**LAB CYCLE - 2**

**Experiment No :1**

**Date :17/10/2024**

**Aim :**   
  
Create a String from the given string where the first and last character are exchanged.

**Pseudocode :**

DISPLAY "Enter a string: "

GET str

DISPLAY str at last position, str from position 1 to 5, str at first position

**Source Code :**

str=input("Enter a string:")

print(str[-1]+str[1:5]+str[0])

**Output :**

Enter a string: python

nythop

**Result :**The program is successfully executed and the output is verified.

**Experiment No :2**

**Date: 17/10/2024**

**Aim :**

Get a String from the input string where all occurrences of the first Character are replaced with “$”, except first character.

**Pseudocode :**

DISPLAY "Enter the string: "

GET str1

SET char = first character of str1

SET str1 = replace all occurrences of char in str1 with '$'

DISPLAY char + str1 from position 1 to end

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| str.replace() | It replaces repeated string with another string | str.replace(old,new) |

**Source Code :**

str1=input("Enter the string:")

char=str1[0]

str1=str1.replace(char,'$')

print(char+str1[1:])

**Output :**

Enter the string: onion

oni$n

**Result :** The program is successfully executed and the output is verified.

**Experiment No :3**

**Date: 17/10/2024**

**Aim :**

Create a single string separated by space from 2 strings by swapping the characters at position 1.

**Pseudocode :**

DISPLAY "Enter first string: "

GET str1

DISPLAY "Enter second string: "

GET str2

SET n = length of str1

SET n2 = length of str2

SET str1sub = second character of str1

SET str2sub = second character of str2

DISPLAY str1 at position 0 + str2sub + str1 from position 2 to n, "", str2 at position 0 + str1sub + str2 from position 2 to n2

**Source Code :**

str1=input("Enter first string:")

str2=input("Enter second string:")

n=len(str1)

n2=len(str2)

str1sub=str1[1]

str2sub=str2[1]

print(str1[0]+str2sub+str1[2:n],"",str2[0]+str1sub+str2[2:n2])

**Output :**

Enter the first string1: hello

Enter the second string2: world

hollo werld

**Result :** The program is successfully executed and the output is verified.

**Experiment No :4**

**Date: 17/10/2024**

**Aim :**

Count the number of characters in a string.

**Pseudocode :**

DISPLAY "Enter the string: "

GET n

SET s = empty dictionary

FOR each character i in n DO

IF i is in s THEN

Increment s[i] by 1

ELSE

SET s[i] = 1

DISPLAY s

**Source Code :**

n=input("Enter the string:")

s={}

for i in n:

        if i in s:

                s[i]+=1

        else:

                s[i]=1

print(s)

**Output :**

Enter the string: hello

{‘h’:1, ‘e’:1, ‘l’:2, ‘o’,1}

**Result :** The program is successfully executed and the output is verified.

**Experiment No :5**

**Date: 17/10/2024**

**Aim :**

Add ‘ing’ at the end of the given string, if it ends with ‘ing’ the add ‘ly’

**Pseudocode :**

DISPLAY "Enter the string: "

GET str

IF last 3 characters of str are "ing" THEN

DISPLAY str + "ly"

ELSE

DISPLAY str + "ing"

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| str.endswith() | method is used to check if a string **ends** with a specified suffix. | str.endswith(“string”) |

**Source Code :**

str=input("Enter the string:")

if str[-3:]=="ing":

        print(str+"ly")

else:

        print(str+"ing")

**Output :**

Enter the string: live  
liveing

Enter the string: living  
livingly

**Result :** The program is successfully executed and the output is verified.

**Experiment No :6**

**Date: 17/10/2024**

**Aim :**   
  
Create and Store a list of first names. Count the occurrences of ‘a’ within the list.

