Experiment No.:1 Lab Cycle:1 Date: 20-09-2023

LEAP YEAR

AIM:

Display future leap years from current year to find years entered by uses

ALGORITHM:

1. Start.
2. Read year and future year.
3. Check whether the given year is divisible by 4 or the year is not divisible by 100 and year is divisible by 400, that it is a leap year otherwise it is not a leap year.
4. Print leap year.
5. Stop.

SOURCE CODE:

```
curr = int(input("Enter the current year:"))

fut = int(input("Enter the future year:"))

print("Leap Years:")

for curr in range(curr,fut+1):

if((curr%4==0) and curr%100!=0 or curr%400==0):

        print(curr)
```

OUTPUT:

```
Enter current year: 2022

Enter future year: 2050

Leap Years:

2024

2028

2032

2036

2040

2044

2048
```

RESULT:

Program ran successfully and output is verified.

Experiment No.:2 Lab Cycle:1 Date: 20-09-2023

POSITIVE LIST OF NUMBERS

AIM:

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

ALGORITHM:

1. Input list of numbers and print the list.
2. Check for each value greater than 0 using loop and if condition.
3. Print the positive numbers.
4. Stop.

SOURCE CODE:

```
num = [21, -4, 21, 100, -3]
print("List:",num)
print("Positive Numbers:")
for x in num:
    if(x>0):
        print(x)
```

OUTPUT:

List: [21, -4, 21, 100, -4]

Positive Numbers:

21

21

100

RESULT:

Program ran successfully and output is verified.

SQUARE OF N NUMBERS

AIM:

(b) Square of N numbers

ALGORITHM:

1. Input the n numbers.
2. Declare a list containing squares of the n numbers.
3. Print the list.

SOURCE CODE:

```
n = int(input("ENTER NUMBER:"))

sq = [i**2 for i in range (1,n+1)]

print("Square of first ",n," numbers:",sq)
```

OUTPUT:

Enter the number: 5

[1, 4, 9,16,25]

RESULT:

Program ran successfully and output is verified.

VOWELS IN A WORD

AIM:

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

ALGORITHM:

1. Input a word.
2. Add a list of vowels.
3. Check vowels in the word.
4. Print the vowels

SOURCE CODE:

```
word = input("ENTER THE WORD:")
vowels = ['a', 'e', 'i', 'o', 'u']
wordvowels = []
for x in word:
    if(x in vowels and x not in wordvowels):
        wordvowels.append(x)
print("Vowels in ", word," are:",wordvowels)
```

OUTPUT:

Enter the word: Kollam

Vowels in Kollam are: [‘o’, ‘a’]

RESULT:

Program ran successfully and output is verified.

ORDINAL OF VALUE ELEMENTS

AIM:

(d) List ordinal value of each element of a word.

ALGORITHM:

1. Input a word.
2. Find ordinal value using ord() function.
3. Print ordinal value of each element of a word.

SOURCE CODE:

```
word = input("Enter the word:")
print("Ordinal Values of ",word,"are:")
for x in word:
    print(x," : ",ord(x))
```

OUTPUT:

```
Enter the word: Hai
Ordinal Values of Hai are:
h: 104
a: 97
i: 105
```

RESULT:

Program ran successfully and output is verified.

OCCURRENCE OF THE WORD

AIM:

Count the occurrences of each word in a line of text

ALGORITHM:

1. Input a word.
2. Read the word and split the line to form a list of words.
3. Check if the words in the list are equal and if they are, increment the word count.
4. Print occurrence of each word in a line of text.

SOURCE CODE:

```
sentence = input("ENTER THE SENTENCE:")

words = sentence.split()

counts = dict()

for word in words:

        if word in counts:

                counts[word] += 1

        else:

                counts[word] = 1

print(counts)
```

OUTPUT:

ENTER THE SENTENCE: which is the fastest car in the world

{‘which’ : 1,‘is’ : 1, ‘the’ : 2, ‘fastest’ : 1, ‘car’ : 1, ‘in’ : 1, ‘world’ : 1 }

RESULT:

Program ran successfully and output is verified.

