Aim: Write python programs to implement Strings: Basic string operations, string methods, decision making and looping statements etc.

Write python programs

a. To Count the Occurrences of Each Word in a Given String Sentence.
b. To find duplicate characters in a given string.
c. To remove punctuations from a string.
d. To Check if a Substring is Present in a Given String

Objective of the Experiment:

1. Understanding basic string operations and string methods etc.

Algorithms:

a. Algorithm to count the occurrences of each word in a given string sentence:

1. Take a string and a word from the user and store it in separate variables.
2. Initialize a count variable to 0.
3. Split the string using space as the reference and store the words in a list.
4. Use a for loop to traverse through the words in the list and use an if statement to check if the word in the list matches the word given by the user and increment the count.
5. Print the total count of the variable.
6. Exit.

b. Algorithm to find duplicate characters in a given string.

1. Take a string from user and store it in a separate variable.
2. Initialize two separate empty lists.
3. Keep track of first time occurring character in one list and duplicate character in the second.
4. Display the characters in the second list as duplicate characters.

d. Algorithm to check if a Substring is Present in a Given String

1. Take a string and a substring from the user and store it in separate variables.
2. Check if the substring is present in the string using find () in-built function.
3. Print the final result.
4. Exit.

Source code for the implementation:

(Write only important functions)

Post Lab Assignment:

1. What can we say about a number `n` if `h(n)` returns `True` for the function `h` given below?

```
def h(n):
  for i in range(2,n):
    if n%i == 0:
      return(True)
  return(False)
```

A. n is a multiple of 2
B. n is a composite number
C. n is a prime number
D. n has at least two distinct factors other than 1 and n

2. For an expression with parentheses, we can define the nesting depth as the maximum number of parentheses that are open when reading the expression from left to right. For instance, the nesting depth of "(33+77)(44-12)" is 1, while the nesting depth of "(a(b+c)-d)(e+f)" is 2.

   Write a Python function depth(s) that takes a string containing an expression with parentheses and returns an integer, the nesting depth of s. You can assume that s is well-parenthesized: that is, that is, every "(" has a matching ")" after it and every ")" has a matching "(" before it.

   Here are some examples to show how your function should work.

```
>>> depth("22")
0
>>> depth("(a+b)(a-b)")
1
>>> depth("(a(b+c)-d)(e+f)")
2
```

3. What is the value of `second` after executing the following lines? Trace the program and write value of `second after each iteration of for loop.`

```
first = "wombat"
second = ""
for i in range(len(first),0,-1):
  second = first[i-1] + second
```