Experiment No:1 Lab Cycle:01 Date:25-09-2024

LEAP YEAR

AIM:

Write a python program to display future leap years from current year to a final entered by user.

ALGORITHM:

1. Start.
2. Input: Enter the final year (year).
3. Initialize: Create an empty list leap.
4. Iterate: Start a loop from 2024 to year (inclusive).
    a. If the year is divisible by 400, it is a leap year.
    b. Else if the year is divisible by 4 and not divisible by 100, it is a leap year.
    c. If any of the above conditions are true, add the year to the leap list.
5. Display result
6. Stop.

SOURCECODE:

```
year=int(input("Enter the final year"))
leap=[]
for i in range(2024,year+1):
        if i%400==0 or i%4==0 and i%100!=0:
                leap.append(i)
print("LEAP YEARS :",leap)
```

OUTPUT:

Enter the final year: 2032
LEAP YEARS: [2024, 2028,2032]

RESULT:

The program ran successfully and output is verified.

Experiment No:2.1 Lab Cycle:01 Date : 25-09-2024

POSITIVE LISTS OF NUMBERS

AIM:

Write a python program to generate positive list of numbers from a given list of numbers

ALGORITHM:

1. Start
2. Input: Provide a list of numbers (numbers).
3. Initialize: Create an empty list pos to store positive numbers.
4. Iterate: For each number i in the list numbers:
   a. If i is greater than 0, it is a positive number.
   b. Append i to the list pos.
5. Output: Display the list pos containing all positive numbers.
6. Stop..

SOURCE CODE:

```
list=[1,-2,3,-1,4,-6,9,-7,5]
pos=[]
for i in list:
    if i >0:
        pos.append(i)
print("Positive numbers : ",pos)
```

OUTPUT:

list = [1, -2, 3, -1, 4, -6, 9, -7, 5]

Positive numbers: [1, 3, 4, 9, 5]

RESULT:

The program ran successfully and output is verified.

Experiment No:2.2 Lab Cycle:01 Date:25-09-2024

SQUARE OF N NUMBERS

AIM:

Write a Python program to generate Square of N numbers.

ALGORITHM:

1. Input the number of elements n
2. Initialize an empty list to store the numbers.
3. Loop from 1 to n. For each iteration: Prompt the user to enter a number, Append the entered number to the list.
4. Compute the squares by using a loop to calculate the square of each number in the list and store the squared values in a new list.
5. Print the result.

SOURCE CODE:

```
n = int(input("Enter the number of elements to square: "))
numbers = []
for i in range(n):
    num = int(input("Enter number " + str(i + 1) + ": "))
    numbers.append(num)
squared_numbers = []
for x in numbers:
     squared_numbers.append(x * x)
print("Original numbers:", numbers)
print("Squares of the numbers:", squared_numbers)
```

OUTPUT:

```
Enter the number of elements to square: 3
Enter number 1: 2
Enter number 2: 5
Enter number 3: 3
Original numbers: [2, 5, 3]
Squares of the numbers: [4, 25,9]
```

RESULT:

The program ran successfully and output is verified.

Experiment No:2.3 Lab Cycle:01 Date:25-09-2024

LIST OF VOWELS IN A WORD

AIM:

Write a Python program to find the list of vowels in a word

ALGORITHM:

1. Start.
2. Input Word: Prompt the user to enter a word.
3. Find Vowels:
   Create an empty list list1.
   For each character in the word, if it is a vowel and not already in list1, add it to list1.
6. Output: Print the vowels stored in list1.
5. Stop.

SOURCE CODE:

```
element=input("Enter a word:")
vowels=['a','e','i','o','u']
list1=[]
for x in element:
    if(x in vowels and x not in list1):
        list1.append(x)
print("vowels present in the given word are: ",list1)
```

OUTPUT:

```
Enter a word: english
vowels present in the given word are: ['e', 'i']
```

RESULT:

The program ran successfully and output is verified.

Experiment No:2.4 Lab Cycle:01 Date:09-10-2024

ORDINAL VALUE

AIM:

Write a Python program to find the list of ordinal value in a word

ALGORITHM:

1. Start.
2. Input Word: Prompt the user to enter a word and store it in word.
3. Iterate through the Word:
   For each character i in the word,
   Find the ordinal value of the character using ord(i).
   Print the character and its ordinal value.
4. Stop.

SOURCE CODE:

```
word = input("Enter word : ")
for i in word:
    print("Ordinal value of ",i," is :",ord(i))
```

OUTPUT:

```
Enter word : hello
Ordinal value of h is : 104
Ordinal value of e is : 101
Ordinal value of l is : 108
Ordinal value of l is : 108
Ordinal value of o is : 111
```

RESULT:

The program ran successfully and output is verified.

Experiment No: 3 Lab Cycle:01 Date:09-10-2024

OCCURRENCE OF THE WORD

AIM:

Count the occurence of each word in a line of text.

ALGORITHM:

1. Start.
2. Input: Take a line of text from the user (str).
3. Initialize:
   An empty dictionary freq to store word occurrences.
4. Split the string: Divide the string into words using the split() function.
5. Iterate: For each word in the list of words:
   a. If the word is already in the dictionary freq, increment its value by 1.
   b. If the word is not in the dictionary, add it with a value of 1.
6. Output: Display each word and its count from the dictionary freq.
7. Stop.

SOURCE CODE:

```
str = input("Enter a string")
freq = {}

for char in str:
    if char in freq:
        freq[char] += 1
    else:
        freq[char] = 1

for char, count in freq.items():
    print(f"{char}: {count}")
```

OUTPUT:

Enter the string: hello hai how are you hello hai

Word occurrences:
hello: 2
hai: 2
how: 1
are: 1
you: 1

RESULT:

The program ran successfully and output is verified.

