|  |  |
| --- | --- |
| **Ex. 5** | **EXPLORING STRINGS** |
| **Date: 19 February 2024** | |
|  | |

**Aim:**

To explore strings in Python by writing programs for the following and executing them:

1. Given a string, find the list of palindromic sub-strings without using built-in functions.
2. List the number of sub-strings with the same first and last characters in a given string.
3. Break a given string into multiple lines with N characters each. Get the value of N from the user.
4. Sort the characters of a string based on their frequency of occurrence.
5. Concatenate two strings of the same length, without using built-in functions, such that the characters of each string are placed alternately in the resultant string.
6. Encode a given string using Caesar's cipher, where each letter in a string is replaced by a letter some fixed number of positions down the alphabet. Get the alphabetic string and the number of places each letter is to be shifted from the user.

**Algorithm:**

**( a )**

**Step – 1 :** Input the Main string from the user

**Step – 2 :** Loop through the string within a loop to get all the possible sub-strings.

**Step – 3 :** Check if the string is a palindrome by equating with its reverse.

**Step – 4 :** Print all the possible palindromes in the string

**( b )**

**Step – 1 :** Input the Main String from the user.

**Step – 2 :** Find the first occurence of the first character of the main string. Then find the number of characters that match the end character of the Main String after the occurence of the first character.

**Step – 3 :** Keep track of the count of the sub-strings needed. And Display it to the user at the end.

**( c )**

**Step – 1 :** Input the Main String from the user.

**Step – 2 :** Get the value of N from the user and while Looping, if you get N characters add a \n and then start counting N characters till the end of the string.

**Step – 3 :** Print the resultant character.

**( d )**

**Step – 1 :** Input the main string from the user.

**Step – 2 :** Firstly, count the number of same charcters for every characters and arrage them in ascending order.

**Step – 3 :** Print the arranged string

**( e )**

**Step – 1 :** Input the Main String from the user.

**Step – 2 :** Add the first character of each string to a new string, remove the first character of each string.

**Step – 3 :** Check if any string gets empty, then just add characters in the string that isn’t empty, else loop till both strings get empty.

**Step – 4 :** Print the resultant string.

**( f )**

**Step – 1 :** Get the Main String as input from the character.

**Step – 2 :** Add N(Number given by the user) to the ascii value of each chracter and also check if the ascii value exceed the alphabets. Add the character for the ascii value into a new string.

**Step – 3 :** Print the resultant string that is encrypted.

**Program:**

**( a )**

# Program to find the palindorme substrings in a main string

String = input("Enter the string : ")

for i in range(len(String)):

for j in range(i,len(String)):

if(String[i:j+1]==String[i:j+1][::-1]):

print(String[i:j+1],end=', ')

**( b )**

# Program to find the number of substrings having the same starting and ending character as the Main String

MainString = input("ENter the String : ")

Counts = 0

S = 0

E = 1

while(S < E):

pos1 = MainString.find(MainString[0],S)

if(pos1==-1):

break

E = S+1

while(MainString.find(MainString[-1],E)!=-1):

E = MainString.find(MainString[-1],E)+1

Counts+=1

S = pos1+1

print(Counts)

**( c )**

# Break the given string into lines with N characters specified by the user

MainString = input("Enter the String : ")

N = int(input("Enter the Value of N : "))

Lines = 0

while(Lines<(len(MainString)//N)):

MainString = MainString[0:(Lines \* N)+Lines]+ '\n'+MainString[(Lines \* N)+Lines:]

Lines+=1

print('Output :\n'+MainString.strip())

**( d )**

# Sort the given string based on the frequency of every letter

s = input("Enter the string : ")

CharsFrequency = {}

for i in s:

if i in CharsFrequency: CharsFrequency[i]+=1

else:CharsFrequency[i] = 1

Chars = list(CharsFrequency.keys())

Freq = list(CharsFrequency.values())

for i in range(len(Freq)):

for j in range(i,len(Freq)):

if Freq[i]>Freq[j]:

Freq[i],Freq[j],Chars[i],Chars[j] = Freq[j],Freq[i],Chars[j],Chars[i]

for j in range(len(Chars)):

print(Chars[j]\*Freq[j],end="")

**( e )**

# Merging two string in Zipper form

S1 = input("Enter the String - 1 : ")

S2 = input("Enter the String - 2 : ")

S3 = ''

while(not(S1 == '' and S2=='')):

if(S1 == ''):

S3 += S2[0]

S2 = S2[1:]

elif(S2 == ''):

S3 += S1[0]

S1 = S1[1:]

else:

S3 += S1[0] + S2[0]

S1 = S1[1:]

S2 = S2[1:]

print(S3)

**( f )**

# Caeser's Cipher | rotate characters own the alphabet for n times to get an encrypted messqage

M\_String = input("Enter the String : ")

N = int(input("Enter the Rotate Code : "))

E\_String = ''

for letter in M\_String:

if(ord(letter.lower())+N > ord('z')):

N\_letter = chr(ord('a')-1+ (ord(letter.lower())+N)-ord('z'))

elif(ord(letter.lower())+N > ord('a')):

N\_letter = chr(ord(letter.lower())+N)

else:

E\_String += letter

continue

if letter.isupper():

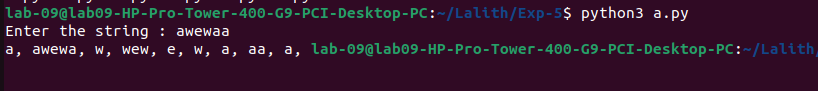
E\_String +=N\_letter.upper()

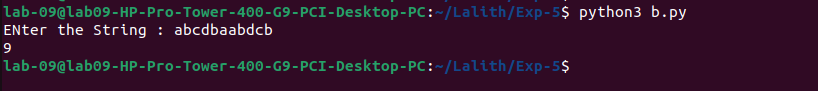
continue

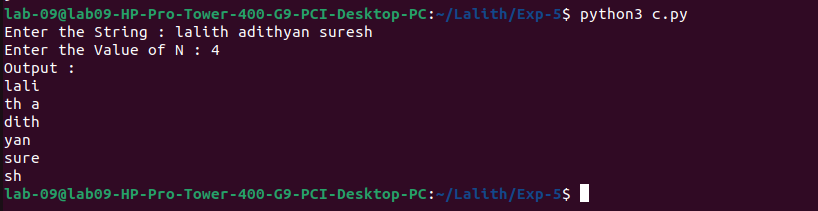
E\_String+=N\_letter

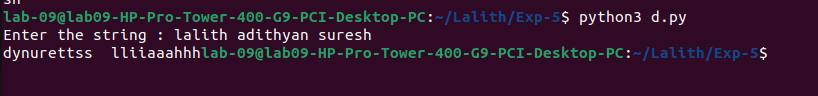
print(E\_String)

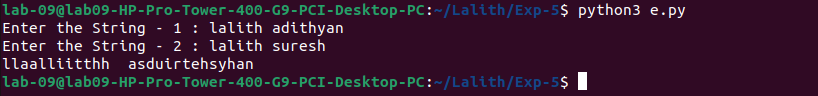
**Screenshot of Output:**

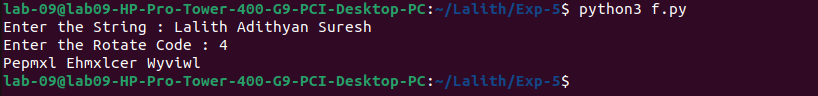
**( a )**

**( b )**

**( c )**

**( d )**

**( e )**

**( f )**

**Result:**

Thus, programs have been written and executed to explore strings in Python.