MASTER OF COMPUTER APPLICATIONS

PRACTICAL RECORD WORK
ON
20MCA131 – PROGRAMMING LAB

Submitted by

NAME : RAHBAR ZAHID S

Reg. No. : VDA24MCA-2039

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING VATAKARA

(CAPE – GOVT. OF KERALA)

JANUARY 2025

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING VATAKARA

CAPE(– GOVT. OF KERALA)

CERTIFICATE

Certified that this is the bonafide record work on the practical course **20MCA131 Programming Lab** done and prepared Mr. **RAHBAR ZAHID.S (Reg.No VDA24MCA-2039**), 1st Semester MCA(2024-26 Batch) student of Department of Computer Applications at College of Engineering Vatakara, in the partial fulfilment of the award of MCA Degree of APJ Abdul Kalam Technological University (KTU).

Date :

Faculty-in-Charge **Head of the Department**

(Office Seal)

Internal Examiners: **External Examiners:**

INDEX

Experiment No: 1 <u>CO1</u>

Aim:

Display future leap years from current year to a final year entered by user.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the current & final year from the user.
Step 3: Print "The future Leap years are:"
Step 4:
- For each year x in the range from c to f :
- Check if (x % 4 == 0 and x % 100 != 0) or (x % 400 == 0).
- If the condition is true, print the year x.
Step 5: End
```

Source Code:

```
c=int(input("Enter the current year:"))
f=int(input("Enter the final year:"))
print("The future Leap years are:")
for x in range(c,f+1):
        if(x%4==0 and x%100!=0) or (x%400==0):
                print(x)
```

Output:

```
Enter the current year:2024
Enter the final year:2040
The future Leap years are:
2024
2028
2032
2036
2040
```

Experiment No: 2

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) List ordinal value of each element of a word. (Hint: use ord() to get ordinal values)

Algorithm/Pseudocode:

```
(a) Step 1: Start
Step 2: Create a list num_list with elements [-5, 10, 30, -45, -3, 33, -12, 24].& print num_list.
Step 3:
        - Create a new list newlist containing elements from num_list that are greater    than 0.
        - Print the newlist.
Step 4: End
(b) Step 1: Get the number of elements from the user.
Step 2:
        - Create an empty list num_list.
        - Use a loop to input n numbers from the user and append them to num_list.
Step 3: Print the num_list.
Step 4:
        - Create a new list sqr_lst using a list comprehension.
        - For each element x in num_list, include x^2 in sqr_lst.
Step 5: Print the sqr_lst.
Step 6: End

(c) Step 1: Start
Step 2: Get a word from user.
Step 3:
        - Create a list vowels containing lowercase vowels present in the input word.
        - Print vowels.
Step 4: End
```

```
(d) Step 1: Start
Step 2: Get a word from the user.
Step 3:
        - Create a list ord containing the ordinal values of characters in the input word.
        - Print "The ordinal value is:" followed by ord.
Step 4: End
```

Source Code:

(a):

```
num_list=[-5,10,30,-45,-3,33,-12,24]
print(num_list)
newlist=[x for x in num_list if x>0]
print("The positive numbers are:",newlist)
```

(b):

```
n=int(input("Enter the number of elements:"))
num_list=[]
for i in range(n):
a=int(input("Enter the numbers:"))
num_list.append(a)
print("Entered list:",num_list)
sqr_lst=[x**2 for x in num_list]
print("Squared list:",sqr_lst)
```

(c):

```
word=input("Enter a word:")
vowels=[x for x in word.lower() if x in['a','e','i','o','u']]
print("The vowels in the words are:",vowels)
```

(d):

```
w=input("Enter a word:")
ord=[ord(x) for x in w]
print("The ordinal value is:",ord)
```

Output:

```
[-5, 10, 30, -45, -3, 33, -12, 24]
The positive numbers are: [10, 30, 33, 24]
Enter the number of elements:5
Enter the numbers:2
Enter the numbers:4
Enter the numbers:6
Enter the numbers:8
Enter the numbers:10
Entered list: [2, 4, 6, 8, 10]
Squared list: [4, 16, 36, 64, 100]
Enter a word: algorithm
The vowels in the words are: ['a', 'o', 'i']
Enter a word:python
The ordinal value is: [112, 121, 116, 104, 111, 110]
```

Experiment No: 3

Aim:

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a line of text from user and store it in the variable s.
Step 3:
        - Initialize an empty dictionary count to store word frequencies.
        - Split the input line of text into words and store them in the list words.
        - For each word x in words:
        - If x is already a key in count, increment its value by 1.
        - If x is not a key in count, add it to count with a value of 1.
Step 4: Print the dictionary count, which represents the frequencies of words in the input line
of text.
Step 5: End
```

Source Code:

```python
s=str(input("Enter the line of text:"))
count=dict()
words=s.split()
for x in words:
        if x in count:
                count[x]+=1
        else:
                count[x]=1
print(count)
```

Output:

```
Enter the line of text: This is a simple example program for word count , This program
will count the occurrences of each word
{'This': 2, 'is': 1, 'a': 1, 'simple': 1, 'example': 1, 'program': 2, 'for': 1, 'word': 2, 'count': 2, ',
': 1, 'will': 1, 'the': 1, 'occurrences': 1, 'of': 1, 'each': 1}
```