Experiment No:4 Lab Cycle:01 Date:09-10-2024

FILTERING INTEGERS

AIM:

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

ALGORITHM:

1. Start Start.
2. Input: Prompt the user to enter a list of integers separated by spaces.
3. Convert Input:
   Split the input string into a list of substrings using split().
   Convert each substring into an integer and store the result in a list number.
4. Process the List:
   Use a list comprehension to iterate through the list number.
   For each integer num in number:
   If num is greater than 100, replace it with the string 'over'.
   Otherwise, keep the integer as it is.
   Store the processed results in a new list result.
5. Output: Print the processed list result.
6. Stop.

SOURCE CODE:

```
user_input = input(“Enter a list of integers separated by spaces: ”)

number =[int(num) for num in user_input.split()]

result = [‘over’ if num>100 else num for num in number]

print(“Processed list : ”,result)
```

OUTPUT:

```
Enter a list of integers separated by spaces: 34 56 102 78 123

Processed list : [34, 56, 'over', 78, 'over']
```

RESULT:

The program ran successfully and output is verified.

Experiment No: 5 Lab Cycle:01 Date:09-10-2024

COUNT THE OCCURRANCE OF 'A'

AIM:

Store a list of first names. Count the occurences of 'a' within the list.

ALGORITHM:

1. Start.
2. Input: Ask the user to input words for the list, separated by spaces.
3. Initialize:
   Convert the user input into a list using the split() function.
   Set a variable out to 0 to store the total count of the letter 'a'.
4. Iterate: For each word i in the list:
   Convert the word to lowercase using the lower() method.
   Count the occurrences of the letter 'a' in the word using the count("a") method.
   Add the count to out.
5. Output: Display the total number of occurrences of 'a' in all the words.
6. Stop.

SOURCE CODE:

```
words = input("Enter the words separated by spaces: ").split()

out = 0

for word in words:
    out += word.lower().count("a")

print("The total number of 'a' in the words is:", out)
```

OUTPUT:

Enter the words separated by spaces: amal Arun aman arjun Anwar

The total number of 'a' in the words is: 8

RESULT:

The program ran successfully and output is verified.

Experiment No: 6 Lab Cycle:01 Date:30-10-2024

LIST OF INTEGERS

AIM:

Enter 2 lists of integers , Check:
(a) whether lists are of same length
(b) whether both lists sums to same value
(c) whether any value occur i both

ALGORITHM:

1. Input two list of integers
2. Check whether lists are of same length
3. Check whether both the list sums to same value
4. Check whether any value occur in both

SOURCE CODE:

```
l1 = list(map(int, input("Enter the elements of the first list separated by spaces: ").split()))
l2 = list(map(int, input("Enter the elements of the second list separated by spaces: ").split()))

if len(l1) == len(l2):
    print("Both lists have the same length.")
else:
    print("Different lengths.")

if sum(l1) == sum(l2):
    print("Both lists have the same sum.")
else:
    print("Different sums.")

common_elements = []
for i in l1:
    if i not in common_elements and i in l2:
        common_elements.append(i)

print("The common elements are:", common_elements)
```

OUTPUT:

```
Enter the elements of the first list separated by spaces: 1 2 3 4 5
Enter the elements of the second list separated by spaces: 7 8 9 10 4

Different lengths.
Different sums.
The common elements are: [4]
```

RESULT:

The program ran successfully and output is verified.

Experiment No:07 Lab Cycle:01 Date:30-10-2024

REPLACING FIRST CHARACTER WITH $

AIM:

Write a Python program to get a string from an input string where all occurrence of first character replaced with '$', except first character.

ALGORITHM:

1. Input a string, input_string
2. Extract the first character of input_string as first_char.
3. Initialize modified_string with first_char.
4. Iterate through the remaining characters of input_string:
   a)If the current character matches first_char (case-insensitive), append '$' to modified_string.
   b)Otherwise, append the current character as is.
5. Print modified_string.

SOURCE CODE:

```
input_string = input("Enter a string: ")
first_char = input_string[0]
modified_string = first_char
for char in input_string[1:]:
    if char.lower() == first_char.lower():
        modified_string += '$'
    else:
        modified_string += char
print("Modified string:", modified_string)
```

OUTPUT:

Enter a string: Onion

Modified string: Oni$n

RESULT:

The program ran successfully and output is verified.

Experiment No:8 Lab Cycle:01 Date:30-10-2024

EXCHANGING FIRST AND LAST

AIM:

Write a Python program to interchange the first and last character of a string.

ALGORITHM:

1.Start

2.Input string from the user.

3.Interchange first and last characters.

4.Print original and modified strings.

5.Stop

SOURCE CODE:

```
string = input("Enter the string: ")
print("Original String:",string)
string = string[-1]+string[1:-1]+string[0]
print("String after interchange:",string)
```

OUTPUT:

Enter the string: python

Original String: python

String after interchange: nythop

RESULT:

The program ran successfully and output is verified.

Experiment No:9 Lab Cycle:01 Date:30-10-2024

AREA OF CIRCLE

AIM:

Write a Python program to accept the radius from user and find area of circle.

ALGORITHM:

1. Start.
2. Input the radius of the circle (radius) from the user.
3. Calculate the area and store it in a variable (area).
   area=3.14*radius*radius
4. Display the area of the circle.
5. Stop.

SOURCE CODE:

```
radius=float(input("Enter the radius of a circle:"))
area=3.14*radius*radius
print("The area of circle is :",area)
```

OUTPUT:

```
Enter the radius of a circle:5
The area of circle is : 78.5
```

RESULT:

The program ran successfully and output is verified.

Experiment No:10 Lab Cycle:01 Date:06-11-2024

BIGGEST OF 3 NUMBERS

AIM:

Write a Python program to find biggest of 3 numbers entered.