FILTERING INTEGERS

AIM:

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

ALGORITHM:

1. Input numbers.
2. Check number is greater than 100
3. Print number greater than 100 as 'over'.

SOURCE CODE:

```
num = []
n = int(input("ENTER THE NUMBER OF ELEMENTS:"))
print("ENTER THE LIST OF INTEGERS:")
for i in range(1,n+1):
    e = int(input())
    if(e>100):
        num.append("OVER")
    else:
        num.append(e)
print("ENTERED LIST:",num)
```

OUTPUT:

ENTER THE NUMBER OF ELEMENTS: 5

ENTER THE LIST OF INTEGERS: 32 999 123 1 5

ENTERED LIST: [32, 'over', 'over', 1, 5]

RESULT:

Program ran successfully and output is verified.

COUNT THE OCCURRENCE OF 'A'

AIM:

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

ALGORITHM:

1. Input number of names in a list.
2. Input the first names to add in a list.
3. Print occurrence of a.

SOURCE CODE:

```
names = []
acount = 0
n = int(input("ENTER THE NUMBER OF FIRST NAMES:"))
print("ENTER THE NAMES:")
for i in range(1,n+1):
    name = input()
    names.append(name)
for name in names:
    acount += name.lower().count('a')
print("Number of a : ",acount)
```

OUTPUT:

```
ENTER THE NUMBER OF FIRST NAMES: 3
ENTER THE NAMES: Kamal   Arun     Arjun
Number of a : 4
```

RESULT:

Program ran successfully and output is verified.

LIST OF INTEGERS

AIM:

Enter 2 lists of integers. Check:

(a) whether lists are of the same length

(b) whether both the lists' sums to same value

(c) whether any value occur in both

ALGORITHM:

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

SOURCE CODE:

```
list1 = []
list2 = []
m = int(input("ENTER THE LIMIT OF LIST1:"))
print("Enter the elements")
for i in range(0,m):
    value = int(input())
    list1.append(value)
n = int(input("ENTER THE LIT OF LIST2:"))
print("Enter the elements")
for i in range(0,n):
    value = int(input())
    list2.append(value)
print(list1,list2)
if len(list1) == len(list2):
    print(" Both List1 and List2 are of the same length")
else:
    print(" Both List1 and List2 are NOT of the same length")
if sum(list1) == sum(list2):
    print(" The Sum of both List1 and List2 are same")
else:
    print(" The Sum of both List1 and List2 are NOT same")
list3 = [each for each in list1 if each in list2]
print("Same Members are:",list3)
```

RESULT:

Program ran successfully and output is verified.

OUTPUT:

```
ENTER THE LIMIT OF LIST1: 3
ENTER ELEMENTS: 1  2  3
ENTER THE LIMIT OF LIST2: 3
ENTER ELEMENTS: 4  5  6
[1, 2, 3] [4, 5, 6]
Both List1 and List2 are of the same length
The Sum of both List1 and List2 are NOT same
```

STRING CHARACTER REPLACED WITH $

AIM:

Get a string from an input string where all occurrences of first character replaced with '$', except first character.

ALGORITHM:

1. Input a string.
2. replace all occurrence of first character with $.
3. Print the string.

SOURCE CODE:

```
string = input("ENTER THE STRING:")

first = string[0]

mod_str = first+string[1:].replace(first,'$')

print("modified string:"mod_str)
```

OUTPUT:

ENTER STRING: PINEAPPLE

modified string: PINEA$$LE

RESULT:

Program ran successfully and output is verified.

STRING WHERE FIRST AND LAST CHARACTERS EXCHANGED

AIM:

Create a string from given string where first and last characters exchanged.

ALGORITHM:

1. Input a string.
2. Store the first letter into the last position of the string.
3. Store the last letter of the string into the first position of the string.
4. print the string

SOURCE CODE:

```
string = input("ENTER STRING")

n = len(string)

first = string[0]

last = string[n-1]

mod_str = last + string[1:n-2] + first

print("Modified String:",mod_str)
```

OUTPUT:

ENTER STRING: DESAI

modified string: IESAD

RESULT:

Program ran successfully and output is verified.

