MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT MECHANICAL ENGINEERING

PYTHON PROGRAMING

LAB MANUAL

SUBJECT NAME	**Python Programming Lab**
SUBJECT CODE	**2030575**
COURSE-BRANCH	**B. Tech - Mechanical Engineering**
YEAR-SEMESTER	**II - I**
ACADEMIC YEAR	**2021-2022**
REGULATION	**MLRS-R20**

MARRI LAXAMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT

MISSION AND VISION OF THE INSTITUTE:

Our Vision:

To establish as an ideal academic institution in the service of the nation the world and the humanity by graduating talented engineers to be ethically strong globally competent by conducting high quality research, developing breakthrough technologies and disseminating and preserving technical knowledge.

Our Mission:

To fulfill the promised vision through the following strategic characteristics and aspirations:

- Contemporary and rigorous educational experiences that develop the engineers and managers;
- An atmosphere that facilitates personal commitment to the educational success of students in an environment that values diversity and community;
- Prudent and accountable resource management;
- Undergraduate programs that integrate global awareness, communication skills and team building across the curriculum;
- Leadership and service to meet society's needs;
- Education and research partnerships with colleges, universities, and industries to graduate education and training that prepares students for interdisciplinary engineering research and advanced problem solving;
- Highly successful alumni who contribute to the profession in the global society.

Vision and Mission statements of the Department of Mechanical Engineering:

Vision Statement:

"The Mechanical Engineering Department strives immense success in the field of education, research and development by nurturing the budding minds of young engineers inventing sets of new designs and new products which may be envisaged as the modalities to bring about a green future for humanity"

Mission Statement:

1. Equipping the students with manifold technical knowledge to make them efficient and independent thinkers and designers in national and international arena.
2. Encouraging students and faculties to be creative and to develop analytical abilities and efficiency in applying theories into practice, to develop and disseminate new knowledge.

3. Pursuing collaborative work in research and development organizations, industrial enterprises, Research and academic institutions of national and international, to introduce new knowledge and methods in engineering teaching and research in order to orient young minds towards industrial development.

PROGRAM EDUCATIONAL OBJECTIVE

PEO 1: Graduates shall have knowledge and skills to succeed as Mechanical engineer's for their career development.

PEO 2: Graduates will explore in research.

PEO 3: Mechanical Graduates shall have the ability to design products with various interdisciplinary skills

PEO 4: Graduates will serve the society with their professional skills

PROGRAM OUTCOMES

- **A.** Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization for the solution of complex engineering problems.
- **B.** Problem Analysis: Identify, formulate, research, review the available literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.
- **C.** Design and development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specific needs with appropriate considerations for public health safety and cultural, societal and environmental considerations.
- **D.** Conduct investigations of complex problems: Use research based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
- **E.** Modern tool usage: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including predictions and modeling to complex engineering activities with an understanding of the limitations.
- **F.** The Engineer and society: Apply reasoning, informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practices.
- **G.** Environment and sustainability: Understand the impact of the professional engineering solutions in society and environmental context and demonstrate the knowledge of and need for sustainable development.
- **H.** Ethics: Apply ethical principles and commitment to professional ethics, responsibilities and norms of the engineering practice.
- **I.** Individual and team work: Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings.
- **J.** Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as being able to comprehend, write effective reports, design documentation, make effective presentations, give and receive clear instructions.
- **K.** Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **L.** Life – long learning: Recognize the need and have the preparation, ability to engage in independent and life – long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES:

PS01: Students acquire necessary technical skills in mechanical engineering that make them employable graduate.

PSO2: An ability to impart technological inputs towards development of society by becoming an entrepreneur.

COURSE OBJECTIVES:

1. To understand the basic principles of fluid mechanics.

2. To identify various types of flows.

3. To understand boundary layer concepts and flow through pipes.

4. To evaluate the performance of hydraulic turbines.

5. To understand the functioning and characteristic curves of pumps.

COURSE OUTCOMES:

ME 272.1 To analyze and solve electrical circuits using network laws and theorems.

ME 272.2 To understand and analyze basic Electric and Magnetic circuits.

ME 272.3 To study the working principles of Electrical Machines.

ME 272.4 To introduce components of Low Voltage Electrical Installations.

ME 272.5 To identify and characterize diodes.

ME 272.6 To identify and characterize various types of transistors.

INSTRUCTIONS TO THE STUDENTS

1. Every student should obtain a copy of the laboratory manual
2. It is important that all students arrive at each session on time.
3. Dress code: Students must come to the laboratory wearing:
    - Trousers.
    - half-sleeve tops.
    - Leather shoes.
    - Half pants, loosely hanging garments and slippers are not allowed.
4. Students should come with thorough preparation for the experiment to be conducted.
5. Students will not be permitted to attend the laboratory unless they bring the practical record fully completed in all respects pertaining to the experiment conducted in the previous class.
6. Experiment should be started only after the staff-in-charge has checked the experimental setup.
7. All the calculations should be made in the observation book. Specimen calculations for one set of readings have to be shown in the practical record.
8. Wherever graphs are to be drawn, A-4 size graphs only should be used and the same should be firmly attached to the practical record.
9. Practical record and observation should be neatly maintained.
10. They should obtain the signature of the staff-in-charge in the observation book after completing each experiment.
11. Theory regarding each experiment should be written in the practical record before procedure in your own words.

LABORATORY SAFETY PRECAUTIONS

1. Laboratory uniform, shoes & safety glasses are compulsory in the lab.
2. Do not touch anything with which you are not completely familiar. Carelessness may not only break the valuable equipment in the lab but may also cause serious injury to you and others in the lab.
3. Please follow instructions precisely as instructed by your supervisor. Do not start the experiment unless your setup is verified & approved by your supervisor.
4. Do not leave the experiments unattended while in progress.
5. Do not crowd around the equipment's & run inside the laboratory.
6. During experiments material may fail and disperse, please wear safety glasses and maintain a safe distance from the experiment.
7. If any part of the equipment fails while being used, report it immediately to your supervisor. Never try to fix the problem yourself because you could further damage the equipment and harm yourself and others in the lab.
8. Keep the work area clear of all materials except those needed for your work and cleanup after your work.

