|  |  |
| --- | --- |
| **Ex. 5** | **EXPLORING STRINGS** |
| **Date: 19/02/24** |  |

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: Prompt the user to enter a string.

Step 2: Calculate the length of the string.

Step 3: Use nested loops to generate substrings:

- The outer loop determines the starting point of the substring.

- The inner loop determines the ending point of the substring.

Step 4: In each iteration of the inner loop, extract the substring from the current start to end index.

Step 5: Check if the first and last characters of the substring are the same.

Step 6: If they are the same, print the substring.

(b)

Step 1: Prompt the user to enter a string.

Step 2: Determine the length of the string.

Step 3: Identify the first character (a) and the last character (b) of the string.

Step 4: Use nested loops to generate and examine substrings:

- The outer loop sets the starting index of the substring.

- The inner loop sets the ending index of the substring.

Step 5: Extract each substring using the current start and end indices.

Step 6: Check if the first character of the substring matches a and the last character matches b.

Step 7: If both conditions are met, print the substring.

(c)

Step 1: Prompt the user for a string and the segment length n.

Step 2: Determine the length of the string.

Step 3: Initialize a counter to track characters printed.

Step 4: Iterate through the string:

- Print each character without moving to a new line.

- Increment the counter.

- If the counter equals n, print a newline and reset the counter.

(d)

Step 1: Prompt the user for a string.

Step 2: Initialize an empty dictionary to keep track of character frequencies.

Step 3: Iterate through each character in the string:

- If the character is already in the dictionary, increment its count.

- If not, add it to the dictionary with a count of 1.

Step 4: Sort the dictionary items by frequency in descending order.

Step 5: Create an empty list to store characters sorted by frequency.

Step 6: For each character-frequency pair in the sorted list, add the character to the new list, repeated by its frequency.

Step 7: Print the list of characters sorted by their frequency.

(e)

Step 1: Prompt the user for two strings.

Step 2: Calculate the lengths of both strings.

Step 3: Check if the lengths are equal:

- If they are, iterate through the strings:

- Print characters from both strings alternately without adding new lines.

- If they are not equal, display a message indicating invalid input.

(f)

Step 1: Prompt the user for a string and the number of shifts.

Step 2: Determine the length of the string.

Step 3: Iterate through each character in the string:

- If the character is not a space:

- Calculate its new ASCII value after applying the shift.

- Adjust the ASCII value to wrap around if it goes beyond 'z' or 'Z' (handling uppercase and lowercase separately).

- Convert the adjusted ASCII value back to a character.

- Print the character without moving to a new line.

- If the character is a space, print a space.

Program:

(a)

string=input("Enter a string:")

length=len(string)

for i in range(length):

   for j in range(i+1,length+1):

       k=string[i:j]

if(k[0]==k[-1]):

  print(k)

(b)

string=input("Enter a string:")

length=len(string)

a=string[0]

b=string[length-1]

for i in range(length):

   for j in range(i+1,length+1):

       k=string[i:j]

if(k[0]==a and k[-1]==b):

    print(k)

(c)

string=input("Enter a string:")

n=int(input("Enter the value of n:"))

length=len(string)

count=0

for i in range(length):

     print(string[i],end="")

     count+=1

     if(count==n):

        print()

        count=0

(d)

string = input("Enter a string: ")

frequency = {}

for char in string:

   if char in frequency: frequency[char] += 1

else:

   frequency[char] = 1

sorted\_chars = sorted(frequency.items(), key=lambda x: x[1], reverse=True)

lst2 = []

for char, freq in sorted\_chars:

    lst2.extend([char] \* freq)

print(lst2)

(e)

string1=input("Enter the first string:")

string2=input("Enter the second string:")

length1=len(string1)

length2=len(string2)

if(length1==length2):

   for i in range(length1):

       print(string1[i],end="")

       print(string2[i],end="")

else:

    print("Invalid string")

(f)

string=input("Enter the string:")

shift=int(input("Enter the number of shifts:"))

length=len(string)

for i in range(length):

    if(string[i]!=" "):

        ascii=ord(string[i])+shift

        if(ascii>=122):

          ascii=ascii-26

        elif(ascii>=90 and ascii<97):

          ascii=ascii-26

          char=chr(ascii)

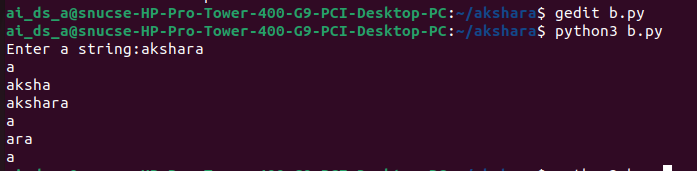
          print(char,end="")

          ascii=0

else:

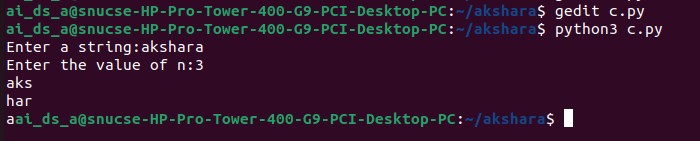
    print(" ",end="")

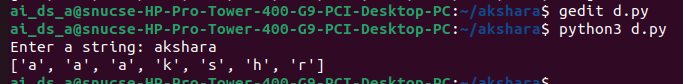
Screenshot of Output:

A computer screen shot of a computer

Description automatically generated (a)

(b)

(c)

(d)

(e)

**A screen shot of a computer program

Description automatically generated**(f)

Result:

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