Experiment No.:9 Lab Cycle:1 Date: 27-09-2023

AREA OF CIRCLE

AIM:

Accept the radius from user and find area of circle.

ALGORITHM:

1. Read radius of the circle.
2. Print the area of the circle (use pi=3.14).

SOURCE CODE:

```
r = int(input("Enter the radius of the circle: "))

print("Area of the circle:", 3.14 * r * r, "square units")
```

OUTPUT:

Enter the radius of the circle: 5

Area of the circle: 78.5 square unit

RESULT:

Program ran successfully and output is verified.

Experiment No.:10 Lab Cycle:1 Date: 27-09-2023

BIGGEST OF THREE NUMBERS

AIM:

Find biggest of 3 numbers entered

ALGORITHM:

1. Input three numbers.
2. Check which number is bigger.
3. Print the biggest number

SOURCE CODE:

```
print("Enter 3 Numbers:")
n1 = int(input())
n2 = int(input())
n3 = int(input())
if(n1 > n2 and n1 >n3):
    print(n1," is the biggest!!")
elif(n2 > n3):
    print(n2," is the biggest!!")
else:
    print(n3," is the biggest!!")
```

OUTPUT:

Enter 3 numbers: 5 12 6

12 is the biggest

RESULT:

Program ran successfully and output is verified.

Experiment No.:11 Lab Cycle:1 Date: 27-09-2023

EXTENSION OF A FILENAME

AIM:

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

ALGORITHM:

1. Input filename.
2. Function split is used.
3. Print the file extension

SOURCE CODE:

```
file = input("Enter the filename:")

file_list = file.split(".")

print("File Extension is : ",file_list[-1])
```

OUTPUT:

Enter the filename: past.jpg

File Extensionis: jpg

RESULT:

Program ran successfully and output is verified.

DISPLAY FIRST AND LAST COLOR

AIM:

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

ALGORITHM:

1. Input a list of colors.
2. Display first and last colors.
3. Print colors.

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[count-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:

Program ran successfully and output is verified.

COMPUTE n+nn+nnn

AIM:

Accept an integer n and compute n+nn+nnn

ALGORITHM:

1. Input an integer.
2. compute n+nn+nnn.
3. Print integer.

SOURCE CODE:

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

print("n + nn + nnn =",n + (n*n) + (n*n*n))
```

OUTPUT:

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

n + nn + nnn = 155
```

RESULT:

Program ran successfully and output is verified.

PRINT COLORS FROM COLOR LIST

AIM:

Print out all colors from color-list1 not contained in color-list2

ALGORITHM:

1. Input 2 list of colors.
2. Print out all colors from list1 not in list2.

SOURCE CODE:

```
clist1 = set()
clist2 = set()
n1 = int(input("Enter the number of colors in List1:"))
print("Enter the colors to LIST1:")
for x in range(n1):
    color = input()
    clist1.add(color)
n2 = int(input("Enter the number of colors in List2:"))
print("Enter the colors to LIST2:")
for x in range(n2):
    color = input()
    clist2.add(color)
diff = clist1.difference(clist2)
print("COLORS IN LIST1 NOT IN LIST2: ",diff)
```

OUTPUT:

```
Enter the number of colors in List1: 3
Enter the colors to LIST1: Red  Blue  Green
Enter the number of colors in List2: 2
Enter the colors to LIST2: Red  Green
COLORS IN LIST1 NOT IN LIST2: Blue
```

RESULT:

Program ran successfully and output is verified.