LIST OF EXPERIMENTS/DEMONSTRATIONS:

PART A: ELECTRICAL

1. Verification of KVL and KCL

2. (i) Measurement of Voltage, Current and Real Power in primary and Secondary Circuits of a Single-Phase Transformer

(ii) Verification of Relationship between Voltages and Currents (Star-Delta, Delta-Delta, Delta- star, Star-Star) in a Three Phase Transformer

3. Measurement of Active and Reactive Power in a balanced Three-phase circuit

4. Performance Characteristics of a Separately Excited DC Shunt Motor

5. Performance Characteristics of a Three-phase Induction Motor

6. No-Load Characteristics of a Three-phase Alternator

PART B: ELECTRONICS

1. Study and operation of

(i) Multi-meters (ii) Function Generator (iii) Regulated Power Supplies (iv)CRO.

2. PN Junction diode characteristics

3. Zener diode characteristics and Zener as voltage Regulator

4. Input & Output characteristics of Transistor in CB / CE configuration

5. Full Wave Rectifier with & without filters

6. Input and Output characteristics of FET in CSconfiguration

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

CONTENTS

Exercise – 1 Python Numbers

1. **Write a programme to determine whether the given year is leap year, using the following formula: a leap year is one that is divisible by four, but not by one hundred, unless it is also divisible by four hundred. For example, 1992, 1996 and 2000 are leap years, but 1967 and 1900 are not. The next leap year falling on a century is 2400**

Solution:

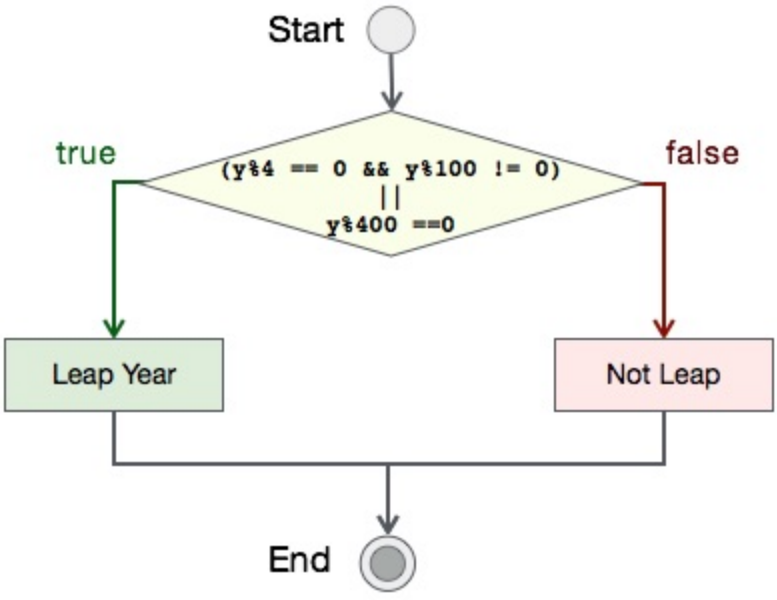
AIM:

To determine whether the given year is leap year or not

ALOGORITHM:

START
Step 1 → Take integer variable year
Step 2 → Assign value to the variable
Step 3 → Check if year is divisible by 4 but not 100, DISPLAY "leap year"
Step 4 → Check if year is divisible by 400, DISPLAY "leap year"
Step 5 → Otherwise, DISPLAY "not leap year"
STOP

FLOW CHART:



SOURCE CODE:

```
n=int(input('enter any year:'))
if (n%4) ==0:
   print('the year is a leap year')
else:
   print('the year is not a leap year')
```

OUTPUT:

```
enter any year:1989
the year is not a leap year
```

(or)

SOURCE CODE:

```
Year=int(input("Enter Year: "))
if((Year % 400 == 0) or
    (Year % 100!= 0) and
    (Year % 4 == 0)):
   print("Given Year is a leap Year");
 # else it is not a leap year
else:
   print ("Given Year is not a leap Year")
```

OUTPUT:

```
enter any year: 1989
the year is not a leap year
```

2. **Write a program to determine the greatest common divisor and least common multiple of a pair of integers**

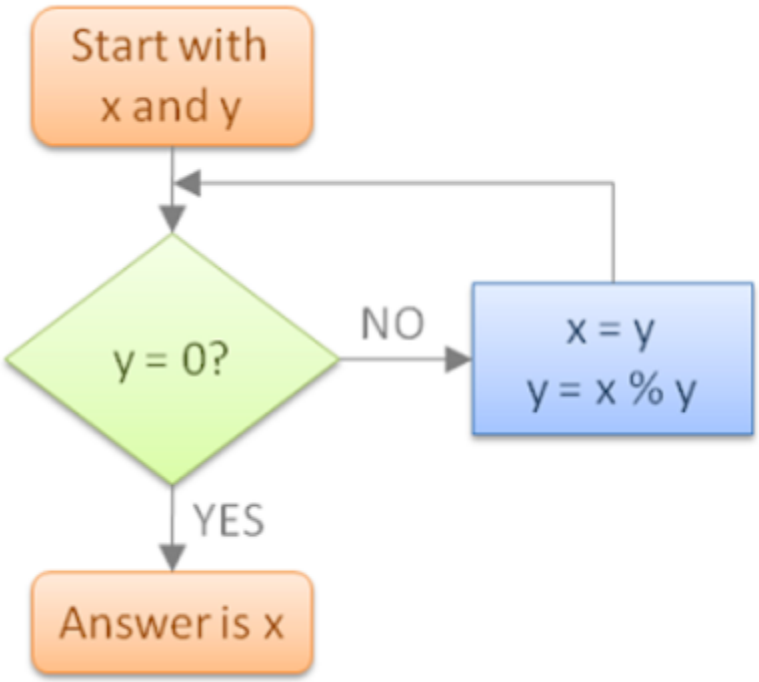
AIM:

To determine the greatest common divisor and least common multiple of a pair of integers

ALGORITHM:

START
Step 1 → Take integer variable number
Step 2 → Assign value to the variable number
Step 3 → Check if the number is gcd or lcm not
Step 4 → Check the number is having same pair of gcd and lcm or not
Stop

FLOW CHART:



SOURCE CODE:

```
num1=int(input('enter any value:'))
num2=int(input('enter any value:'))
a=num1
b=num2
lcm=0
while(num2!=0):
    temp=num2
    num2=num1%num2
    num1=temp
    gcd=num1
    lcm=((a*b)/gcd)
print("\n gcd or hcf of",a,"and",b"=",gcd)
print("\n lcm of",a,"and",b"=",lcm)
```

OUTPUT:

```
enter any value:48
enter any value:4
gcd or hcf of 48 and b'=' 4
 lcm of 48 and b'=' 48.0
```

3. **Create a calculator application. Write a code that will take two numbers and operator in the format. N1 OP N2, where N1 and N2 are floating point or integer values, and OP is one of the following: +, -, *, /, %, **, representing addition, subtraction, multiplication, division, modulus / remainder, and exponential, respectively, and displays the result of carrying out that operation of the input operands.**
   Hint: You may use the string split () method, but you cannot use the exal () built – in function

AIM:

To write a python program code of a calculator application that will take two numbers and operator in the format.

ALGORITHM:

Step 1: Get input from the user (choice corresponding to various operations like addition, subtraction, multiplication, division)

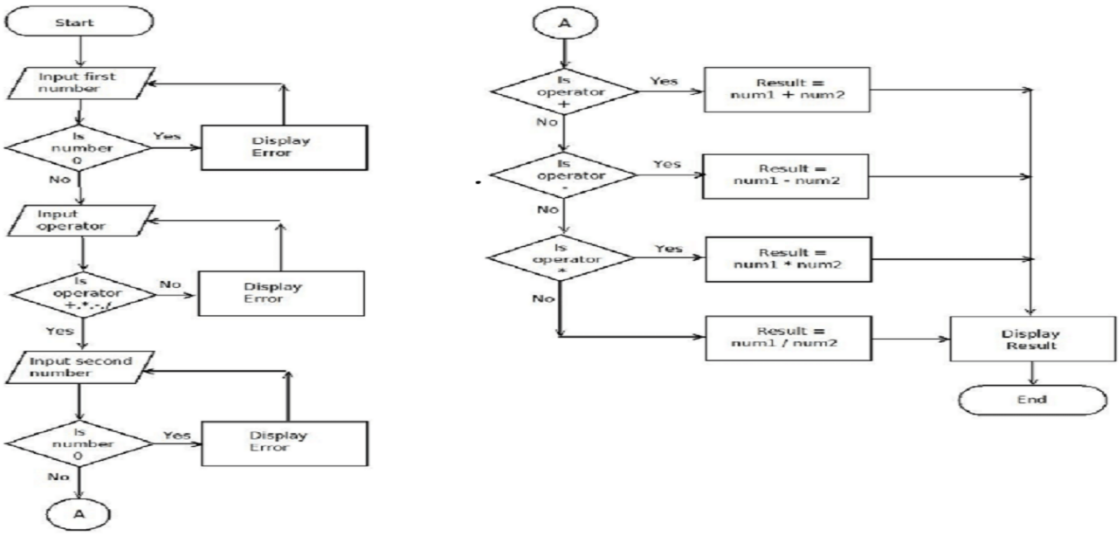
Step 2: Get two integer inputs from the user.

Step 3: Perform the operations based on the user's choice.

Step 4: Print the output

Step 5: End the program

FLOW CHART:



SOURCE CODE:

```
A=int(input('Enter your first number:'))
operator=input('which operation would you like to perform?(+, -, *, /,//,**):');
B=int(input('Enter your second number:'))
#Addition
if operator=="+":
    ans=A+B
    print(str(ans))
#Subtraction
elif operator=="-":
    ans=A-B
    print(str(ans))
#Division
elif operator=="/":
    ans=A/B
    print(str(ans))
#Multiplication
```

```
elif operator=="*":
    ans=A*B
    print(str(ans))
#Interdivision
elif operator=="//":
    ans=A//B
    print(str(ans))
#Exponent
elif operator=="**":
    ans=A**B
    print(str(ans))
else:
    print('wrong answer')
```

OUTPUT:

```
Enter your first number:5
which operation would you like to perform?(+, -, *, /,//,**):/
Enter your second number:4
1.25
```

Exercise -2 Control Flows

1. **Write a programme for checking whether the given number is prime or not?**

AIM:

To write a python programme for checking whether the given number is prime or not

ALGORITHM:

Step1: Start

Step2: [Accept a number] read n

Step3: Set i=2

Step4: Repeat steps 5 and 6 until i<n

Step5: [Check whether n is divisible or not]
if n%i==0 then
Go to step 7
Else
Go to step 6