ALGORITHM:

1. Start.
2. Input three numbers (x, y, and z).
3. Compare the numbers using the following conditions:
   a. If x >= y and x >= z, assign biggest = x.
   b. Else if y >= x and y >= z, assign biggest = y.
   c. Else, assign biggest = z.
4. Display the value of biggest.
5. Stop.

SOURCE CODE:

```
x=int(input("Enter first number:"))
y=int(input("Enter second number:"))
z=int(input("Enter third number:"))
if x>=y and x>=z:
    biggest=x
elif y>=x and y>=z:
    biggest=y
else:
    biggest=z

print("The biggest number is", biggest)
```

OUTPUT:

```
Enter first number:234
Enter second number:567
Enter third number:123
The biggest number is 567
```

RESULT:

The program ran successfully and output is verified.

Experiment No:11 Lab Cycle:01 Date:06-11-2024

FILE EXTENSION

AIM:

Accept a file name from user and print extension of that.

ALGORITHM:

1. Input the file name from the user
2. Split the file name using the separator(.)
3. Extract the last part of the split result as file extension
4. Print the file extension

SOURCE CODE:

```
file_name = input("Enter the file name=")
file_extension = file_name.split('.')[-1]
print("The extension of the file is =", file_extension)
```

OUTPUT:

```
Enter the file name: example.txt
The extension of the file is = txt
```

RESULT

The program ran successfully and output is verified.

Experiment No:12 Lab Cycle:01 Date:06-11-2024

DISPLAY FIRST AND LAST COLOR

AIM:

Write a python program to create a list of colors from comma-separated color names entered by user. Display first and last colors.

ALGORITHM:

1.Start.
2. Input the Number of Colors: Prompt the user to enter count.
3. Initialize List: Create an empty list clrs.
4. Input Colors:
   Use a for loop to input count colors from the user.
   Append each color to clrs.
5. Display Results:
   Print the first color (clrs[0]) and the last color (clrs[-1]).
6. Stop.

SOURCE CODE:

```
clrs = []
count = int(input("Enter the number of colors: "))

print("Enter the colors:")

 for x in range(count):
      color = input()
      clrs.append(color)

print("First Color:", clrs[0], "Last Color:", clrs[-1])
```

OUTPUT:

```
Enter the number of colors: 3
Enter the colors: Red White Blue
Colors: Red White Blue
First Color: Red
Last Color: Blue
```

RESULT

The program ran successfully and output is verified.

Experiment No:13 Lab Cycle:01 Date:06-11-2024

COMPUTE n+nn+nnn

AIM:

Write a python program to accept an integer n and compute n+nn+nnn

ALGORITHM:

1. Start.
2. Input n: Read integer n.
3. Compute nn and nnn:
   Set nn = n**2.
   Set nnn = n**3.
4. Calculate Result: Set res = int(n) + int(nn) + int(nnn).
5. Output: Print n + nn + nnn = res.
6. Stop.

SOURCE CODE:

```
n=int(input("ENTER THE VALUE OF 'n':"))

t1=n**2

t2=n**3

res =(int(n)+int(t1)+int(t2))

print(n+"+"+t1+"+"+t2"="+res)
```

OUTPUT:

```
ENTER THE VALUE OF 'n': 5

5+25+125=155
```

RESULT

The program ran successfully and output is verified.

Experiment No:14 Lab Cycle:01 Date:13-11-2024

PRINT COLORS FROM LIST

AIM:

Write a Python Program to print out all colors from color-list1 not contained in color-list2.

ALGORITHM:

1. Take space-separated input for List1 and store it in a set.
2. Take space-separated input for List2 and store it in a set.
3. Find the difference between List1 and List2.
4. Print the colors that are in List1 but not in List2.

SOURCE CODE:

```
list1 = set(input("Enter colors for List1 (separated by spaces): ").split())
list2 = set(input("Enter colors for List2 (separated by spaces): ").split())
diff = list1.difference(list2)
print("COLORS IN LIST1 NOT IN LIST2:", diff)
```

OUTPUT:

Enter the colors to LIST1: Red Blue Green

Enter the colors to LIST2: Red Green

COLORS IN LIST1 NOT IN LIST2: {'Blue'}

RESULT:

The program ran successfully and output is verified.

Experiment No:15 Lab Cycle:01 Date:13-11-2024

SWAPPING FIRST CHARACTERS

AIM:

Write a Python program to create single string separated with space from two strings by swapping the character at position 1.

ALGORITHM:

1. Input two strings, str1 and str2.
2. Swap their first characters:
   a)Replace the first character of str1 with the first character of str2.
   b)Replace the first character of str2 with the first character of str1.
3. Combine the modified strings with a space.
   a)Store the combined string in result.
4. Print the result.

SOURCE CODE:

```
str1 = input("Enter the first string: ")
str2 = input("Enter the second string: ")
swapped_str1=str2[0] + str1[1:]
swapped_str2=str1[0] + str2[1:]
result=swapped_str1+" "+swapped_str2
print("Swapped String: ",result)
```

OUTPUT:

```
Enter the first string: Hello
Enter the second string: World
Swapped String:  Wello Horld
```

RESULT:

The program ran successfully and output is verified.

Experiment No:16 Lab Cycle:01 Date:13-11-2024

SORTING DICTIONARY

AIM:

Write a Python Program to sort dictionary in ascending and descending order.

ALGORITHM:

1. Define a dictionary with key-value pairs.
2. Sort the dictionary in ascending order using sorted() and convert it to a dictionary using dict(). Store the result in asc.
3. Print the dictionary in ascending order.
4. Sort the dictionary in descending order using sorted() with reverse=True and convert it to a dictionary using dict(). Store the result in des.
5. Print the dictionary in descending order.

SOURCE CODE:

```
mydict = {'apple': 1, 'banana': 4, 'orange': 3, 'mango': 2}
asc = dict(sorted(mydict.items()))
print("Ascending order:", asc)
des = dict(sorted(mydict.items(), reverse=True))
print("Descending order:", des)
```

OUTPUT:

Ascending order: {'apple': 1, 'banana': 4, 'mango': 2, 'orange': 3}

Descending order: {'orange': 3, 'mango': 2, 'banana': 4, 'apple': 1}

RESULT:

The program ran successfully and output is verified.