SWAPPING THE CHARACTER AT POINTER

AIM:

Create a single string separated with space from two strings by swapping the character at position 1

ALGORITHM:

1. Input 2 strings.
2. Swap two strings.
3. print swapped string

SOURCE CODE:

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

string2=input("Enter second string :")

stringn=string2[0]+string1[1:]+" "+string1[0]+string2[1:]

print (stringn)
```

OUTPUT:

```
Enter first string: Hello

Enter second string: World

Wello Horld
```

RESULT:

Program ran successfully and output is verified.

SORTING DICTIONARY

AIM:

Sort dictionary in ascending and descending order.

ALGORITHM:

1. declare list of keys and values
2. use sorted() function and operator.itemgetter to sort dict in ascending and descending order
3. print dictionary

SOURCE CODE:

```
import operator
mydict={}
while True:
    key=input("enter a key(or 'q' to quit):")
    if key=='q':
    break
    value=int(input("enter a value :"))
    mydict[key]=value
print('original dictionary: ',mydict)
sd=dict(sorted(mydict.items(),key=operator.itemgetter(1)))
print('Dictionary in ascending order by value : ',sd)
sd=dict(sorted(mydict.items(),key=operator.itemgetter(1),reverse=True))
print('Dictionary in descending order by value : ',sd)
```

OUTPUT:

```
enter a key(or 'q' to quit): 1
enter a value: 2
enter a key(or 'q' to quit): 3
enter a value: 4
enter a key(or 'q' to quit): q
Original dictionary : {‘1’: 2, ‘3’: 4 }
Dictionary in ascending order by value : {‘1’: 2, ‘3’: 4}
Dictionary in descending order by value : { ‘3’: 4, ‘1’: 2 }
```

RESULT:

Program ran successfully and output is verified.

MERGE TWO DICTIONARIES

AIM:

Merge two dictionaries

ALGORITHM:

1. input two dictionaries
2. merge two dictionaries
3. print dictionary

SOURCE CODE:

```
mydict1={}
print("Enter elements of first dic")
while True:
    key=input("enter a key(or 'q' to quit):")
    if key=='q':
        break
    value=int(input("enter a value :"))
    mydict1[key]=value
print("Enter elements of second dic")
mydict2={}
while True:
    key=input("enter a key(or 'q' to quit):")
    if key=='q':
        break
    value=int(input("enter a value :"))
    mydict2[key]=value
print(mydict1|mydict2)
```

OUTPUT:

Enter elements of first dic

enter a key(or ‘q' to quit):2

enter a value :21

enter a key(or 'q' to quit):3

enter a value :22

enter a key(or 'q' to quit):q

Enter elements of second dic:

enter a key(or 'q' to quit):4

enter a value :23

enter a key(or ‘q' to quit):5

enter a value :24

enter a key(or ‘q' to quit):q

{‘2’:21, '3’: 22, ‘4’:23, ‘5’:24}

Experiment No.:18 Lab Cycle:1 Date: 04-10-2023

GCD

AIM:

Find gcd of 2 numbers

ALGORITHM:

1. read gcd of two numbers
2. calculate gcd
3. print gcd

SOURCE CODE:

```
def findgcd(a,b):
    while b:
        a,b=b,a%b
    return a
num1=int(input("Enter the first number: "))
num2=int(input("Enter the second number: "))
gcd=findgcd(num1,num2)
print(f"The GCD of {num1} and {num2} is {gcd}.")
```

OUTPUT:

```
Enter the first number: 2
Enter the second number: 4
The GCD of 2 and 4 is 2.
```

RESULT:

Program ran successfully and output is verified.

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:

```
c=int(input("How many elements: "))
list1=[]
for i in range(c):
    list1.append(int(input("Enter the element: ")))
for i in list1:
    if(i%2==0):
        list1.remove(i)
print(list1)
```

OUTPUT:

```
How many elements: 3
Enter the element: 1
Enter the element: 3
Enter the element: 4
[1, 3]
```

RESULT:

Program ran successfully and output is verified.

Experiment No.:20 Lab Cycle:2 Date: 11-10-2023

FACTORIAL

AIM:

Program to find the factorial of a number.

ALGORITHM:

1. Read an integer n from user as input.
2. Initialize a variable f to 1.
3. For each integer i in the range from 1 to n, inclusive.
   a. Multiply f by i and update f with the result.
4. Print the result in the format “n! = f”.

SOURCE CODE:

```
n = int(input("ENTER NUMBER:"))

f = 1

for i in range(1, n + 1):

   f = f * i

print(n, "! =", f)
```

OUTPUT:

```
ENTER NUMBER: 16

16! = 20922789888000
```

RESULT:

Program ran successfully and output is verified.

FIBONACCI SERIES

AIM:

Generate Fibonacci series of N terms.

ALGORITHM:

1. Ask the user to input the number of terms, 'N'.
2. Check if 'N' less than or equal to 0. If it is, print a message indicating that it is invalid.
3. Initialize two variables, 'first' and 'second', to 0 and 1 respectively as these are the first two terms in the Fibonacci series.
4. Print a message indicating that you will be printing the first N numbers in the Fibonacci series.
5. Print the values of 'first' and 'second' separated by a comma and end the print statement with a comma.
6. Use a loop starting from 2 to N-1 to generate the remaining Fibonacci numbers. Within the loop:
   a. Calculate the next Fibonacci number, 'fib' by adding 'first' and 'second'.
   b. Update 'first' to 'second' and 'second to 'fib' to prepare for the next iteration.
   c. Check if 'fib' is greater than N and if it is, break out of the loop.
   d. Print fib separated with a comma.
7. The loop will generate and print the first N terms of the Fibonacci series.

SOURCE CODE:

```
n = int(input("ENTER NUMBER OF N TERMS:"))
if n <= 0:
    print("Fibonacci series up to", n, "is not defined.")
else:
    first = 0
    second = 1
    print("The first", n, "numbers in the Fibonacci Series =")
    print(first, ",", second, end=", ")
    for i in range(2, n):
        fib = first + second
        first = second
        second = fib
        if fib > n:
            break
```

```
        if i == n - 1:
            print(fib, end=" ")
        else:
            print(fib, end=", ")
```

OUTPUT:

ENTER NUMBER OF N TERMS: 12

The first 12 numbers in the Fibonacci Series = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

RESULT:

Program ran successfully and output is verified.

SUM OF LIST

AIM:

Find the sum of all items in a list.

ALGORITHM:

1. Ask the user to input the number of values (n).
2. Initialize an empty list called num_list to store the items.
3. Use a loop to read n values from the user and append them to num_list.
4. Calculate the total sum of the elements in num_list ad store it in the total variable.
5. Print the total sum with a descriptive message.

SOURCE CODE:

```
num_list = []

n = int(input("Enter the number of values in list :"))

print("Enter the elements to the list:")

for i in range(n):

    val = int(input())

    num_list.append(val)

total = 0

for item in num_list:

    total = total + item

print("The sum of all items in the list is:", total)
```

OUTPUT:

```
Enter the number of values in the list: 3

Enter the elements for the list:

1

4

2

The sum of all items in the list is: 7
```

RESULT:

Program ran successfully and output is verified.

PERFECT SQUARE

AIM:

Generate a list of four-digit number in a given range with all their digits even and the number is a perfect square.

ALGORITHM:

1. Ask the user to input the number range (start and end four numbers).
2. Check whether the input ranges are valid (start is less than or equal to end, and both are four-digit numbers). If not, print an error message.
3. If the input ranges are valid, initialize an empty list called 'result' to store the four-digit even perfect square numbers.
4. Use a loop to iterate through all numbers in the given range(inclusive).
5. For each number in the range, check if all digits are even by converting the number to be a string and checking the evenness of each digit.
6. If all digits are even, calculate the square root of the number.
7. Check if the square of the square root equals the number, indicating that it is a perfect square.
8. If it is a perfect square, add it to the 'result' list.
9. After the loop, print the 'result' list containing four digit even perfect square numbers in the given range.

SOURCE CODE:

```
result = []

start = int(input("Enter the starting range (four-digit number): "))
end = int(input("Enter the ending range (four-digit number): "))

if start < 1000 or end > 9999 or start > end:
    print("Invalid range. Please enter a valid four-digit range.")
else:
    for num in range(start, end + 1):
        if num % 2 == 0:
            root = int(num ** 0.5)  # taking the square root
            if root * root == num:  # checking if it is a perfect square
                result.append(num)  # adding it to the list
```