Experiment No: 4

Aim:

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the number of elements from the user.
Step 3:
        - Initialize an empty list num_list.
        - For each element x in the range from 1 to n:
            - Prompt the user to enter an element x.
            - If x is less than 100, append it to num_list, else append 'over' to num_list.
Step 4: Print the list num_list, which contains elements or 'over' based on the conditions.
Step 5: End
```

Source Code:

```
n=int(input("Enter the number of elements in the list:"))
num_list=[]
for x in range(n):
        x=int(input("Enter the elments:"))
        if x<100:
                num_list.append(x)
        else:
                num_list.append('over')
print(num_list)
```

Output:

```
Enter the number of elements in the list:4
Enter the elments:100
Enter the elments:50
Enter the elments:40
Enter the elments:150
['over', 50, 40, 'over']
```

Experiment No: 5

Aim:

Store a list of first names. Count the occurrences of ‘a’ within the list.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the limit for the number of names from the user.
Step 3:
        - Initialize an empty list li to store names.
        - Initialize a counter variable c to 0.
        - For each element x in the range from 1 to n:
           - Take a name from the user and append it to li.
        - For each name i in li:
           - For each character j in i:
                - If 'a' is present in j, increment the counter c.
Step 4: Print the value of c.
Step 5: End
```

Source Code:

```
n=int(input("Enter the limit:"))
li=[]
c=0
for x in range(n):
        x=input("Enter a name:")
        li.append(x)
for i in li:
        for j in i:
                if 'a' in j.lower():
                        c=c+1
print("The occurance of 'a' is:",c)
```

Output:

```
Enter the limit:3
Enter a name:arun
Enter a name:varun
Enter a name:kiran
The occurance of 'a' is: 3
```

Experiment No: 6

Aim :

Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Initialize list1 with elements [1, 2, 3, 4, 5] & list2 with elements [2, 4, 6, 8] and print lists.
Step 3:
        - If the length of list1 is equal to the length of list2:
        - Print "List have same length" otherwise "List have different length".
Step 4:
        - Initialize variables s1 and s2 to 0.
        - For each element i in list1, add it to the sum s1.
           - Print "The sum of the first list is", s1.
        - For each element i in list2, add it to the sum s2.
           - Print "The sum of the second list is", s2.
Step 5:
        - If s1 is equal to s2:
        - Print "Sum of lists is same" otherwise "The sum of lists is different".
Step 6:
        - Print "The common element in lists is: ".
        - For each element i in list1:
           - For each element j in list2:
                - If i is equal to j, print i.
Step 7: End
```

Source Code:

```
list1 = [1, 2, 3, 4, 5]
list2 = [2, 4, 6, 8]
print("Elements in list one is: ", list1)
print("Elements in list two is: ", list2)
if len(list1) == len(list2):
    print("Lists have same length")
else:
    print("Lists have different length")
s1 = 0
s2 = 0
for i in range(len(list1)):
    s1 = s1 + list1[i]
print("The sum of list1 is:", s1)
for i in range(len(list2)):
    s2 = s2 + list2[i]
print("The sum of list2 is:", s2)
if s1 == s2:
    print("The sum of lists are same")
else:
    print("The sum of lists are different")
print("The common element in list is: ")
for i in list1:
    for j in list2:
        if i == j:
            print(i)
```

Output:

```
Elements in list one is: [1, 2, 3, 4, 5]
Elements in list two is: [2, 4, 6, 8]
Lists have different length
The sum of list1 is: 15
The sum of list2 is: 20
The sum of lists are different
The common element in list is:
2
4
```

Experiment No: 7

Aim :

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

[eg: onion -> oni$n]

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Take a string from the user and store it in variable a.
Step 3:
- Initialize a variable b with the first character of a.
- Initialize a variable s with the substring of a starting from the second character.
Step 4:
- For each character i in the range of the length of s:
- If s[i] is equal to the first character of a:
- Append "$" to b otherwise, Append s[i] to b.
Step 5: Print the modified string b.
Step 6: End
```

Source Code:

```
a = input("Enter a string: ")
b = a[0]
s = a[1:len(a)]
for i in range(len(s)):
if s[i] == a[0]:
b = b + "$"
else:
b = b + s[i]
print(b)
```

Output:

```
Enter a string: malayalam
malayala$
```

Experiment No: 8

Aim :

Create a string from given string where first and last characters exchanged. [eg: python –> nythop]

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a string from user and store it in variable a.
Step 3: Create a new string s by concatenating the last character of a, the substring from the second
        character to the second-to-last character, and the first character of a.
Step 4: Print the modified string s.
Step 5: End
```

Source Code:

```
a = input("Enter a string: ")
s = a[-1] + a[1: -1] + a[0]
print(s)
```

Output:

```
Enter a string: programming
grogramminp
```

Experiment No: 9

Aim :

Accept the radius from user and find area of circle.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the radius of the circle from user and store it in variable r.
Step 3: Calculate the area of the circle using the formula: area = 3.14 * r^2.
Step 4: Print the calculated area.
Step 5: End
```

Source Code:

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

Output:

```
Enter the radius:3
Area of the circle is: 28.259999999999998
```

Experiment No: 10

Aim :