Step6: Set i = i+1

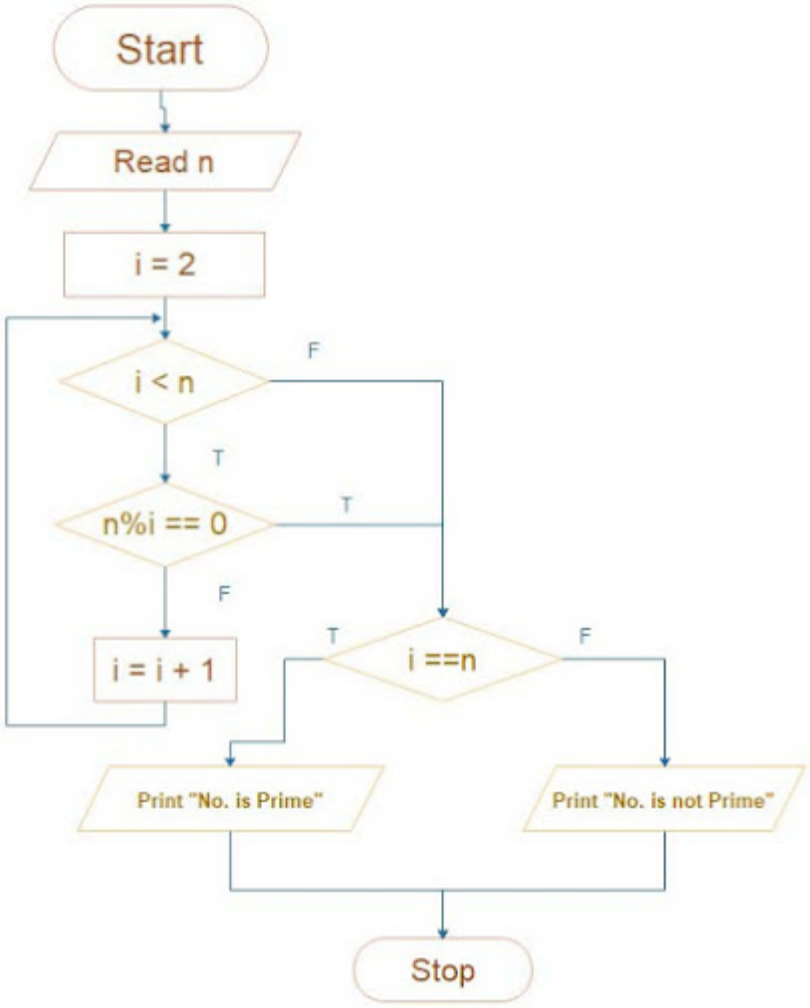
Step7: if i == n then

print “number is prime”

Else
print “number is not prime”

Step8: Stop

FLOWCHART:



SOURCE CODE:

```
number = int(input("Enter any number: "))
if number > 1:
    for i in range(2, number):
        if (number % i) == 0:
            print(number, "is not a prime number")
            break
    else:
        print(number, "is a prime number")
else:
```

```
    print(number, "is not a prime number")
```

OUTPUT:

```
Enter any number: 58
58 is not a prime number
```

2. **Write a program to print Fibonacci series up to given n values?**

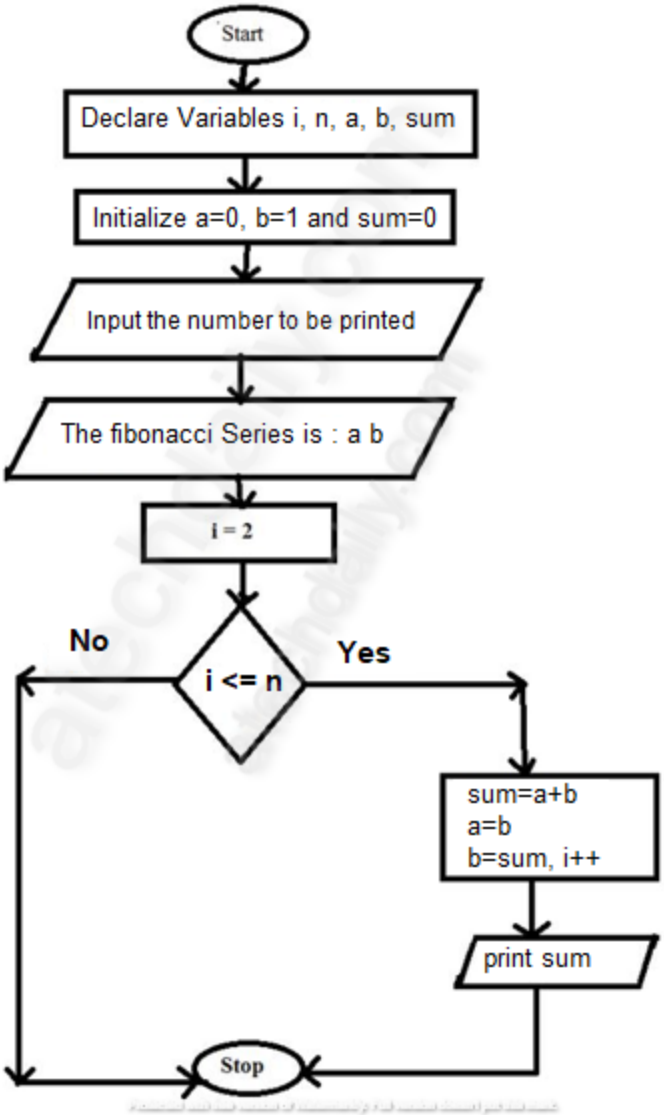
AIM:

To write a program to print Fibonacci series up to given n values

ALGORITHM:

Step 1: Start
Step 2: Declare variable a, b, c, n, i
Step 3: Initialize variable a=0, b=1 and i=2
Step 4: Read n from user
Step 5: Print a and b
Step 6: Repeat until i<=n :
    Step 6.1: c=a+b
    Step 6.2: print c
    Step 6.3: a=b, b=c
    Step 6.4: i=i+1
Step 7: Stop

FLOWCHART:



SOURCE CODE:

```
n=int(input("enter the number of terms needed in fibonacci series:"))
f1,f2=0,1
if n == 1:
    print(f1)
elif n==2:
    print(f1,",f2)
else:
```

```
    print(f1,f2,end=")
    for i in range(3,n+1):
        f3=f1+f2
        print(f3,end=")
        f1=f2
        f2=f3
```

OUTPUT:

enter the number of terms needed in fibonacci series:4

0, 1, 1, 2

3. **Write a program to calculate factorial of given integer number?**

AIM:

To write a program to calculate factorial of given integer number

ALGORITHM:

Step1: Start

Step2: Read number N

Step3: FACT = 1 CTRL = 1

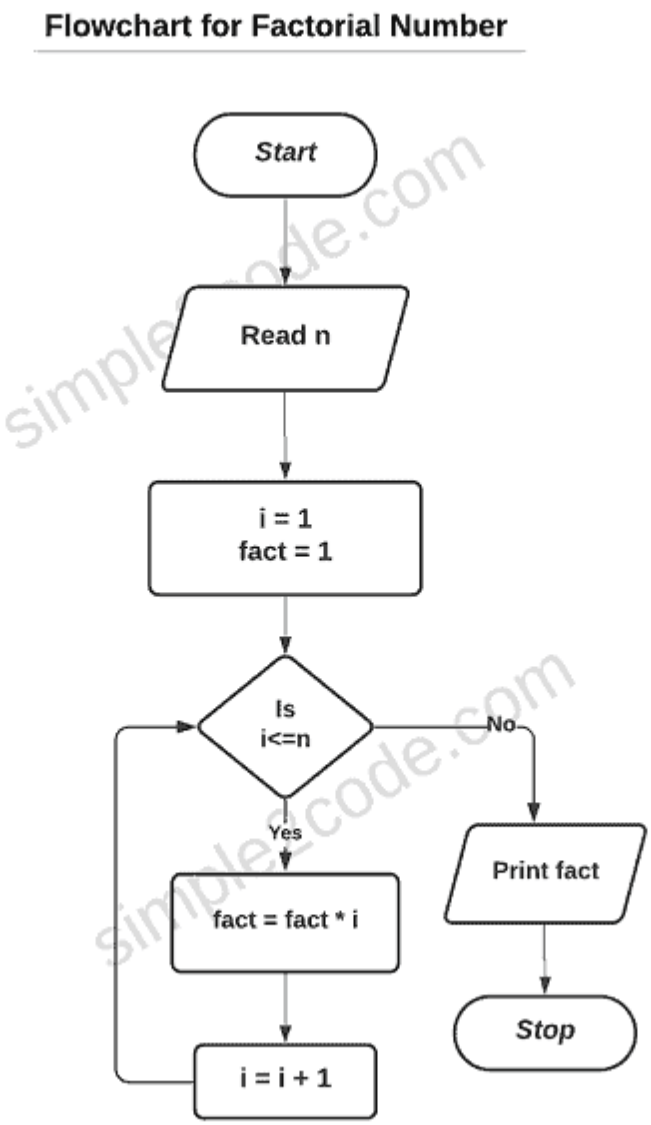
Step4: While (CTRL<=N)
Do
Fact = Fact * i
CTRL = CTRL + 1

Done

Step5: Display Fact

Step6: Stop

FLOWCHART:



SOURCE CODE:

```
num=int(input('enter a number:'))
factorial = 1
if num<0:
    print('Factorial does not exist for negative numbers')
elif num == 0:
    print('The factorial of 0 is 1')
```

```
else:
    for i in range(1,num+1):
        factorial=factorial*i
    print('The factorial of',num,'is',factorial)
```

OUTPUT:

```
enter a number:25
The factorial of 25 is 15511210043330985984000000
```

Exercise -3 Control Flows – Continued

1. **Write a program to calculate value of the following series 1 – x + x2 – x3 + x4 - …..+xn**

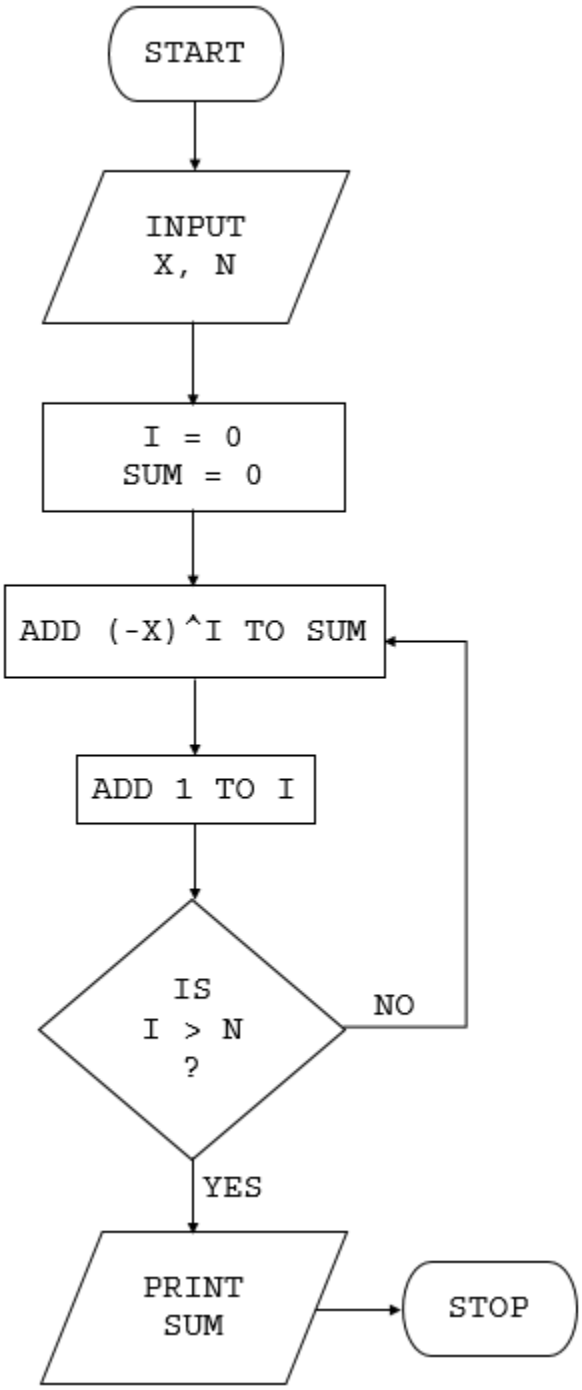
AIM:

To write a python program to cx2alculate value of the following series 1 – x + x2 – x3 + x4 - …+xn

ALGORITHM:

Step1: Start
Step2: Input value of N,X
Step3: I = 1, Sum = 1, Term = 1
Step4: If (I>N) then
Go to Step 9
ENDIF
Step5: Term = -Term * X
Step6: Sum = Sum + Term
Step7: I = I + 1
Step8: Go to Step – 4
Step9: Display value of Sum
Step10: Stop

FLOW CHART:



SOURCE CODE:

```
x=float(input('enter base number:'))
n=int(input('enter the power:'))
sum=1
```

```
for a in range(1,n+1):
    sum=sum+((-1)**a)*(x**a)
print('sum of the series is',sum)
```

OUTPUT:

```
enter base number:5
enter the power:4
sum of the series is 521.0
```

2. **Write a program to print pascal triangle**

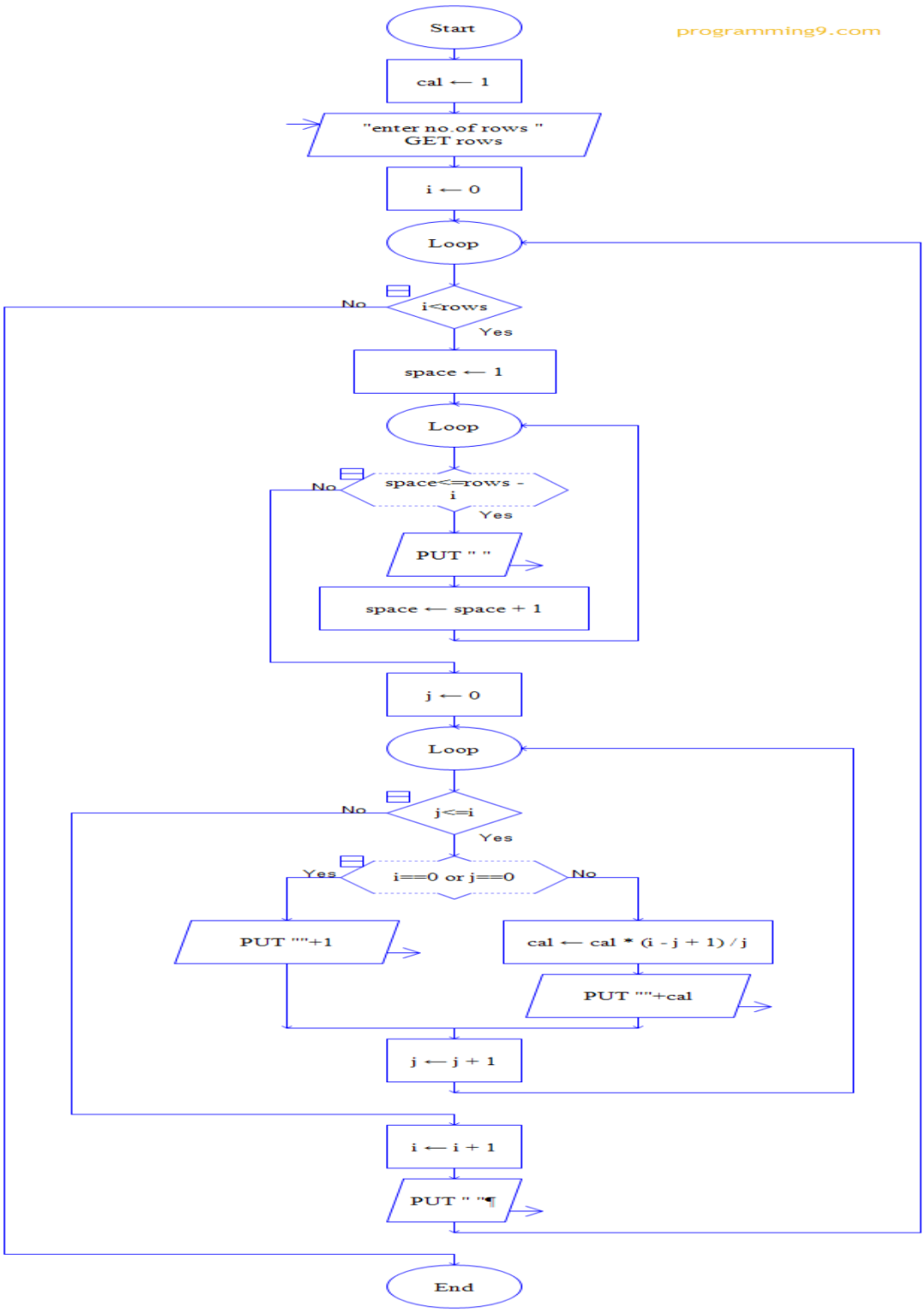
AIM:

To write a python program to print pascal triangle

ALGORITHM:

Step1: Start
Step2: Declare variables i,j,k
Step3: Enter the limits
Step4: Take a number of rows to be printed, let's assume it to be n
Step5: Make outer iteration i from 0 to n times to print the rows.
Step6: Make inner iteration for j from 0 to (N – 1).
Step7: Print single blank space " ".
Step8: Close inner loop (j loop) //its needed for left spacing.
Step9: Make inner iteration for j from 0 to i.
Step10: Print nCr of i and j.
Step11: Close inner loop.
Step12: Print newline character (\n) after each inner iteration.
Step13: Stop