**Pseudocode :**

DISPLAY "Enter the first names separated by comma: "

GET name

SET count\_a = count occurrences of 'a' in name converted to lowercase

DISPLAY "The letter 'a' appears ", count\_a, " times in the list of first names"

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| str.lower() | It is used to convert the string from uppercase to lowercase | name=ABC  name.lower()=abc |
| str.count() | method in Python is used to count the occurrences of a specific value in a sequence of string | sequence.count(value) |

**Source Code :**

name=input("Enter the first names separated by comma:")

count\_a=name.lower().count('a')

print(f"The letter 'a' appears {count\_a} times in the list of first names")

**Output :**

Enter the first names separated by comma:hrai,hari

The letter 'a' appears 2 times in the list of first names

**Result :** The Program is successfully executed and the output is verified.

**Experiment No :7**

**Date: 17/10/2024**

**Aim :** Write a program to read 2 lists color list 1 and color list 2 print the all the colors from the color list 1 and not contained in color list 2

**Pseudocode :**

DISPLAY "Enter colors separated by comma: "

GET l1

SPLIT l1 by ','

DISPLAY "Enter colors separated by comma: "

GET l2

SPLIT l2 by ','

FOR each color in l1 DO

IF color is NOT in l2 THEN

DISPLAY color

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| str.split() | method in Python is used to divide a string into a list of substrings based on a specified delimiter (separator). | str1=a,b,c  str1.split(‘,’) |

**Source Code :**

l1=input("Enter colors separated by comma:").split(',')

l2=input("Enter colors separated by comma:").split(',')

for color in l1:

        if color not in l2:

                print(color)

**Output :**

Enter colors separated by comma:red,blue,black

Enter colors separated by comma:blue,red,white

black

**Result :** The program is successfully executed and the output is verified.

**Experiment No :8**

**Date: 17/10/2024**

**Aim :**   
  
Create a list of colors from comma-separated color names entered by the user. Display first and last colors.

**Pseudocode :**

DISPLAY "Enter colors separated by comma: "

GET color

SPLIT color by ','

DISPLAY first element of color, "", last element of color

**Source Code :**

color=input("Enter colors separated by comma:").split(',')

print(color[0],"",color[-1])

**Output :**

Enter colors separated by comma:red,blue,black

red black

**Result :** The program is successfully executed and the output is verified.

**Experiment No :9**

**Date: 17/10/2024**

**Aim :**

Write a program to prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead.

**Pseudocode :**

DISPLAY "Enter integers separated by comma: "

GET inp

SPLIT inp by ','

SET res = empty list

FOR each num in inp DO

IF integer value of num > 100 THEN

APPEND "over" to res

ELSE

APPEND integer value of num to res

DISPLAY res

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| str.append() | method is used to add a single item to the end of a list. It modifies the original list in place and does not return a new list. | list.append(item) |

**Source Code :**

inp=input("Enter integers separated by comma:").split(',')

res=[]

for num in inp:

        if int(num)>100:

                res.append('over')

        else:

                res.append(int(num))

print(res)

**Output :**

Enter integers separated by comma:1,121,200,0

[1, 'over', 'over', 0]

**Result :** The program is successfully executed and the output is verified.

**Experiment No :10**

**Date: 17/10/2024**

**Aim :**

From a list of integers, create a list after removing even numbers.

**Pseudocode :**

DISPLAY "Enter integers separated by comma: "

GET inp

SPLIT inp by ','

SET res = empty list

FOR each num in inp DO

IF integer value of num is odd THEN

APPEND num to res

ELSE IF integer value of num is 0 THEN

APPEND num to res

DISPLAY "New list is: ", res

**Source Code :**

inp=input("Enter integers separated by comma:").split(',')

res=[ ]

for num in inp:

        if int(num)%2!=0:

                res.append(num)

        elif int(num)==0:

                res.append(num)

print("New list is:",res)

**Output :**

Enter integers separated by comma:1,2,3,4

New list is: ['1', '3']

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 11**

**Date: 24/10/2024**

**Aim :** Accept a list of words and return the length of the longest word.

**Pseudocode :**

DISPLAY "Enter words separated by space: "

GET wd

SPLIT wd by space

SET max = 0

FOR each word w in wd DO

IF length of w is greater than max THEN

SET max = length of w

DISPLAY "Length of longest word is ", max

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| str.len() | used to determine the length of an object. | len(object) |

**Source Code :**

wd=input("Enter words separated by space:").split(' ')

max=0

for w in wd:

        if len(w)>max:

                max=len(w)

print("Length of longest word is ",max)

**Output :**

Enter words separated by space:hello hellloooo

Length of longest word is 9

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 12**

**Date: 24/10/2024**

**Aim :** Write a program to prompt the user to enter two lists of integers and check

(a) Whether lists are of the same length.

