

|  |
| --- |
| ASSIGNMENT 3 |
|  |
| Name:- Harsh Balaprasad Baheti  Div :- A  Roll No.:-SECOA112 |

# Assignment 3

|  |
| --- |
| **Objective :**   To understand the different string manipulation and operations.  **Outcome :**  Students will be able to perform different string operations.  **Theory:**  **What is String:**   The string can be defined as the sequence of characters represented in the quotation marks.  In python, we can use single, double, or triple quotes to define a string.  In the case of string handling, the operator + is used to concatenate two strings as the operation "hello"+" python" returns "hello python".  The operator \* is known as repetition operator as the operation "Python " \*2 returns "Python Python "  **String in Python with example:**  str1 = 'hello rahul' #string str1  str2 = ' how are you' #string str2  print (str1[0:2]) #printing first two character using slice operator  print (str1[4]) #printing 4th character of the string  print (str1\*2) #printing the string twice  print (str1 + str2) #printing the concatenation of str1 and str2    **Output:**  he  o  hello rahul hello rahul  hello rahul how are you  **Problem Statement :**  **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    **Algorithm :**  **1.** **longtest\_word Function:-**   1. **start** 2. **Intilise the longest as empty string.** 3. **for every i element in spilted list of input\_string jump to step4 else step6.** 4. **if length of i is greater than longest jump to step5 else step3.** 5. **set i to longest jump to step 3.** 6. **Return longest** 7. **exit()**   **2.** **frequency\_chr Function:-**   1. **start** 2. **Intilise the dictionary as an empty dictionary.** 3. **for every i character in input\_string jump to step4 else step5.** 4. **if key i is in dictionary then increment its value by 1 else set to 1 jump to step3.** 5. **Return dictionary** 6. **exit()**   **3.** **palindrome Function:-**   1. **start** 2. **for every i range from 0 to half the length of string +1 jump to 3 else jump to 5** 3. **if chr at 'i' and 'n-i-1' are same then jump to 2 else jump to 4.** 4. **return "not a palindrome" jump to 6.** 5. **return "a palindrome" jump to 6.** 6. **exit()**   **4.** **first\_indexFunction:-**   1. **start** 2. **if substr in input\_string jump to 3 else jump to 4.** 3. **return input\_string.find(substr) jump to 5.** 4. **Return "substring not present in string" jump to 5.** 5. **exit()**   **5.Counter Function:-**   1. **start** 2. **Intilise the dictionary as an empty dictionary.** 3. **for every i in spilted string jump to step4 else step5.** 4. **if key i is in dictionary then increment its value by 1 else set to 1 jump to step3.** 5. **Return dictionary** 6. **exit()**   **Program/Code:** |
| # a) To display word with the longest length def longtest\_string(input\_string):  longest = ""  for i in input\_string.split():  if len(longest) < len(i):  longest = i  return longest   # b) To determines the frequency of occurrence of particular character in the string def frequency\_chr(input\_string):  dictionary = {}   for i in input\_string:  dictionary[i] = dictionary.get(i**, 0**) + **1** return dictionary   # c) To check whether given string is palindrome or not def palindrome(input\_string):  for i in range(**0,** len(input\_string) // **2** + **1**):  if input\_string[i] == input\_string[len(input\_string) - i - **1**]:  continue  else:  return "not a palindrome"  return ' a palindrome'   # d) To display index of first appearance of the substring def first\_index(input\_string**,** substr):  if substr in input\_string:  return input\_string.find(substr)   # e) To count the occurrences of each word in a given string def counter(input\_string):  dictionary = {}  for i in input\_string.split():  dictionary[i] = dictionary.get(i**, 0**) + **1** return dictionary   string = input("Enter String : ") substring = input("Enter Substring : ")  print("Word with the longest length"**,** longtest\_string(string)) print("The frequencies of occurrence of particular character in the string"**,** frequency\_chr(string)) print("The string '{}' is {}".format(string**,** palindrome(string))) print("The index of first appearance of the substring "**,** first\_index(string**,** substring)) print("Count the occurrences of each word in a given string"**,** counter(string)) |
| **Output :**  Enter String : hello world it's harsh Enter Substring : harsh Word with the longest length hello The frequencies of occurrence of particular character in the string {'h': **3,** 'e': **1,** 'l': **3,** 'o': **2,** ' ': **3,** 'w': **1,** 'r': **2,** 'd': **1,** 'i': **1,** 't': **1,** "'": **1,** 's': **2,** 'a': **1**} The string 'hello world it**'s harsh** ' is not a palindrome The index of first appearance of the substring **17** Count the occurrences of each word in a given string {'hello': **1,** 'world': **1,** "it's": **1,** 'harsh': **1**} |

**Time Complexity :**

|  |  |  |
| --- | --- | --- |
| sr no. | Function | Time Complexity |
| 1 | longtest\_word | O(n) |
| 2 | frequency\_chr | O(n) |
| 3 | palindrome | O(n) |
| 4 | first\_index | O(n) |
| 5 | counter | O(n) |
| 6 | print | O(1) |

Total Time Complexity:= O(n)

Conclusion :-

Understood the different string manipulation and operations.