Find biggest of 3 numbers entered.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get 2 numbers from the user and store it in 3 variables a,b and c.
Step 3:
        - If a is greater than both b and c:
           - Print "Largest number is: ", a.
        - Else if b is greater than both a and c:
           - Print "Largest number is: ", b.
        - Else:
           - Print "Largest number is: ", c.
Step 4: End
```

Source Code:

```
print("Enter 3 numbers: ")
a = int(input())
b = int(input())
c = int(input())
if a > b and a > c:
        print("Largest number is : ", a)
elif b > a and b > c:
        print("Largest number is : ", b)
else:
        print("Largest number is : ", c)
```

Output:

```
Enter 3 numbers: 10
20
15
Largest number is : 20
```

Experiment No: 11

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a file name from user with an extension and store it in the variable file.
Step 3:
        - Split the file name using the dot (.) as a separator and extract the last element.
        - Print the extension.
Step 4: End
```

Source Code:

```
file = input("Enter a file name with extension: ")
print("Extension of the file is: ", file.split(".")[-1])
```

Output:

```
Enter a file name with extension: sample.py
Extension of the file is: py
```

Experiment No: 12

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the number of colors from user.
Step 3: Create an empty list named color_list
- For each element i in the range from 1 to n:
- Take the color from the user and append it to the list.
Step 4: Print the list containing the entered colors.
Step 5: Print first & last colour from the list using it’s index
Step 6: End
```

Source Code:

```python
n = int(input("Enter the no of colours: "))
color_list = []
for i in range(n):
        color = input("Enter the colour: ")
        color_list.append(color)
print(color_list)
print("The first color is:", color_list[0])
print("The last color is:", color_list[-1])
```

Output:

```
Enter the no of colours: 3
Enter the colour: red
Enter the colour: green
Enter the colour: blue
['red', 'green', 'blue']
The first color is: red
The last color is: blue
```

Experiment No: 13

Aim :

Accept an integer n and compute n+nn+nnn.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a number n from the user and store it in the variable n.
Step 3:
- Calculate nn as n * 10 + n.
- Calculate nnn as n * 100 + nn.
Step 4:
- Calculate the sum of n, nn, and nnn.
- Print the calculated sum.
Step 5: End
```

Source Code:

```
n=input("enter an integer")
sum= int(n)+int(n*2)+int(n*3)
print(f"value of n+nn+nnn: {sum}")
```

Output:

```
Enter a number: 5
The value of n + nn + nnn is: 615
```

Experiment No: 14

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2:
        - Initialize col1 with elements ['red', 'blue', 'yellow'].
        - Initialize col2 with elements ['green', 'blue'].
Step 3: Print lists col1 & col2
Step 4: Find Colors in col1 Not Present in col2
        - For each element i in the range of the length of col1:
           - If col1[i] is not in col2, print "The color is:" followed by col1[i].
Step 5: End
```

Source Code:

```
col1 = ['red', 'blue', 'yellow']
col2 = ['green', 'blue']
print(col1)
print(col2)
for i in range(len(col1)):
        if col1[i] not in col2:
                print("The color is:", col1[i])
```

Output:

```
['red', 'blue', 'yellow']
['green', 'blue']
The color is: red
The color is: yellow
```

Experiment No: 15

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Take two strings from the user and store it in variables a & b.
Step 3:
        - Create a new string s1 by concatenating the first character of b and the substring
          of a starting from the second character.
        - Create a new string s2 by concatenating the first character of a and the substring
          of b starting from the second character.
Step 4: Print the concatenation of s1 and s2.
Step 5: End
```

Source Code:

```
a = input("Enter first string: ")
b = input("Enter second string: ")
s1 = b[0] + a[1:]
s2 = a[0] + b[1:]
print("The swapped string is:", s1 + s2)
```

Output:

```
Enter first string: good morning
Enter second string: hello world
The swapped string is: hood morninggello world
```

Experiment No: 16

Aim :

Sort dictionary in ascending and descending order.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Initialize my_dict with key-value pairs {'apple': 5, 'banana': 2, 'orange': 8, 'grape': 3}.
Step 3:
        - Create a new dictionary ascending_dict by sorting the items of my_dict in ascending order & print
Step 4:
        - Create a new dictionary descending_dict by sorting the items of my_dict in descending order & print.
Step 5: End
```

Source Code:

```
my_dict = {'apple': 5, 'banana': 2, 'orange': 8, 'grape': 3}
ascending_dict = dict(sorted(my_dict.items()))
print("Ascending is:", ascending_dict)
descending_dict = dict(sorted(my_dict.items(), reverse=True))
print("Descending is:", descending_dict)
```

Output:

```
Ascending is: {'apple': 5, 'banana': 2, 'grape': 3, 'orange': 8}
Descending is: {'orange': 8, 'grape': 3, 'banana': 2, 'apple': 5}
```

Experiment No: 17

Aim :

Merge two dictionaries.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2:
        - Initialize fruit_dict with key-value pairs {'apple': 5, 'banana': 2, 'orange': 8, 'grape': 3}.
        - Initialize veg_dict with key-value pairs {'carrot': 10, 'broccoli': 5, 'spinach': 7, 'tomato': 12}.
Step 3:
        - Create a new dictionary merged_dict as a copy of fruit_dict.
        - Update merged_dict with the items from veg_dict.
Step 4: Print merged_dict
Step 5: End
```