Experiment No:17 Lab Cycle:01 Date:13-11-2024

MERGING 2 DICTIONARIES

AIM:

Write a Program to merge two dictionaries.

ALGORITHM:

1. Define the first dictionary(dict1)
2. Define the second dictionary(dict2)
3. Use the update() function to merge dict2 into dict1
4. Print the merged dictionary

SOURCE CODE:

```
dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
dict2 = {'f': 6, 'g': 7}
dict1.update(dict2)
print("Merged Dictionary =", dict1)
```

OUTPUT:

Merged Dictionary = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7}

RESULT :

The program ran successfully and output is verified.

Experiment No:18 Lab Cycle:01 Date:20-11-2024

GCD

AIM:

Write a Python program to Find gcd of 2 numbers.

ALGORITHM:

1.Read two integers,num1 and num2.
2.If num1 is less than num2, swap the values of num1 and num2.
3.Repeat the following steps until num2 becomes zero:
  1.Compute the remainder of num1 divided by num2 and assign it to num2.
  2.Assign the value of num2 (before this operation) to num1.
4.When num2 becomes zero, the value of num1 is the GCD.
5.Print the GCD.

SOURCE CODE:

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
if num1 < num2:
    num1, num2 = num2, num1
while num2 != 0:
    num1, num2 = num2, num1 % num2

print(f"The GCD is: {num1}")
```

OUTPUT:

```
Enter the first number: 123
Enter the second number: 321
The GCD is: 3
```

RESULT:

The program ran successfully and output is verified.

Experiment No:19 Lab Cycle:01 Date:20-11-2024

CREATE A LIST REMAINING EVEN NUMBER

AIM:

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

ALGORITHM:

1. input a list of integers
2. check a number is divisible by 2 or not.
3. If number is divisible by 2, i.e, even number
4. Remove even number from list
5. Print List

SOURCE CODE:

```
str_input = input("Enter the numbers, comma-separated: ")
listl [int(num) for num in str_input.split(',')]
print("List of numbers:", listl)
res=[ n for n in listl if n % 2 != 0]
print(res)
```

OUTPUT:

Enter the numbers, comma-separated:

1,2,3,4

[1,3]

RESULT:

The program ran successfully and output is verified.

Experiment No:20 Lab Cycle:02 Date:20-11-2024

FACTORIAL OF N

AIM:

Write a Program to find the factorial of a number.

ALGORITHM:

1. Define a function factorial(n) that:
2. a. Returns 1 if n is 0 or 1.
3. b. Calculates the factorial by multiplying numbers from 1 to n.
4. Take an integer n as input from the user.
5. Call the factorial() function with the input number.
6. Display the factorial result in the format "n! = result".

SOURCE CODE:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    fact = 1
    for i in range(1, n + 1):
        fact *= i
    return fact
n = int(input("Enter a number: "))
result = factorial(n)
print(f"{n}! = {result}")
```

OUTPUT:

```
Enter a number: 5
5! = 120
```

RESULT:

The program ran successfully and output is verified.

Experiment No:21 Lab Cycle:02 Date:20-11-2024

FIBONACCI SERIES

AIM:

Write a Python program to generate the Fibonacci series up to N numbers, where N is the limit provided by the user.

ALGORITHM:

1.Accept the input value limit from the user to determine the number of terms in the Fibonacci series.

2. Initialize a list fib with the first two numbers of the Fibonacci series, i.e., [0, 1].

3. Loop from 2 to limit - 1 (because the first two Fibonacci numbers are already present):

Calculate the next Fibonacci number by adding the last two numbers in the fib list.

Append the calculated number to the fib list.

5.Print the entire Fibonacci series stored in the fib list.

SOURCE CODE:

```
limit = int(input("Enter the limit: "))
fib = [0, 1]
for i in range(2, limit):
fib.append(fib[-1] + fib[-2])
print("Fibonacci series:", fib)
```

OUTPUT:

Enter the limit: 5

Fibonacci series: [0, 1, 1, 2, 3]

RESULT:

The program ran successfully and output is verified.

Experiment No:22 Lab Cycle:02 Date:27-11-2024

SUM OF ITEMS IN A LIST

AIM:

To find the sum of all items in a list where the list is provided by the user.

ALGORITHM:

1. Accept the list of numbers as input from the user.
2. Use the sum() function to calculate the sum of all items in the list.
3. Display the sum of the list.

SOURCE CODE:

```
list_input = input("Enter numbers separated by spaces: ").split()
list_numbers = [int(i) for i in list_input]
print(f"Sum of Lists: {sum(list_numbers)}")
```

OUTPUT:

Enter numbers separated by spaces: 1 2 3 4 5 6 7 8 9 10

SumofLists:55

RESULT:

The program ran successfully and output is verified

Experiment No:23 Lab Cycle:02 Date:27-11-2024

PERFECT SQUARE

AIM:

Write a Python program to generate a list of four-digit numbers within a given range where all the digits are even, and the number is a perfect square.

ALGORITHM:

1. Accept the starting and ending range values as input from the user.
2. Initialize an empty list out to store the numbers that meet the conditions.
3. Check if the starting and ending ranges are valid (both positive, and end greater than start)
4. Loop through the numbers in the given range:

   Check if the number is a perfect square.

   Check if all digits of the number are even.

   If both conditions are satisfied, append the number to the out list.

