, double or even triple quotes.
- Python has a built-in string class named "str" that has many useful features.
- We can simultaneously declare and define a string by creating a variable of string type. This can be done in several ways which are as follows:

```
name = "India"

graduate = 'N'

country = name

nationality = str("Indian")
```

In python, strings can be created by enclosing the character or the sequence of characters in the quotes. Python allows us to use single quotes, double quotes, or triple quotes to create the string.

str = "Hi PICT !"

print(type(str)), then it will **print** string (str).



```
str[0] = 'H'       str[:] = 'HELLO'
str[1] = 'E'       str[0:] = 'HELLO'
str[2] = 'L'       str[:5] = 'HELLO'
str[3] = 'L'       str[:3] = 'HEL'
str[4] = 'O'       str[0:2] = 'HE'
                   str[1:4] = 'ELL'
```

Indexing: Individual characters in a string are accessed using the subscript ([ ]) operator.

- The expression in brackets is called an index. The index specifies a member of an ordered set and in this case it specifies the character we want to access from the given set of characters in the string.
- The index of the first character is 0 and that of the last character is n-1 where n is the number of characters in the string.
- If you try to exceed the bounds (above n-1), then an error is raised.

Traversing a String: A string can be traversed by accessing character(s) from one index to another. For example, the following program uses indexing to traverse a string from first character to the last.

Example: Program to demonstrate string traversal using indexing

```
message= “Hello!”

index=0

for i in message:

        print(“message[“, index, “]=”, i)

        index+=1
```

Output:

```
message[ 0 ] = H

message[ 1 ] = e

message[ 2 ] = l
```

message[3] = l

message[4] = o

message[5] = !

a) To display word with the longest length :

Algorithm/Pseudocode:

Step 1: Read a Sentence

Step 2: Break up the sentence into an array of individual words (list)

Step 3: Initialize a counter variable

Step 4: Loop through each word in the array(list)

Step 5: Get the length of each word

Step 6: If the length is greater than the counter, set the counter

Step 7: Return the counter

Time Complexity: O(n), where n is the length of string.
Space: O(n), where n is the length of string.

===

b) To determines the frequency of occurrence of particular character in the string:

Algorithm/ Pseudocode:

1. Start
2. Declare a string
3. Ask the user to initialize it.
4. Use a frequency array to store the frequency of each character.
5. Convert the string to a character array (list)
6. Use two for loops to calculate the frequency of each element.
7. Use the first for loop to iterate through each character of the array(list).
8. Initialize each element of the frequency array as 1(list).
9. Use another for loop to iterate through the remaining characters.
10. Check for the total occurrence of the element.
11. If the element occurs again, increment the value in the frequency array(list).
12. Set the character array to 0 to avoid counting visited characters.
13. Print the characters and their corresponding frequency.
14. Stop.

Time Complexity: O(n).
Space: O(1)

==

c) To check whether given string is palindrome or not :

We are starting the algorithm by taking the string to be checked as input from the user. After that, the length of the string is calculated and stored in a variable, say ‘length’. To check whether a string is palindrome or not, the given string must be reversed. To store the reversed string, we are initializing a variable ‘rev’ as an empty string. That being done, we are starting a loop with initial value i = length – 1. In this loop, we are reversing the string, one character at a time by performing: rev = rev + character at position i. Here, the ‘+’ operator performs the concatenation of the characters of the string in reverse order. After that, the value of ‘i’ is decremented by 1. This loop runs until i >= 0. Once this loop ends, we have the reversed string in the variable rev.

We will now check whether both the strings are equal or not, ignoring the cases of the characters. If both the strings are equal, the given string is palindrome, else, the given string is not palindrome.

Algorithm/ Pseudocode:

```
Step 1. Start
Step 2. Read the string from the user
Step 3. Calculate the length of the string
Step 4. Initialize rev = “ ”[empty string]
Step 5. Initialize i = length - 1
Step 6. Repeat until i>=0:
        6.1: rev = rev + Character at position ‘i’ of the string
        6.2: i = i – 1
Step 7. If string = rev:
        7.1: Print “Given string is palindrome”
Step 8. Else:
        8.1: Print “Given string is not palindrome”
Step 9. Stop
```

Time complexity : O(n)
Space : O(1)
==

d) To display index of first appearance of the substring

Algorithm/Pseudocode:

Step1: Read String 1

Step2: Read substring 2

Step3: Apply loop till length of string 1

Step4: Apply Loop till length of string 2

Step5: Check if ith index of str1 is equal to jth index of str 2

Step6: If true, print I and come out of the loop using Break

Step7: Print Substring not present

Time Complexity: O(n).
Space: O(1)

===

e) To count the occurrences of each word in a given string

Algorithm/ Pseudocode:

```
Step1: Read String 1
Step2: Convert the string in to list
Step3: Create one more list CountList initialized with 1, of length 4
Step4:  apply for loop till length of list using counter variable i
Step5: apply for loop till length of list using counter variable j
Step6: check if lst [i] is equal to list[j]
Step7: if true, increment ith element of countlist
Step8: Complete internal loop of counter variable j
Step9: Complete outer loop of counter variable i
Step10: Display the countlist
```

Time Complexity: O(n).
Space: O(1)

==

Conclusion:

Students Successfully Executed the string manipulation operations without using string built in methods. Students learnt to find length of string, to calculate frequency of words in the string manually.

Review Questions:

1 What is the output of the following code ?

```
example = "snow world"

example[3] = 's'

print example
```

A. snow

B. snow world

C. Error

D. snos world

2. What is the output of “hello”+1+2+3 ?

A. hello123

B. hello

C. Error

D. hello6

3. Suppose i is 5 and j is 4, i + j is same as

A. i.__add(j)

B. i.__add__(j)

C. i.__Add(j)

D. i.__ADD(j)

4. The Index of the first character of the string is:

A. 0
B. 1
C. N-1
D. N

5. What is string in python explain String indexing and String traversing with examples.

6.Explain Concatenating, Appending and Multiplying Strings with examples.

7. Explain str () with examples.

8. String is mutable or immutable? Explain with an example.

9. Explain String formatting operator with example.

10. Explain any seven in-built string methods with example