Source Code:

```
fruit_dict = {'apple': 5, 'banana': 2, 'orange': 8, 'grape': 3}
veg_dict = {'carrot': 10, 'broccoli': 5, 'spinach': 7, 'tomato': 12}
merged_dict = fruit_dict.copy()
merged_dict.update(veg_dict)
print("Merged Dictionary:", merged_dict)
```

Output:

```
Merged Dictionary: {'apple': 5, 'banana': 2, 'orange': 8, 'grape': 3, 'carrot': 10, 'broccoli': 5,
 'spinach': 7, 'tomato': 12}
```

Experiment No: 18

Aim :

Find gcd of 2 numbers.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get two numbers from the user and store it in two variables num1 & num2.
Step 3: Initialize a variable i to 1.
Step 4:
        - While i is less than or equal to num1 and num2:
        - If both num1 and num2 are divisible by i:
           - Update the GCD variable gcd to the current value of i.
           - Increment i by 1.
Step 5: Print the calculated GCD.
Step 6: End
```

Source Code:

```
a = int(input("Type the first number : "))
b = int(input("Type the second number : "))
while b != 0:
   r = a % b
   a = b
   b = r
print("The GCD of the numbers is", a)
```

Output:

```
Enter 1st number: 100
Enter 2nd number: 24
GCD is: 4
```

Experiment No: 19

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Initialize mylist with elements [12, 13, 20, 45, 43, 23, 36].
Step 3: Create a new list mylist by using list comprehension to filter out even numbers from the
        original list.
Step 4: Print the modified list mylist after removing even numbers.
Step 5: End
```

Source Code:

```
mylist = [12, 13, 20, 45, 43, 23, 36]
mylist = [x for x in mylist if x % 2 != 0]
print(mylist)
```

Output:

```
[13, 45, 43, 23]
```

Experiment No: 20 **CO2**

Aim :

Program to find the factorial of a number.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a number from the user and store it in the variable num.
Step 3: Initialize a variable fact to 1.
Step 4:
- For each value i in the range from 1 to num + 1:
- Multiply fact by i.
Step 5: Print the result fact.
Step 6: End
```

Source Code:

```
fact = 1
num = int(input("Enter the number: "))
for i in range(1, num + 1):
        fact = fact * i
print("Factorial of", num, "is:", fact)
```

Output:

```
Enter the number: 5
Factorial of 5 is: 120
```

Experiment No: 21

Aim :

Generate Fibonacci series of N terms.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the limit for the number of terms in the Fibonacci series from the user & store it in the variable limit.
Step 3: Initialize first and second variables to 0 and 1 respectively.
Step 4:
    - Print "The Fibonacci series up to", limit, "terms are:"
    - Print the values of first and second.
Step 5:
    - For each value i in the range from 0 to (limit - 2):
        - Calculate the next term (third) as the sum of first and second.
        - Update first with the value of second.
        - Update second with the value of third.
        - Print the value of third.
Step 6: End
```

Source Code:

```
first = 0
second = 1
limit = int(input("Enter the limit: "))
print("The Fibonacci series up to", limit, "terms are:")
print(first)
print(second)
for i in range(limit - 2):
        third = first + second
        first = second
        second = third
        print(third)
```

Output:

```
Enter the limit: 10
The Fibonacci series up to 10 terms are:
0
1
1
2
3
5
8
13
21
34
```

Experiment No: 22

Aim :

Find the sum of all items in a list.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the limit for the number of elements in the list from the user & store it in variable limit.
Step 3: Initialize an empty list named list and also a variable sum to 0.
Step 4:
        - Print "Enter", limit, "numbers:".
        - For each value i in the range from 0 to limit:
            - Get a number from user and store it in list2 and append it to the list.
Step 5: Print the entered list.
Step 6:
        - For each element i in the list:
            - Add i to the sum.
Step 7: Print the calculated sum.
Step 8: End
```

Source Code:

```
list = []
sum = 0
limit = int(input("Enter the limit: "))
print("Enter", limit, "numbers: ")
for i in range(limit):
    list2 = int(input())
    list.append(list2)
print(list)
for i in list:
    sum = sum + i
print("Sum of list: ", sum)
```

Output:

```
Enter the limit: 6
Enter 6 numbers:
5
12
3
15
10
20
[5, 12, 3, 15, 10, 20]
Sum of list: 65
```

Experiment No: 23

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the starting and ending range in 4 digits from the user & store it in start, end respectively.
Step 3: Initialize an empty list named perfect to store perfect square numbers.
Step 4:
        - For each number i in the range from start to end:
        - Initialize a flag variable to 0.
        - Initialize num as i.
        - While num > 0:
           - Extract the last digit from num.
           - If digit is not in [0, 2, 4, 6, 8], set flag to 1 and break.
              - Update num by removing the last digit.
           - If flag is 0 and the square root of i is an integer:
                - Append i to the perfect list.
Step 5: Print the elements in the "perfect" list.
Step 6: End
```

Source Code:

```
import math
start = int(input("Enter a starting range in 4 digits: "))
end = int(input("Enter an ending range in 4 digits: "))
perfect = []
for i in range(start, end + 1):
        flag = 0
        num = i
        while num > 0:
                digit = num % 10
                if digit not in [0, 2, 4, 6, 8]:
                        flag = 1
                        break
                num = int(num / 10)
        if flag == 0 and math.sqrt(i) % 1 == 0:
        perfect.append(i)
print("The list of perfect square numbers are: ", perfect)
```