(b) Whether the list sums to the same value.

(c) Whether any value occurs in both Lists

**Pseudocode :**

DISPLAY "Enter the number of Integers for List-1: "

GET n1

SET list1 = empty list

FOR i FROM 0 TO n1-1 DO

DISPLAY "Enter the Integers: "

GET num1

APPEND num1 to list1

DISPLAY "Enter the number of Integers for List-2: "

GET n2

SET list2 = empty list

FOR i FROM 0 TO n2-1 DO

DISPLAY "Enter the Integers: "

GET num2

APPEND num2 to list2

IF length of list1 is equal to length of list2 THEN

DISPLAY "Length of Both Lists are Same"

ELSE

DISPLAY "Length of Both Lists are Different"

IF sum of list1 is equal to sum of list2 THEN

DISPLAY "Sum of both Lists are Same"

ELSE

DISPLAY "Sum of both Lists are Different"

SET common\_val = intersection of list1 and list2

IF common\_val is not empty THEN

DISPLAY common\_val, "is the common value of both List"

ELSE

DISPLAY "Common value does not exist in both List"

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| sum() | used to calculate the total of all numeric values in an iterable (like a list, tuple, or set). | sum(iterable, start=0) |

**Source Code :**

n1=int(input("Enter the no:of Integers for List-1: "))

list1=[]

for i in range(n1):

num1=int(input("Enter the Integers: "))

list1.append(num1)

n2=int(input("Enter the no:of integers for List-2: "))

list2=[]

for i in range(n2):

num2=int(input("Enter the integers: "))

list2.append(num2)

if len(list1)==len(list2):

print("Length of Both Lists are Same")

else:

print("Length of Both Lists are Different")

if sum(list1)==sum(list2):

print("Sum of both Lists are Same")

else:

print("Sum of both Lists are Different")

common\_val=set(list1).intersection(list2)

if common\_val:

print(f"{common\_val} is the common value of both List")

else:

print("common value does not exist in both List")

**Output :**

Enter the no:of Integers for List-1: 2

Enter the Integers: 12

Enter the Integers: 23

Enter the no:of Integers for List-2: 2

Enter the Integers: 45

Enter the Integers: 56

Length of the Both Lists are Same

Sum of both Lists are Different

Common value does not exist in both List

**Result :** The program is successfully executed and the output is verified.

**Experiment No :13**

**Date: 24/10/2024**

**Aim :**

Write a Python program to count the occurrences of each word in a line of text.

Sample input : the quick brown fox jumps over the lazy dog

**Pseudocode :**

DISPLAY "Enter the line of text: "

GET inp

CONVERT inp to lowercase

SPLIT inp into words by spaces

SET word\_s = empty dictionary

FOR each word in words DO

IF word is in word\_s THEN

Increment word\_s[word] by 1

ELSE

SET word\_s[word] = 1

DISPLAY word\_s

**Source Code :**

inp=input("enter the line of text:".lower())

words=inp.split()

word\_s={}

for inp in words:

        if inp in word\_s:

            word\_s[inp]+=1

        else:

        word\_s[inp]=1

print(word\_s )

**Output :**

enter the line of text:hello hello hi

{'hello': 2, 'hi': 1}

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 14**

**Date: 24/10/2024**

**Aim :**

List comprehensions:

(a) Generate positive list of numbers from a given list of integers

(b) Square of N numbers

(c) Form a list of vowels selected from a given word

(d) Form a list ordinal value of each element of a word (Hint: use ord() to get

ordinal values)