```
print("Four-digit even perfect square numbers in the given range:")
print(result)
```

OUTPUT:

```
Enter the starting range (four-digit number): 5000
Enter the ending range (four-digit number): 5210
Four-digit even perfect square numbers in the given range:
[5184]
```

RESULT:

Program ran successfully and output is verified.

Experiment No.:24 Lab Cycle:2 Date: 18-10-2023

PYRAMID

AIM:

Display the given pyramid with step number accepted from user.

Eg. N=4

1

2 4

3 6 9

4 8 12 16

ALGORITHM:

1. Ask the user to input the number of steps, ‘step'.
2. Use nested loops to iterate through rows and columns:
    a. The outer loop (variable 'i') iterates from 1 to step, representing rows.
    b. The inner loop (variable 'j') iterates from 1 to ‘i’, representing rows.
3. Inside the inner loop, calculate and print the product of "i" and "j", separated by a space.
4. After spent printing all columns for a row, print a newline character to move to the next row.

SOURCE CODE:

```
step = int(input("Enter number of steps:"))
for i in range(1,step+1):
    for j in range(1,i+1):
        print(i*j,end=" ")
    print("\n")
```

OUTPUT:

```
Enter number of steps: 4
1
2 4
3 6 9
4 8 12 16
```

RESULT:

Program ran successfully and output is verified.

Experiment No.:25 Lab Cycle:2 Date: 18-10-2023

CHARACTER FREQUENCY

AIM:

Count the number of characters (character frequency) in a string.

ALGORITHM:

1. Ask the user to input a string, ‘input_string’.
2. Create on empty dictionary called "char_frequency" to store character frequency (key - character, value - count).
3. Iterate through each character in 'input_string’.
4. For each character:
    a) check if it is already a key is the char_frequency dictionary.
    b) If it is, increment the count of that character.
    c) If not, add it as a key in the dictionary with initial count of 1.
5. After counting all characters, print the character frequencies.

SOURCE CODE:

```
input_string = input("Enter a string: ")

char_frequency = {}

for char in input_string:

    char_frequency[char] = char_frequency.get(char, 0) + 1

print(f"Character frequencies in '{input_string}':")

for char, count in char_frequency.items():

    print(f"{char} occurs {count} times.")
```

OUTPUT:

```
Enter a string: clever

Character frequencies in 'clever':

c occurs 1 times.

l occurs 1 times.

e occurs 2 times.

v occurs 1 times.

r occurs 1 times.
```

RESULT:

Program ran successfully and output is verified.

END OF A STRING

AIM:

Add ‘ing' at the end of a given string. If it already ends with ‘ing’, then add ‘ly’.

ALGORITHM:

1. Ask the user to input a string, ‘input_string’.
2. Check if ‘input-string' ends with the substring 'ing'.
3. If it does, append ‘ly’ to the end of the string and save it as ‘modified_string’.
4. If it doesn't, append ‘ing’ to the end of the ‘input_string' to create ‘modified_string’.
5. Print the ‘modified_string’.

SOURCE CODE:

```
input_string = input("Enter a string: ")

if input_string.endswith('ing'):

        modified_string = input_string + 'ly'

else:

        modified_string = input_string + 'ing'

print("Modified string:", modified_string)
```

OUTPUT:

Enter a string: catch

Modified string: catching

RESULT:

Program ran successfully and output is verified.

Experiment No.:27 Lab Cycle:2 Date: 25-10-2023

LONGEST WORD

AIM:

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

ALGORITHM:

1. Ask the user to enter a list of words (separated with commas). and save it to word_list.
2. Initialize a variable ‘long_len' to 0(used to denote the length of the longest two word).
3. Iterate through each word in the word_list.
4. remove unwanted spaces from each word by using 'strip()' method.
5. calculate each word length and save it to curr_len.
6. Now compare the curr_len with long_len:
    a) if curr_len > long_len, then long_len = curr_len and store long_word = word.
7. print the longest word and its length.

SOURCE CODE:

```
word_list = input("Enter a list of words (comma-separated): ").split(',')

long_len = 0

for word in word_list:

   word = word.strip()

   curr_len = len(word)

   if curr_len > long_len:

      long_len = curr_len

      long_word = word

print(f"Longest word: '{long_word}' with length {long_len}")
```

OUTPUT:

```
Enter a list of words (comma-separated): joy, philip, jacob, jerry

Longest word: 'philip' with length 6
```

RESULT:

Program ran successfully and output is verified.

Experiment No.:28 Lab Cycle:2 Date: 25-10-2023

PATTERN

AIM:

Construct following pattern using nested loop:

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

ALGORITHM:

1. Initialize step as 5.
2. Use a loop to iterate from 1 to 'step' to represent number of rows in first half of the pyramid.
3. For each row, use a nested loop to iterate from 1 to the current row index.
4. Print '*' for each internal iteration with a whitespace.
5. Print ‘\n’ for moving to next line after each row.
6. Now use a loop to iterate from the second half of the pyramid, from 'step-1' to 1.
7. Repeat steps 3 to 5.

SOURCE CODE:

```
step = 5

# Print the upper part of the pyramid

for i in range(1, step + 1):

    for j in range(1, i + 1):

        print("*", end=" ")

    print()

# Print the lower part of the pyramid

for i in range(step - 1, 0, -1):

    for j in range(1, i + 1):

        print("*", end=" ")

    print()
```

OUTPUT:

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

RESULT:

Program ran successfully and output is verified.

Experiment No.:29 Lab Cycle:2 Date: 25-10-2023

FACTORS

AIM:

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 a 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 a number: 30

Factors of 30 are: [1, 2, 3, 5, 6, 10, 15, 30]

RESULT:

Program ran successfully and output is verified.

AREA USING LAMBDA

AIM:

Write lambda functions to find area of square, rectangle and triangle.

ALGORITHM:

1. Define three lambda functions for calculating the areas of square, rectangle, and triangle with their respective arguments.
2. Read the inputs of each function as save them in variables:
    a) ‘sqs' for side of the square
    b) 'ht' and 'bs' for height for and base of the triangle
    c) 'l' and 'b' for the length and breadth of the rectangle.
3. Pass the values to the respective lambda functions and print them after execution.

SOURCE CODE:

```
# Lambda function to calculate the area of a square

calculate_square_area = lambda side_length: side_length ** 2

# Lambda function to calculate the area of a rectangle

calculate_rectangle_area = lambda length, width: length * width

# Lambda function to calculate the area of a triangle

calculate_triangle_area = lambda base, height: 0.5 * base * height


sqs = int(input("Enter the side of the square: "))

print("Enter the base and height of the triangle:")

ht = int(input("Height: "))

bs = int(input("Base: "))


print("Enter the length and breadth of the rectangle:")

l = int(input("Length: "))

b = int(input("Breadth: "))


square_area = calculate_square_area(sqs)

print("Area of the square:", square_area)


triangle_area = calculate_triangle_area(bs, ht)

print("Area of the triangle:", triangle_area)
```

```
rectangle_area = calculate_rectangle_area(l, b)
print("Area of the rectangle:", rectangle_area)
```

OUTPUT:

```
Enter the side of the square: 4

Enter the base and height of the triangle:
Height: 5
Base: 4

Enter the length and breadth of the rectangle:
Length: 7
Breadth: 3

Area of the square: 16
Area of the triangle: 10
Area of the rectangle: 21
```

RESULT:

Program ran successfully and output is verified.

Experiment No.:31 Lab Cycle:3 Date: 01-11-2023

BUILT-IN PACKAGES

AIM:

Work with built-in packages.

ALGORITHM:

1. Import required Python modules: random, datetime, and math.
2. Generate a random integer between 1 and 100 using random.randint.
    - Print the generated random number.
3. Get the current date and time using datetime.datetime.now().
    - Print the current date and time.
4. Prompt the user to enter a number.
    - Calculate the square root of the entered number using math.sqrt.
    - Print the calculated square root.

SOURCE CODE:

```
import math

import random

import datetime

random_number = random.randint(1, 100)

print(f"Random number between 1 and 100: {random_number}")

current_datetime = datetime.datetime.now()

print(f"Current date and time: {current_datetime}")

number = int(input("Enter a number:"))

square_root = math.sqrt(number)

print(f"The square root of {number} is: {square_root}")
```

OUTPUT:

```
Enter a number: 25

Random number between 1 and 100: 7

Current date and time: 2023-11-28 01:44:00

The square root of  25 is: 5.0
```

RESULT:

Program ran successfully and output is verified.

Experiment No.:32 Lab Cycle:3 Date: 15-11-2023

PACKAGES AND SUB PACKAGES

AIM:

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. Create a folder 'graphics' containing python files: circle.py and rectangle.py and define methods to find area and perimeter circle and rectangle in them respectively
2. Inside 'graphics' folder, create another subfolder named '_3D_graphics' containing python files: sphere.py and cuboid.py and define methods to find area and perimeter sphere and cuboid in them respectively.
3. Both 'graphics' and '_3D_graphics should contain an empty python file named '_init_'.
4. Now create 'main.py' which imports all the packages and read the values for dimensions.
5. Call the methods and print result.

SOURCE CODE:

main.py

```
from graphics.rectangle import area as rect_area, perimeter as rect_perimeter

from graphics.circle import area as circle_area, perimeter as circle_perimeter

from graphics._3D_graphics.cuboid import surface_area as cuboid_surface_area, volume as cuboid_volume

from graphics._3D_graphics.sphere import surface_area as sphere_surface_area, volume as sphere_volume


print("Enter the length and breadth of the rectangle:")

l = int(input("Length: "))

b = int(input("Breadth: "))

print("Rectangle Area:", rect_area(l, b))

print("Rectangle Perimeter:", rect_perimeter(l, b))

print("\n\n")

print("Enter the radius of the circle:")

radius = int(input("Radius: "))

print("Circle Area:", circle_area(radius))

print("Circle Perimeter:", circle_perimeter(radius))
```

```
print("\n\n")
print("Enter the dimensions of the cuboid:")
length = float(input("Length: "))
width = float(input("Width: "))
height = float(input("Height: "))
print("Cuboid Surface Area:", cuboid_surface_area(length, width, height))
print("Cuboid Volume:", cuboid_volume(length, width, height))
print("\n\n")
print("Enter the radius of the sphere:")
radius = float(input("Radius: "))
print("Sphere Surface Area:", sphere_surface_area(radius))
print("Sphere Volume:", sphere_volume(radius))
```

graphics package

circle.py

```
import math
def area(radius):
    return math.pi * radius ** 2
def perimeter(radius):
    return 2 * math.pi * radius
```

rectangle.py

```
def area(length, width):
    return length * width
def perimeter(length, width):
    return 2 * (length + width)
```

_3D_graphics package

cuboid.py

```
def surface_area(length, width, height):
    return 2 * (length * width + length * height + width * height)
```

```
def volume(length, width, height):
    return length * width * height
```

sphere.py

```
import math
def surface_area(radius):
    return 4 * math.pi * radius ** 2
def volume(radius):
    return (4/3) * math.pi * radius ** 3
```

OUTPUT:

Enter the length and breadth of the rectangle:

Length: 7

Breadth: 3

Rectangle Area: 21

Rectangle Perimeter: 20

Enter the radius of the circle:

Radius: 5

Circle Area: 78.5

Circle Perimeter: 31.4

Enter the dimensions of the cuboid:

Length: 5

Width: 4

Height: 3

Cuboid Surface Area: 94.0

Cuboid Volume: 60.0

Enter the radius of the sphere:

Radius: 7

Sphere Surface Area: 615.44

Sphere Volume: 1,436.026666666

RESULT:

Program ran successfully and output is verified.