Output:

```
Enter a starting range in 4 digits: 1000
Enter an ending range in 4 digits: 9999
The list of perfect square numbers are: [4624, 6084, 6400, 8464]
```

Experiment No: 24

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/Pseudocode:

```
Step 1: Start
Step 2: Get the step number from user and store it in variable n.
Step 3:
    - For each value i in the range from 1 to n + 1:
    - For each value j in the range from 1 to i + 1:
        - Print the product of i and j followed by a space, without a newline.
    - Print a newline to move to the next row.
Step 4: End
```

Source Code:

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

Output:

```
Enter the step number: 4
1
2 4
3 6 9
4 8 12 16
```

Experiment No: 25

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a string from user and store it in the variable string.
Step 3: Initialize a variable count to 0.
Step 4:
        - For each character i in the range of the length of the string:
           - If string[i] is not a space (' '), increment the count by 1.
Step 5: Print the calculated count.
Step 6: End
```

Source Code:

```python
string = input("Enter the string: ")
count = 0
for i in range(len(string)):
        if string[i] != ' ':
                count = count + 1
print("The total number of characters in the string is", count)
```

Output:

```
Enter the string: programming
The total number of characters in the string is 11
```

Experiment No: 26

Aim :

Add the 'ing' at the end of the given string, if it already ends with 'ing',then add 'ly'.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a string from user and store it in the variable str.
Step 3:
        - If the string str ends with "ing":
           - Append "ly" to the string str.
        - Else:
           - Append "ing" to the string str.
Step 4: Print the Modified string.
Step 5: End
```

Source Code:

```
str = input("Enter a string: ")
if str.endswith("ing"):
        str = str + "ly"
else:
        str = str + "ing"
print("Modified string is:", str)
```

Output:

```
Enter a string: driving
Modified string is: drivingly
```

Experiment No: 27

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the number of words in the list from the user.
Step 3: Initialize an empty list named list to store words.
Step 4:
        - For each value i in the range from 0 to n:
           - Get x from and append it to the list.
Step 5: Print the entered list.
Step 6:
        - Initialize max to the length of the first word in the list.
        - Initialize temp to the first word in the list.
        - For each word i in the list:
           - If the length of i is greater than max, update max to the length of i and temp to i.
Step 7: Print "The word having the longest length is:", temp, "and its length is:", max.
Step 8: End
```

Source Code:

```
n = int(input("Enter the number of words: "))
list = []
for i in range(n):
        x = (input("Enter the word: "))
        list.append(x)
print(list)
max = len(list[0])
temp = list[0]
for i in list:
        If len(i) > max:
                max = len(i)
                temp = i
```

```
print("The word having longest length is:",temp,"and its length is:",max)
```

Output:

```
Enter the number of words: 3
Enter the word: abhishek
Enter the word: nandana
Enter the word: ajith
Word list: ['abhishek', 'nandana', 'ajith']
The word with the longest length is: abhishek and its length is: 8
```

Experiment No: 28

Aim :

Construct following pattern using nested loop

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the number of rows from the user.
Step 3:
        - For each value i in the range from 1 to n:
            - For each value j in the range from 0 to i:
                - Print " * " without a newline.
        - Print a newline to move to the next row.
Step 4:
        - For each value i in the range from n to 1, with a step of -1:
            - For each value j in the range from 0 to i:
                    - Print " * " without a newline.
        - Print a newline to move to the next row.
Step 5: End
```

Source Code:

```
n = int(input("Enter the number of rows: "))
for i in range(1, n):
        for j in range(i):
                print(" * ", end=' ')
        print()
for i in range(n, 0, -1):
        for j in range(i):
                print(" * ", end=' ')
        print()
```

Output:

```
Enter the number of rows: 5

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

Experiment No: 29

Aim :

Generate all factors of a number.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get a number from the user.
Step 3:
        - Print "The factors of", n, "are:".
        - For each value i in the range from 1 to n + 1:
            - If n is divisible by i (n % i == 0), print i.
Step 4: End
```

Source Code:

```
n = int(input("Enter the number: "))
print("The factors of", n, "are:")
for i in range(1, n + 1):
        if n % i == 0:
                print(i)
```

Output:

```
Enter the number: 15
The factors of 15 are:
1
3
5
15
```

Experiment No: 30

Aim :

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

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Get the side of a square from the user.
Step 3:
        - Define a lambda function s_area that takes a single parameter and returns the square of that
        parameter.
        - Calculate the area of the square using s_area(a).
        - Print “Area of the square”
Step 4: Get the length and breadth of a rectangle from the user.
Step 5:
        - Define a lambda function r_area that takes two parameters and returns their product.
        - Calculate the area of the rectangle using r_area(l, b).
        - Print “Area of rectangle”
Step 6: Get the base and heigh of a triangle.
Step 7:
        - Define a lambda function t_area that takes two parameters and returns half of their product.
        - Calculate the area of the triangle using t_area(b, h).
        - Print "Area of triangle"
Step 8: End
```

Source Code:

```
a = int(input("Enter the sides of square: "))
s_area = lambda a: a * a
print("Area of square is: ", s_area(a))
l = int(input("Enter the length of rectangle: "))
b = int(input("Enter the breadth of rectangle: "))
r_area = lambda l, b: l * b
print("Area of rectangle is: ", r_area(l,b))
b = int(input("Enter the base of the triangle: "))
h = int(input("Enter the height of the triangle: "))
t_area = lambda b, h:.5 * b * h
print("Area of triangle is: ", t_area(b,h))
```

Output:

```
Enter the sides of square: 4
Area of square is: 16
Enter the length of rectangle: 5
Enter the breadth of rectangle: 6
Area of rectangle is: 30
Enter the base of the triangle: 5
Enter the height of the triangle: 4
Area of triangle is: 10.0
```

Experiment No: 31 **CO3**

Aim :

Work with built-in packages.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Print "3 to the power 2: " followed by the result of 3 raised to the power of 2 using
        math.pow().
Step 3: Print "Square root of 64: " followed by the square root of 64 using math.sqrt().
Step 4: End
```

Source Code:

```
import math
print("3 to the power 2: ", math.pow(3, 2))
print("Square root of 64: ", math.sqrt(64))
```

Output:

```
3 to the power 2: 9.0
Square root of 64: 8.0
```

Experiment No: 32

Aim :

Create a package graphics with modules rectangle, circle and sub-package 3-Dgraphics 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/Pseudocode:

```
Step 1: Start
Step 2: Create a package Graphics
Step 3:
        - Create two python files rectangle.py & circle.py
        - Define Functions to calculate the area and perimeter in rectangle.py & circle.py
Step 4:
        - Create a subpackage 3D-Graphics and create two python files cuboid.py & sphere.py
        - Define Functions to calculate the area and perimeter in cuboid.py & sphere.py
Step 5: Import and Use the Functions in main.py
        - Import the functions from the defined modules to use them in the main script.
        - Print the results for rectangle, circle, cuboid, and sphere.
Step 8: End
```

Source Code:

Package Graphics:

rectangle.py

```
def area(h, w):
   return h * w
def perimeter(h, w):
     return 2 * (h + w)
```

circle.py

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

Package 3D-Graphics:

cuboid.py

```
def area(l, h, w):
   return (2 * l * h) + (2 * h * w) + (2 * l * w)
def perimeter(l, h, w):
   return 4 * (l + h + w)
```

sphere.py

```
import math
def area(r):
   return 4 * math.pi * r * r
def perimeter(r):
   return math.pi * 2 * r
```

main.py

```
import grahics.rectangle as re
import grahics.circle as c
import grahics.trd_graphics.cuboid as cb
import grahics.trd_graphics.sphere as s
print("Rectangle")
print("Area:", re.area(5, 6))
print("Perimeter:", re.perimeter(5, 6))
print("Circle")
print("Area:", c.area(4))
print("Perimeter:", c.perimeter(4))
print("Cuboid")
print("Area:", cb.area(4, 5, 6))
print("Perimeter:", cb.perimeter(4, 5, 6))
print("Shere")
print("Area:", s.area(5))
print("Perimeter:", s.perimeter(5))
```

Output:

```
Rectangle
Area: 30
Perimeter: 22
Circle
Area: 50.26548245743669
Perimeter: 25.132741228718345
Cuboid
Area: 148
Perimeter: 60
Sphere
Area: 314.1592653589793
Circumference: 31.41592653589793
```

Experiment No: 33 **CO4**

Aim :

Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.

Algorithm/Pseudocode:

```
Step 1: Start
Step 2: Define Rectangle Class
        - Define a class named Rectangle.
        - Include a constructor (__init__) that initializes length and breadth attributes.
        - Include methods for calculating the area and perimeter of the rectangle.
Step 3: Input Dimensions for Rectangle 1
        - Prompt the user to enter the length and breadth of rectangle 1 (l1, b1).
        - Create an instance (r1) of the Rectangle class with the provided dimensions.
Step 4: Calculate and Print Area and Perimeter of Rectangle 1
        - Calculate the area and perimeter of rectangle 1 using the methods of the Rectangle class.
        - Print the calculated area and perimeter for rectangle 1.
Step 5: Input Dimensions for Rectangle 2
        - Prompt the user to enter the length and breadth of rectangle 2 (l2, b2).
        - Create an instance (r2) of the Rectangle class with the provided dimensions.
Step 6: Calculate and Print Area and Perimeter of Rectangle 2
        - Calculate the area and perimeter of rectangle 2 using the methods of the Rectangle class.
        - Print the calculated area and perimeter for rectangle 2.
Step 7: Compare Areas of Rectangle 1 and Rectangle 2
        - Compare the areas (a1 and a2) of rectangle 1 and rectangle 2.
        - Print whether the area of rectangle 1 is higher, equal area, or the area of rectangle 2 is
        higher.
Step 8: End
```

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)

l1 = int(input("Enter length 1: "))
b1 = int(input("Enter breadth 1: "))
r1 = Rectangle(l1, b1)
print("Area of rectangle 1:", r1.area())
print("Perimeter of rectangle 1:", r1.perimeter())
l2 = int(input("Enter length 2: "))
b2 = int(input("Enter breadth 2: "))
r2 = Rectangle(l2, b2)
print("Area of rectangle 2:", r2.area())
print("Perimeter of rectangle 2:", r2.perimeter())
a1 = r1.area()
a2 = r2.area()
if a1 > a2:
        print("Area of rectangle 1 is higher")
elif a1 == a2:
        print("Areas are equal")
else:
        print("Area of rectangle 2 is higher")
```

Output:

```
Enter length 1: 4
Enter breadth 1: 6
Area of rectangle 1: 24
Perimeter of rectangle 1: 20
Enter length 2: 5
Enter breadth 2: 8
Area of rectangle 2: 40
```

Experiment No: 34

Aim :

Create a Bank account with members account number, name, type of account and balance.Write constructor and methods to deposit at the bank and withdraw an amount from the bank.

Algorithm/Pseudocode:

```
Step1.Start
Step2. Define the Bank class with the following methods:
    -__init__(self, acc_no, name, type, bal):
    - Initialize account number (acc_no), account holder name (name), account type (type), and
       account balance (`bal`).
    - deposit(self, amt):
        - If the amount is less than or equal to 0, display an error message.
        - Otherwise, add the amount to the balance and display the updated balance.
        - withdraw(self, amt):
        - If the amount is less than or equal to 0, display an error message.
        - If the amount is greater than the balance, display an "insufficient balance" message.
        - Otherwise, subtract the amount from the balance and display the updated balance.
    - display(self):
        - Display account details, including account number, holder name, account type, and current
       balance.
Step3. Input User Data:
     - Prompt the user to enter the account number.
     - Prompt the user to enter their name.
     - Prompt the user to enter the account type.
     - Prompt the user to enter the initial balance.
Step4. Create a Bank object usr1 with the provided user data.

Step5. Enter a loop to manage user operations:
     - Display the options:
         - 1) Deposit
         - 2) Withdrawal
         - 3) Account Information
        - 4) Exit
    - Prompt the user to select an option.
Step6. Handle User Choices:
    - If the option is 1 (Deposit):
        - Prompt the user to enter the deposit amount.
        - Call the deposit method of usr1 with the entered amount.
    - If the option is 2 (Withdrawal):
        - Prompt the user to enter the withdrawal amount.
        - Call the `withdraw` method of `usr1` with the entered amount.
    - If the option is 3 (Account Information):
        - Call the display method of usr1 to show account details.
    - If the option is 4 (Exit):
        - Exit the program.
    - Otherwise:
        - Display an "invalid option" message.
Step7. Repeat Steps 5–6 until the user selects the exit option.
Step8. End
```

Source Code:

```
class Bank:
    def __init__(self,acc_no,name,type,bal):
        self.acc_no=acc_no
        self.name=name
        self.type=type
        self.bal=bal
    def deposit(self,amt):
        if amt<=0:
            print("it should be an positive number")
        else:
            self.bal+=amt
            print("Your current balance is ",self.bal)
    def withdraw(self,amt):
        if amt<=0:
            print("it should be an positive number")
        elif amt>self.bal:
            print("insufficient balance")
        else:
            self.bal-=amt
            print("Your current balance is ",self.bal)
    def display(self):
        print(f"account number: {self.acc_no}")
        print(f"account holder: {self.name}")
        print(f"account type: {self.type}")
        print(f"Current balance: {self.bal}")

no=int(input("enter the account number"))
name=input("enter your name")
type=input("enter the account type")
bal=int(input("enter your balance"))
usr1=Bank(no,name,type,bal)
while True:
    print("\n1)DEPOSIT\n2)WITHDRAWAL\n3)ACCOUNT INFORMATION\n4)EXIT\n")
    opt=int(input("enter the option"))
    if opt==1:
        amt=int(input("enter your amount"))
        usr1.deposit(amt)
    elif opt==2:
        amt=int(input("enter your withdrawal amount"))
        usr1.withdraw(amt)
    elif opt==3:
        usr1.display()
    elif opt==4:
        exit(0)
    else:
        print("invalid option")
```

Output:

```
enter the account number:233233
enter your name:amal
enter the account type:savings
enter your balance: 2000

1)DEPOSIT
2)WITHDRAWAL
3)ACCOUNT INFORMATION
4)EXIT

enter the option:2
enter your withdrawal amount:1500
Your current balance is:  500

1)DEPOSIT
2)WITHDRAWAL
3)ACCOUNT INFORMATION
4)EXIT

enter the option:3
account number: 233233
account holder: amal
account type:  savings
Current balance: 500

1)DEPOSIT
2)WITHDRAWAL
3)ACCOUNT INFORMATION
4)EXIT

enter the option4
Process finished with exit code 0
```

Experiment No: 35

Aim :

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/Pseudocode:

```
Step1. Start
Step2. Define the Publisher class:
            - __init__(self, name):
            - Print "publisher class activated".
            - Initialize the name attribute.
            - display(self):
            - Print the publisher's name.
Step3. Define the Book class (inherits from Publisher):
            - __init__(self, name, title, author):
            - Call the __init__ method of the Publisher class using super().
            - Print "Book class is activated".
            - Initialize the title and author attributes.
             - display(self):
                - Call the display method of the Publisher class using super().
                - Print the book's title and author.
Step4.   Define the Python class (inherits from `Book`):
            - __init__(self, name, title, author, price, no_pages):
            - Call the __init__ method of the Book class using super().
            - Print "python class activated".
            - Initialize the `price` and `no_pages` attributes.
            - display(self):
                - Call the `display` method of the `Book` class using `super()`.
                - Print the price and the number of pages.
Step5.   Create an object of the `Python` class:
            - Pass the following arguments to the constructor:
```

```
        - name: "DC", title: "Aadujeevitham", author: "Basheer" , price: 54 , no_pages: 250
        - The __init__ method of Python, Book, and Publisher will execute sequentially,
        initializing their respective attributes.