**Pseudocode :**

SET ls1 = [-5, 8, 10, 8, -15, 18]

SET pls1 = list comprehension to get numbers from ls1 that are greater than 0

DISPLAY "+ve numbers", pls1

DISPLAY list comprehension to get squares of numbers from 1 to 5

SET wd = "hello"

SET vow = set comprehension to get vowels from wd that are in ['a', 'e', 'i', 'o', 'u']

DISPLAY vow

SET ordval = list comprehension to get the ordinal values of each character in wd

DISPLAY "Ordinal values:", ordval

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| ord() | returns the Unicode code point (integer representation) of a given character. | ord(char) |

**Source Code :**

ls1=[-5,8,10,8,-15,18]

pls1=[num for num in ls1 if num>0]

print("+ve numbers",pls1)

print([i\*i for i in range(1,6)])

wd="hello"

vow={word for word in wd if word in['a','e','i','o','u']}

print(vow)

ordval=[ord(ch) for ch in wd]

print("Ordinal values:",ordval)

**Output :**

+ve numbers [8, 10, 8, 18]

[1, 4, 9, 16, 25]

{'e', 'o'}

Ordinal values: [104, 101, 108, 108, 111]

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 15**

**Date: 24/10/2024**

**Aim :**

Sort dictionary in ascending and descending order.

**Pseudocode :**

SET my\_dict = {'banana': 3, 'apple': 5, 'orange': 2, 'kiwi': 4}

SET askey = sorted keys of my\_dict in ascending order

SET dskey = sorted keys of my\_dict in descending order

DISPLAY "Ascending sorting of keys:", askey

DISPLAY "Descending sorting of keys:", dskey

SET asv = sorted values of my\_dict in ascending order

SET dsv = sorted values of my\_dict in descending order

DISPLAY "Ascending sorting of values:", asv

DISPLAY "Descending sorting of values:", dsv

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| key() | This method returns a view object displaying all the keys in the dictionary. | dict.key() |
| value() | This method returns a view object displaying all the values in the dictionary. | dict.value() |
| sorted() | returns a sorted list of the specified iterable’s elements. | sorted(dict.keys) |

**Source Code :**

my\_dict={'banana':3,'apple':5,'orange':2,'kiwi':4}

askey=sorted(my\_dict.keys())

dskey=sorted(my\_dict.keys(),reverse=True)

print("Ascending sorting of keys:",askey)

print("Descending sorting of keys:",dskey)

asv=sorted(my\_dict.values())

dsv=sorted(my\_dict.values(),reverse=True)

print("Ascending sorting of values:",asv)

print("Descending sorting of values:",dsv)

**Output :**

Ascending sorting of keys: ['apple', 'banana', 'kiwi', 'orange']

Descending sorting of keys: ['orange', 'kiwi', 'banana', 'apple']

Ascending sorting of values: [2, 3, 4, 5]

Descending sorting of values: [5, 4, 3, 2]

**Result :** The Program is successfully executed and the Output is verified.

**Experiment No : 16**

**Date: 24/10/2024**

**Aim :**

Merge two dictionaries.

**Pseudocode :**

SET dict1 = {'banana': 3, 'apple': 5}

SET dict2 = {'orange': 2, 'kiwi': 4}

DISPLAY dict1

DISPLAY dict2

UPDATE dict1 with the contents of dict2

DISPLAY "Merged:", dict1

**Method :**

|  |  |  |
| --- | --- | --- |
| Functions | Description | Syntax |
| update() | method adds the key-value pairs from one dictionary to another. | dict1.update(dict2) |

**Source Code :**

dict1={'banana':3,'apple':5}

dict2={'orange':2,'kiwi':4}

print(dict1)

print(dict2)

dict1.update(dict2)

print("Merged:",dict1)

**Output :**

{'banana': 3, 'apple': 5}

{'orange': 2, 'kiwi': 4}

Merged: {'banana': 3, 'apple': 5, 'orange': 2, 'kiwi': 4}

**Result :** The Program is successfully executed and the Output is verified.