5. Print the list of numbers that satisfy the conditions. If the ranges are invalid, print a message indicating so.

SOURCE CODE:

```
import math
start = int(input("Enter the starting range: "))
end = int(input("Enter the ending range: "))
out = []
if start >= 1 and end >= 1 and end > start:
        for i in range(start, end + 1):
            if int(math.sqrt(i))**2 == iand all(int(digit) % 2 == 0 for digit in str(i)):
                out.append(i)
        print(f"Output: ", out)
else:
        print("Give Valid Ranges")
```

OUTPUT:

Enter the starting range: 1000

Enter the ending range: 9999

Output: [1600, 3600, 6400, 8400]

RESULT:

The program ran successfully and output is verified

Experiment No:24 Lab Cycle:02 Date:27-11-2024

NUMBER PYRAMID

AIM:

Write a Python program to display the given pyramid with step number accepted by user

eg: n=4

1

24

369

481216

ALGORITHM:

1. Accept the limit value n from the user to determine the number of rows in the pyramid.
2. Loop through each row (from 1 to n):

   For each row, initialize count = 1.

   Loop through the number of elements in the row (equal to the row number) and print the product of the current row number and the count value.

   Increment count after each element is printed.

3.Print the pyramid structure.

SOURCE CODE:

```
n = int(input("Enter the limit: "))

for i in range(1, n + 1):
    count = 1
    for j in range(0, i):
        print(i * count, end=" ")
        count += 1
    print("\n")
```

OUTPUT:

Enter the limit: 4

1

24

369

481216

RESULT:

The program ran successfully and output is verified

Experiment No:25 Lab Cycle:02 Date:27-11-2024

CHARACTER FREQUENCY

AIM:

Write a Python program to count the number of character (character frequency) in a string.

ALGORITHM:

1. Start
2. Store a string(str) and initialise an empty frequency
3. For each character in string

   check if char in freq:

   freq[char]+=1, increment the count of character

   otherwise

   Add a character and count to 1
4. Print the dictionary
5. Stop

SOURCE CODE:

```
str=input("Enter a string:")
freq={}
for char in str:
    if char in freq:
        freq[char]+=1
    else:
        freq[char]=1
print("Character frequencies:")
for char,count in freq.items():
    print(f"'{char}':{count}")
```

OUTPUT:

Enter a string:hello

Character frequencies:

'h':1

'e':1

'l':2

'o':1

RESULT:

The program ran successfully and output is verified

Experiment No:26 Lab Cycle:02 Date:04-12-2024

END OF A STRING

AIM:

Write a Python program to add "ing" at the end of a given string. If the string already ends with "ing", add "ly" instead.

ALGORITHM:

1.Accept a string as input from the user.

2.Check if the length of the string is greater than 3. If not, do nothing.

3.If the string ends with "ing", append "ly" to the string.

4. If the string does not end with "ing", append "ing" to the string.

5. Display the modified string.

SOURCE CODE:

```
str = input("Enter the string: ")
print("Input string:", str)

if len(str) > 3:
    if str[-3:] == "ing":
        str += "ly"
    else:
        str += "ing"

print("Formatted String:", str)
```

OUTPUT:

Enter the string: play

Input string: play

Formatted String: playing

Enter the string: swimming

Input string: swimming

Formatted String: swimmingly

RESULT:

The program ran successfully and output is verified

Experiment No:27 Lab Cycle:02 Date:04-12-2024

LONGEST WORD

AIM:

Write a Python program to accept a list of words and return length of longest word.

ALGORITHM:

1.Start

2.Input list of words

3.Find length and check if it is largest if true then store in l

4.Print the largest word length

5.Stop

SOURCE CODE:

```
a=[]
n=int(input("enter size of list:"))
print("enter list elements:")
for i in range(n):
    c=input()
    a.append(c)
print("List:",a)
l=0
for i in a:
    ln=len(i)
    if ln>l:
        l=ln
        w=i

print("longest word:" ,w ," and length is",l)
```

OUTPUT:

List:[‘one’,’two’,’three’,’four’]

Longest word: three and length is 5

RESULT:

The program ran successfully and output is verified

Experiment No:28 Lab Cycle:02 Date:04-12-2024

PATTERN

AIM:

Write a Python program to construct the following pattern using nested loops:

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

ALGORITHM:

1. Accept the value n (number of rows) from the user.
2. Use a nested loop to print the first half of the pattern (increasing number of stars).
3. Use another nested loop to print the second half of the pattern (decreasing number of stars).
4. In each iteration, print stars (*) according to the row number.

SOURCE CODE:

```
n = int(input("Enter the number of rows (n): "))
for i in range(1, n + 1):
    for j in range(i):
```

```
            print("*", end=" ")
        print()


    for i in range(n -1, 0, -1):
        for j in range(i):
            print("*", end=" ")
        print()
```

OUTPUT:

```
Enter the number of rows (n): 6

*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

RESULT:

The program ran successfully and output is verified

Experiment No:29 Lab Cycle:02 Date:04-12-2024

FACTORS

AIM:

Write a Python program to generate all factors of a number.

ALGORITHM:

1. Ask the user to enter a number.
2. Initialize an empty list called factors.
3. Use a loop to iterate from 1 to number.
4. If ‘number' is divisible by current number, add the current number to factors list.
5. After the loop, print the list.

SOURCE CODE:

```
number = int(input("Enter number: "))
factors = []
for i in range(1, number + 1):
    if number % i == 0:
        factors.append(i)

print(f"Factors of {number} are: {factors}")
```

OUTPUT:

```
Enter number: 20
Factors of 20 are: [1, 2, 3, 5, 6, 10, 15, 30]
```

RESULT:

The program ran successfully and output is verified

Experiment No:30 Lab Cycle:02 Date:11-12-2024

AREA USING LAMBDA

AIM:

Write a Python program to find area of square, rectangle and triangle using lambda functions.

ALGORITHM:

1. Start
2. Input the side length of the square, base and height of the triangle, and length and breadth of the rectangle from the user.
3. Calculate the area of the square using a lambda function.
4. Calculate the area of the triangle using a lambda function.
5. Calculate the area of the rectangle using a lambda function.
6. Print the areas of the square, triangle, and rectangle to the user.
7. Stop.

SOURCE CODE:

```
side = int(input("Enter the side length of the square: "))
height=int(input("Enter the height of the triangle: "))
base = int(input("Enter the base of the triangle: "))
length = int(input("Enter the length of the rectangle: "))
breadth = int(input("Enter the breadth of the rectangle: "))
square = lambda x: x**2
rectangle = lambda l, b: l * b
triangle = lambda ht, bs: 0.5 * ht * bs
print("Area of Square = ", square(side))
print("Area of Triangle = ", triangle(height, base))
print("Area of Rectangle = ", rectangle(length, breadth))
```

OUTPUT:

Enter the side length of the square:4

Enter the height of the triangle: 10

Enter the base of the triangle: 5

Enter the length of the rectangle: 12

Enter the breadth of the rectangle: 4

Area of Square = 16

Area of Triangle = 25.0

Area of Rectangle = 48

RESULT:

The program ran successfully and output is verified

Experiment No:31 Lab Cycle:03 Date:11-12-2024

BUILT IN PACKAGES

AIM:

Write a Python program to demonstrate the usage of built-in Python packages..

ALGORITHM:

1. Import the necessary built-in Python packages (math, random, datetime,).
2. Use functions from each package to perform basic operations.
3. Display the results to show how the built-in packages work.

SOURCE CODE:

```
number = 16
sqrt_value = math.sqrt(number)
print(f"Square root of {number} is: {sqrt_value}")


random_number = random.randint(1, 100)
print(f"A random number between 1 and 100 is: {random_number}")


current_datetime = datetime.datetime.now()
print(f"Current date and time: {current_datetime}"
```

OUTPUT:

```
Square root of 16 is: 4.0
A random number between 1 and 100 is: 57
Current date and time: 2025-01-05 10:45:12.123456
```

RESULT:

The program ran successfully and output is verified

Experiment No:32 Lab Cycle:03 Date:11-12-2024

PACKAGES AND SUB PACKAGES

AIM:

Write a program to create a package graphics with modules rectangle, circle and sub- package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements. (Include selective import of modules and import * statements)

ALGORITHM:

1.Start

2.Create a package graphics and inside this create init.py, rectangle.py, circle.py

3.Create a sub-package TDgraphics and inside this create init.py, cuboid.py, sphere.py

4.In Main.py import rectangle and circle using import statement.

5.Also use import * to import methods of cuboid.py, sphere.py from TDgraphics

6.Display the area and perimeter of rectangle, circle, cuboid and sphere

7.Stop

SOURCE CODE:

Main.py:

```
from graphics import rectangle,circle
from graphics.TDgraphics.cuboid import *
from graphics.TDgraphics.sphere import *


print("Area of rectangle:",rectangle.rarea(2,3))
print("Perimeter of rectangle:",rectangle.rperimeter(2,3))

print("Area of circle:",circle.carea(2))
print("Perimeter of circle:",circle.cperimeter(2))
```

```
print("Area of cuboid:",qarea(1,2,3))
print("Perimeter of cuboid:",qperimeter(1,2,3))
print("Area of sphere:",sarea(2))
print("Perimeter of sphere:",sperimeter(2))
```

graphics(package)

rectangle.py:

```
def rarea(l,b):
  return l*b
def rperimeter(l,b):
  return 2*(l+b)
```

circle.py:

```
from math import pi
def carea(r):
  return pi*r*r
def cperimeter(r):
  return 2*pi*r
```

init.py

graphics/TDgraphics(package)

cuboid.py:

```
def qarea(l,b,h):
  return 2*(l*b+b*h+l*h)
def qperimeter(l,b,h):
  return 4*(l+b+h)
```

sphere.py:

```
from math import pi
def sarea(r):
  return 4*pi*r*r
def sperimeter(r):
  return 2*pi*r
```

init.py

OUTPUT:

Area of rectangle: 6

Perimeter of rectangle: 10

Area of circle: 12.566370614359172

Perimeter of circle: 12.566370614359172

Area of cuboid: 22

Perimeter of cuboid: 24

Area of sphere: 50.26548245743669

Perimeter of sphere: 12.566370614359172

RESULT:

The program ran successfully and output is verified

Experiment No:33 Lab Cycle:04 Date:11-12-2024

COMPARE TWO RECTANGLES

AIM:

Write a Program to create a Rectangle class with attributes length and breadth and methods to find area and perimeter. Also, compare two Rectangle objects by their area.

ALGORITHM:

1. Define a Rectangle class.
2. Add attributes length and breadth to represent the dimensions of the rectangle.
3. Define the method area() to calculate and return the area of the rectangle (length * breadth).
4. Define the method perimeter() to calculate and return the perimeter of the rectangle (2 * (length + breadth)).
5. Create two Rectangle objects by taking user inputs for length and breadth.
6. Compare the areas of the two Rectangle objects and print which one has a larger area, or if they are equal.

SOURCE CODE:

```
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth
    def area(self):
        return self.length * self.breadth
    def perimeter(self):
        return 2 * (self.length + self.breadth)

length1 = int(input("Enter length of first rectangle: "))
breadth1 = int(input("Enter breadth of first rectangle: "))
```

```
rect1 = Rectangle(length1, breadth1)
length2 = int(input("Enter length of second rectangle: "))
breadth2 = int(input("Enter breadth of second rectangle: "))
rect2 = Rectangle(length2, breadth2)
area1 = rect1.area()
area2 = rect2.area()
print(f"Area of first rectangle: {area1}")
print(f"Areaof second rectangle: {area2}")
if area1 > area2:
    print("The first rectangle has a larger area.")
elif area1 < area2:
    print("The second rectangle has a larger area.")
else:
    print("Both rectangles have equal areas.")
```

OUTPUT:

```
Enter length of first rectangle: 5
Enter breadth of first rectangle: 4
Enter length of second rectangle: 6
Enter breadth of second rectangle: 3
Area of first rectangle: 20
Area of second rectangle: 18
The first rectangle has a larger area
```

RESULT:

The program ran successfully and output is verified

Experiment No:34 Lab Cycle:04 Date:18-12-2024

BANK ACCOUNT DETAILS

AIM:

Write a Python program to create a Bank Account class with attributes such as account number, account holder's name, account type, and balance, and implement methods to deposit, withdraw, and display account details.

ALGORITHM:

1.Define a class Bank with a constructor init to initialize account details: account number, name, type, and balance.

2.Define a method display to print the account details.

3.Define a method deposit to add funds to the account balance.

4.Define a method withdraw to subtract funds from the account balance, ensuring sufficient balance exists.

5.Create an object of the Bank class and use the methods to deposit, withdraw, and display account information.

SOURCE CODE:

```
class Bank:
    def __init__(self, no, name, type, bal):
        self.acc_no = no
        self.acc_name = name
        self.acc_type = type
        self.acc_bal = bal

    def display(self):
        print("Account number:", self.acc_no)
        print("Account holder name:", self.acc_name)
        print("Account type :",self.acc_type)
        print("Account balance is :",self.acc_bal,"\n")
```

```
    def deposit(self,val):
        if val>0:
            print("Current balance is",self.acc_bal)
            self.acc_bal+=val
            print(val,"deposited and current balance is :",self.acc_bal,"\n")
        else:
            print("enter a valid amount \n")


    def withdraw(self,val):
        if val>0:
            if self.acc_bal>=val:
                print("Current balance is",self.acc_bal)
                self.acc_bal-=val
                print(val,"withrawed and current balance is :",self.acc_bal,"\n")
            else:
                print("insuffient balance \n")
        else:
            print("enter a avlid amount \n")



    ob1=Bank(111,"ABCD","Savings",100000)
    ob1.display()
    ob1.withdraw(20000000)
    ob1.deposit(50000)
    ob1.display()
```

OUTPUT :

```
Account number: 111
Account holder name: ABCD
Account type: Savings
Account balance is: 100000

Insufficient balance

Current balance is 100000
50000 deposited and current balance is: 150000
 Account number: 111

Account holder name: ABCD
Account type: Savings
Account balance is: 150000
```

RESULT:

The program ran successfully and output is verified

Experiment No:35 Lab Cycle:04 Date:18-12-2024

OPERATOR OVERLOADING

AIM:

Write a Python program to create a class Rectangle with private attributes length and width.Overload '<' operator to compare the area of 2 rectangles

ALGORITHM:

1. Create a class named 'Rectangle
2. Initialize private attributes '__length','__width' and '__area"
3. Calculate and set the area during initialization.
4. Update the '__area' attribute
5. Compare the areas of two rectangle.
6. Take user input for length and width of two rectangles
7. Call 'cale area' method
8. Use the '<' operator to compare the areas of two rectangles.
9. Print which input is larger

SOURCE CODE:

```
class rectangle:
    __area=0
    __perimeter=0
    def__init__(self,length,width):
        self.__length=length
        self.__width=width
    def calc_area(self):
        self.__area=self.__length*self.__width
        print("Area is:" self.__area)
```

```
    def __lt__(self,second):
        if self.__area<second.__area:
            return True
        else:
            return False
length1= int(input("Enter length of the rectangle 1: "))
width1=int(input("Enter width of the rectangle 1: "))
length2=int(input("Enter length of the rectangle 2: "))
width2=int(input("Enter width of the rectangle 2: "))

obj1=rectangle(length 1, width1)
obj2=rectangle(length2, width2)
objl.calc_area()
obj2.calc_area()

if objl < obj2:
        print("Rectangle two is large")
else:
        print("Rectangle one is large or these are equal")
```

OUTPUT:

```
Enter length of rectangle1:6
Enter length of rectangle1:4
Enter length of rectangle2:5
Enter length of rectangle2:3

Area of 1 is 24
Area of 2 is 15
Rectangle one is larger
```

RESULT:

The program ran successfully and output is verified

Experiment No:36 Lab Cycle:04 Date:18-12-2024

SUM OF TIMES

AIM:

Write a Python program to create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.

ALGORITHM:

1. Create a class Time with attributes hour, minute and second.
2. Define a function add with attribute other which calculates totalhour, totalminute, totalsecond and returns Time (totalhour, totalminute, totalsecond).
3. Define a function display which displays the time.
4. Create two objects t1 and t2 with values.
5. Set t3=t1+t2
6. Display t1, t2 and t3.

SOURCE CODE:

```
Class Time:
        def __init__(self, hour=0, minute=0, second=0):
                self.__hour=hour
                self.__minute=minute
                self.__second=second

        def __add__(self, other):
                total_second = self.__second + other.__second
                total_minute = self.__minute + other.__minute + total_second // 60
```

```
            total_hour = self.__hour + other.__hour + total_minute // 60
            return Time(total_hour, total_minute % 60, total_second % 60)

        def display(self):
            return f"{self.__hour} {self.__minute} {self.__second}"

t1=Time(2,5,50)
t2=Time(1,20,30)
t3=t1 + t2
print("Time 1:", t1.display())
print("Time 2:", t2.display())
print("Sum of time:", t3.display())
```

OUTPUT:

```
Time 1: 2:5:50
Time 2: 1:20:30
Sum of time: 3:26:20
```

RESULT:

The program ran successfully and output is verified

Experiment No:37 Lab Cycle:04 Date:18-12-2024

METHOD OVERLOADING IN CLASS

AIM:

Write a Python program to create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no_of_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding

ALGORITHM:

1. Start

2. Define a super class Publisher with constructor to assign publisher name and a method display() to return the name.

3. Define derived class Book inherit from Publisher with attributes title and author, constructors to assign the values and a method display() to return the values

4. Define derived class Python inherit from Book with attributes price and no_of_pages, constructors to assign the values and a method display() to return the values.

5: Create an object for Python and invoke the method display() to return the result.

6: Stop

SOURCE CODE:

```
class Publisher:
    def __init__(self,name):
        self.name=name
    def display(self):
        print(f ” Publisher: {self.name}”)
class Book(Publisher):
    def __init__(self,name,title,author):
        super().__init__(name)
```

```
            self.title=title
            self.author=author
        def display(self):
            super().display()
            print(f "Title: {self.title}")
            print(f "Author: {self.author }")

class Python(Book):
    def __init__(self,name,title,author,price,no_of_pages):
            super().__init__(name,title,author)
            self.price=price
            self.no_of_pages= no_of_pages
    def display(self):
            super().display()
            print(f "Price: {self.price}")
            print(f "No of pages: {self.no_of_pages}")
publisher = Python("O' Reilly Media", "Think Python", "Allen B Downey",750,1250)
publisher.display()
```

OUTPUT:

Publisher : O' Reilly Media

Title : Think Python

Author : Allen B Downey

Price : ₹750

No of pages : 1250

RESULT:

The program ran successfully and output is verified

Experiment No:38 Lab Cycle:05 Date:01-01-2025

READ FILE LINE BY LINE

AIM:

Write a Python program to read a file line by line and store it into a list.

ALGORITHM:

1.Open the file

2.Initialize an empty list

3.Read each line

4.Strip whitespace

5.Append to list

6.Print the list

SOURCE CODE:

```
files=open('file1.txt','r')
list1=[]
for x in files:
    list1.append(x.strip())
print(list1)
```

OUTPUT:

```
['hi','how','are','you']
```

RESULT:

The program ran successfully and output is verified

Experiment No:39 Lab Cycle:05 Date:01-01-2025

COPY ODD LINES

AIM:

Write a Python program to copy odd lines of one file to other.

ALGORITHM:

1.Open the file1.txt for reading

2.Write every other line to file2.txt

3.Read and clean data from file2.txt

4.Print the cleaned lines

SOURCE CODE:

```
with open("file1.txt","r") as file:
    data=file.readlines()
with open("file2.txt","w") as outfile:
    for i in range(0, len(data),2):
        outfile.write(data[i])
with open("file2.txt","r") as outfile:
    out=[line.strip() for line in outfile.readlines()]
print(out)
```

OUTPUT:

```
['1.apple', '3.blueberry', '5.grapes']
```

RESULT:

The program ran successfully and output is verified

Experiment No:40 Lab Cycle:05 Date:01-01-2025

READ EACH ROW FROM CSV FILE

AIM:

Write a Python to read each row from a given CSV file and print a list of strings..

ALGORITHM:

1.Import the csv module to handle CSV files.

2.Open the given CSV file in read mode.

3.Use csv.reader() to read the file row by row.

4.Loop through each row of data and print it as a list of strings.

SOURCE CODE:

```
import csv
filename = "data.csv"
with open(filename, "r") as file:
    data = csv.reader(file)
    for i in data:
        print(i)
```

OUTPUT:

data.csv

```
Name, Age, Country
John, 25, USA
Jane, 30, Canada
['Name', ' Age', ' Country']
['John', ' 25', ' USA']
['Jane', ' 30', ' Canada']
```

RESULT:

The program ran successfully and output is verified

Experiment No:41 Lab Cycle:05 Date:01-01-2025

DISPLAY CONTENT OF COLUMN FROM CSV FILE

AIM:

Write a Python program to read a specific column (e.g., "Username") from a given CSV file and print its content.

ALGORITHM:

1.Import the csv module to handle CSV files.

2.Open the given CSV file in read mode. .

3.Use csv.DictReader() to read the file, which allows accessing columns by name.

4.Loop through each row of data and print the content of the specified column (e.g., 'Username').

SOURCE CODE:

```
import csv
filename = "data.csv"
with open(filename, "r") as file:
    data = csv.DictReader(file)
    for i in data:
        print(i['Username'])
```

OUTPUT:

data.csv

```
Username, Email, Age
john_doe, john@example.com, 28
jane_doe, jane@example.com, 32
john_doe
jane_doe
```

RESULT:

The program ran successfully and output is verified

Experiment No:42 Lab Cycle:05 Date:01-01-2025

DICTIONARY TO A CSV FILE

AIM:

Write a Python program to write a Python dictionary to a CSV file, then read and display the content of the CSV file.

ALGORITHM:

1. Define a list of dictionaries that contains data.
2. Open a CSV file in write mode.
3. Write the dictionary keys (column names) as the header in the CSV file.
4. Loop through the list of dictionaries and write the values to the CSV file.
5. Open the same CSV file in read mode.
6. Use csv.reader() to read the content of the file.
7. Display the content row by row.

SOURCE CODE:

```
import csv

data = [{'name': 'Keshu', 'role': 'student', 'college': 'cec'},
        {'name': 'Ramu', 'role': 'teacher', 'college': 'cec'},
        {'name': 'raju', 'role': 'lab', 'college': 'cec'}]

with open("dict.csv", 'w') as file:
    file.write(','.join(data[0].keys()))
    file.write('\n')
    for i in data:
        file.write(','.join(str(x) for x in i.values()))
        file.write('\n')
```

```
with open("dict.csv", 'r') as f:
    d = csv.reader(f)
    for i in d:
        print(i)
```

OUTPUT:

dict.csv

```
name,role,college
Keshu,student,cec
Ramu,teacher,cec
raju,lab tech,cec
```

```
['name', 'role', 'college']
['Keshu', 'student', 'cec']
['Ramu', 'teacher', 'cec']
['raju', 'labtech','cec']
```

RESULT:

The program ran successfully and output is verified