Step6. Call the `display` method of the `Python` object:
        - The display method of `Python` will execute, which calls the display methods of `Book` and
          Publisher sequentially.
        - Display all details, including publisher name, book title, author, price, and the number of
          pages.
Step7. End
```

Source Code:

```
class Publisher:
    def __init__(self,name):
        print("publisher class activated")
        self.name=name
    def display(self):
        print(f"Name: {self.name}")
class Book(Publisher):
    def __init__(self,name,title,author):
        super().__init__(name)
        print("Book class is activated")
        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_pages):
        super().__init__(name,title,author)
        print("python class activated")
        self.price= price
```

```
 self.no_pages=no_pages
   def display(self):
          super().display()
         print(f"Price: {self.price}")
         print(f"No of pages: {self.no_pages}")
book1=Python("DC","Aadujeevitham","Basheer",540,250)
book1.display()
```

Output:

```
publisher class activated
Book class is activated
python class activated
Name: DC
Title: Aadujeevitham
Author: Basheer
Price: 540
No of pages: 250
```

Experiment No: 36

Aim :

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

Algorithm/Pseudocode:

```
Step1. Start
Step2. Input Dimensions for Rectangle 1
  - Prompt the user to enter the length of Rectangle 1 (l1).
  - Prompt the user to enter the width of Rectangle 1 (w1).
Step3. Create Rectangle 1
  - Use the inputs (l1 and w1) to create an instance of the Rectangle class (r1).
Step4. Input Dimensions for Rectangle 2
  - Prompt the user to enter the length of Rectangle 2 (l2).
  - Prompt the user to enter the width of Rectangle 2 (w2).
Step5. Create Rectangle 2
  - Use the inputs (l2 and w2) to create an instance of the Rectangle class (r2).
Step6. Calculate Areas
  - Call the area method of Rectangle 1 (r1) to compute its area.
  - Call the area method of Rectangle 2 (r2) to compute its area.
Step7. Compare Areas
  - Use the __lt__ method to compare the areas of r1 and r2.
Step8. Output the Result
  - If r1 is smaller than r2 (r1 < r2):
    - Print: "Area of r2 is greater: <area of r2>."
  - Else:
    - Print: "Area of r1 is greater: <area of r1>."
Step9. End
```

Source code:

```
class Rectangle:
   def __init__(self,l,w):
      self.__length=l
      self.__width=w
   def area(self):
      return self.__length*self.__width
   def __lt__(self, other):
      if self.area()< other.area():
         return True
      else:
         return False
l1=int(input("enter the length: "))
w1=int(input("enter the width: "))
r1=Rectangle(l1,w1)
l2=int(input("enter the length: "))
w2=int(input("enter the width: "))
r2=Rectangle(l2,w2)
if r1<r2:
   print(f"area of r2 is greater:{r2.area()}")
else:
   print(f"area of r1 is greater:{r1.area()}")
```

Output:

```
enter the length: 2
enter the width: 3
enter the length: 4
enter the width: 2
area of r2 is greater:8
```

Experiment No: 37

Aim :

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

Algorithm/Pseudocode:

```
Step1. Start
Step2. Initialize Time Object 1 (t1)
  - Assign hour, minute, and second values for Time Object 1(t1).
Step3. Initialize Time Object 2 (t2)
  - Assign hour, minute, and second values for Time Object 2 (t2).
Step4. Define Addition of Two Time Objects (__add__)
  - Add the hour values of t1 and t2 to calculate h.
  - Add the minute values of t1 and t2 to calculate m.
  - Add the second values of t1 and t2 to calculate s.
Step5. Adjust Seconds if Greater than or Equal to 60
  - If s >= 60:
    - Increment m=m+int(s/60).
    - Update s = s % 60
Step6. Adjust Minutes if Greater than or Equal to 60
  - If m >= 60:
    - Increment h=h+int(m/60)
    - Update m = m % 60
Step7. Create New Time Object(t3)
  - Use the updated h, `m`, and s values to initialize a new Time object (t3).
Step8. Return the New Time Object (t3)
  - The __add__ method returns t3.
Step9. Display the Resulting Time (display Method)
  - Print "hour:minute:second".
  - Print the values of hour, minute, and second from t3.
Step10. End
```

Source code:

```
class Time:
   def __init__(self,h,m,s):
      self.__hour=h
      self.__minute=m
      self.__second=s
   def __add__(self, other):
      h=self.__hour+other.__hour
      m=self.__minute+other.__minute
      s=self.__second+other.__second

      if s>=60:
         m=m+int(s/60)
         s=s%60
      if m>=60:
         h=h+int(m/60)
         m=m%60
      t3=Time(h,m,s)
      return t3
   def display(self):
      print("hour:minute:second")
      print(f"{self.__hour}:{self.__minute}:{self.__second}")

t1=Time(2,34,56)
t2=Time(1,45,50)
t3=t1+t2
t3.display()
```

Output:

```
hour:minute:second
4:20:46
```