FLOW CHART:


Start
programming9.com
cal ← 1
"enter no.of rows "
GET rows
i ← 0
Loop
No
i<rows
Yes
space ← 1
Loop
No
space<=rows - i
Yes
PUT " "
space ← space + 1
j ← 0
Loop
No
j<=i
Yes
Yes
i==0 or j==0
No
PUT ""+1
cal ← cal * (i - j + 1) / j
PUT ""+cal
j ← j + 1
i ← i + 1
PUT " "¶
End

SOURCE CODE:

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

OUTPUT:

```
Enter the number of rows:8
        1
       1  1
      1  2  1
     1  3  3  1
    1  4  6  4  1
   1  5  10  10  5  1
  1  6  15  20  15  6  1
 1  7  21  35  35  21  7  1
```

Exercise -4 Python Sequences

1. **Write a program to sort the numbers in ascending order and strings in reverse alphabetical orders**

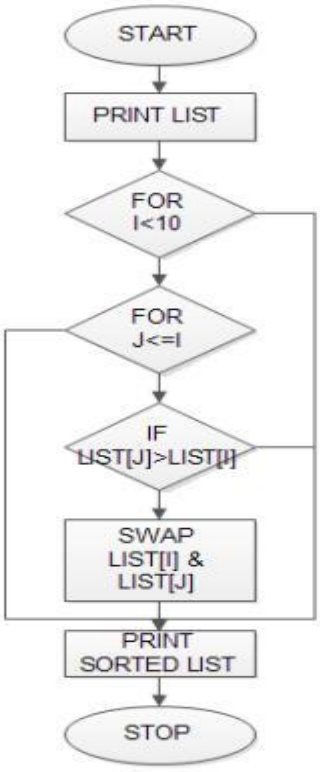
AIM:

To write a python program to sort the numbers in ascending order and strings in reverse alphabetical orders

ALGORITHM:

Step1: Start
Step 2: Enter the array size and read as n.
Step 3: Enter the array elements and read as [i].
Step 4: Print the array elements before sorting.
Step5: After sorting, print the statement array elements.
Step6: Take 2 nested loops, take i variable in 1 loop, and j variable in 1 loop.
Step7: Check condition i> n in the I loop.
Step8: If the condition is false, go to the j loop. Go to the other end.
Step9: In j loop, check condition [j]> [a + j + 1].
Step10: If the condition is true, swap one [j] and one [j + 1]. Otherwise, go to the end.
Step11: Print the array elements [i].
Step 12: End

FLOW CHART:



SOURCE CODE:

```
val=eval(input('enter a list:'))
val.sort()
print('sorted in ascending order:',val)
val.sort(reverse=True)
print('sorted in descending order:',val)
```

OUTPUT:

```
enter a list:[8,9,2,63,79,95,54]
sorted in ascending order: [2, 8, 9, 54, 63, 79, 95]
sorted in descending order: [95, 79, 63, 54, 9, 8, 2]
```

2. **Given an integer value, return a string with the equivalent English text of each digit. For example, an input of 89 results in "eight – nine" being returned. Write a program to implement it**

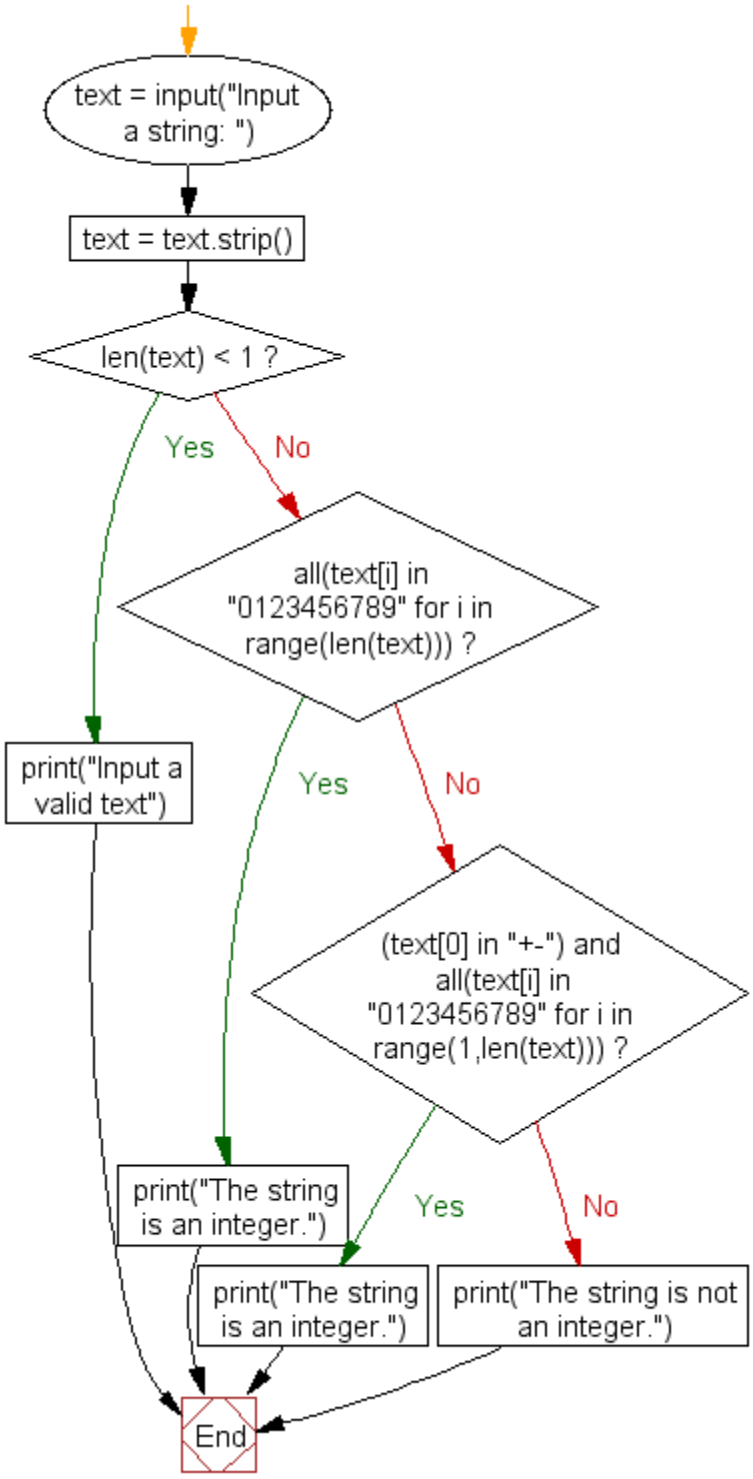
AIM:

To write a python program for an integer value, return a string with the equivalent English text of each digit.

ALGORITHM:

Step1: Start
Step2: Give the operands in the integer format
Step3: Enter the values either in Thousands or Hundreds or tens whatever you want
Step4: Select the values
Step5: The values you have to convert it into the string with the equivalent English text of each digit.
Step6: Print the values in the string format
Step7: Stop

FLOW CHART:


text = input("Input a string: ")
text = text.strip()
len(text) < 1 ?
Yes
No
all(text[i] in "0123456789" for i in range(len(text))) ?
print("Input a valid text")
Yes
No
(text[0] in "+-") and all(text[i] in "0123456789" for i in range(1,len(text))) ?
print("The string is an integer.")
Yes
No
print("The string is an integer.")
print("The string is not an integer.")
End

SOURCE CODE:

```
number=["","One","Two","Three","Four","Five","Six","Seven","Eight","Nine"]
nty=["","","Twenty","Thirty","Fourty","Fifty","Sixty","Seventy","Eighty","Ninty"]
tens=["Ten","Eleven","Twelve","Thirteen","Fourteen","Fifteen","Sixteen","Seventeen","Eightee
n","Nineteen"]
n=int(input("Enter any number:"))
if n>99999:
    print("cant solve for more than 5 digits")
else:
    d=[0,0,0,0,0]
    i=0
    while n>0:
        d[i]=n%10
        i+=1
        n=n//10
    num=""
    if d[4]!=0:
        if(d[4]==1):
            num+=tens[d[3]]+"Thousand"
        else:
            num+=nty[d[4]]+""+number[d[3]]+"Thousand"
    else:
        if d[3]!=0:
            num+=number[d[3]]+"Thousand"
    if d[2]!=0:
        num+=number[d[2]]+"Hundred"
    if d[1]!=0:
        if(d[1]==1):
            num+=tens[d[0]]
        else:
```

```
        num+=nty[d[1]]+""+number[d[0]]
    else:
        if[d[0]]!=0:
            num+=number[d[0]]
    print(num)
```

OUTPUT:

```
Enter any number:89
Eighty-Nine
```

Exercise -5 Python Sequences

1. **Write a python program to create a function that will return another string similar to the input string, but with its case inverted. For example, input of “Mr.Ed” will result in “mR.eD” as the output string**

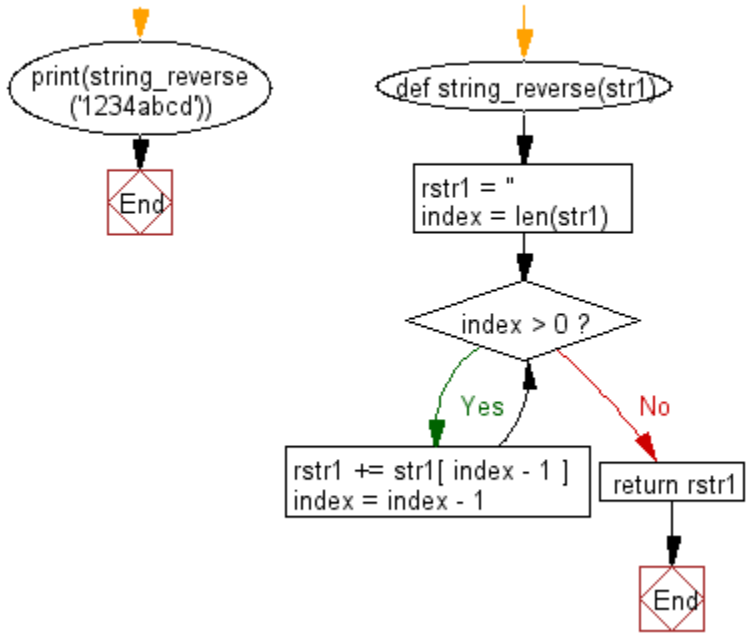
AIM:

To write a python program to create a function that will return another string similar to the input string, but with its case inverted.

ALGORITHM:

Step1: Start
Step2: Give the operands either string or integers
Step3: Select the operands either upper case, lower case, swap case, title
Step4: Stop

FLOW CHART:



SOURCE CODE:

```
str1=input('enter the string:')

print("1.upper 2.lower 3.swapcase 4.title")

n=int(input('select the operation:'))

if n==1:
```

```
    print(str1.upper())
elif n==2:
    print(str1.lower())
elif n==3:
    print(str1.swapcase())
elif n==4:
    print(str1.title())
else:
    print('select the correct option')
```

OUTPUT:

```
enter the string:Mr.Ed
1.upper 2.lower 3.swapcase 4.title
select the operation:3
MR.ED
```

2. **Write a program to take a string and append a backward copy of that string, making a palindrome**

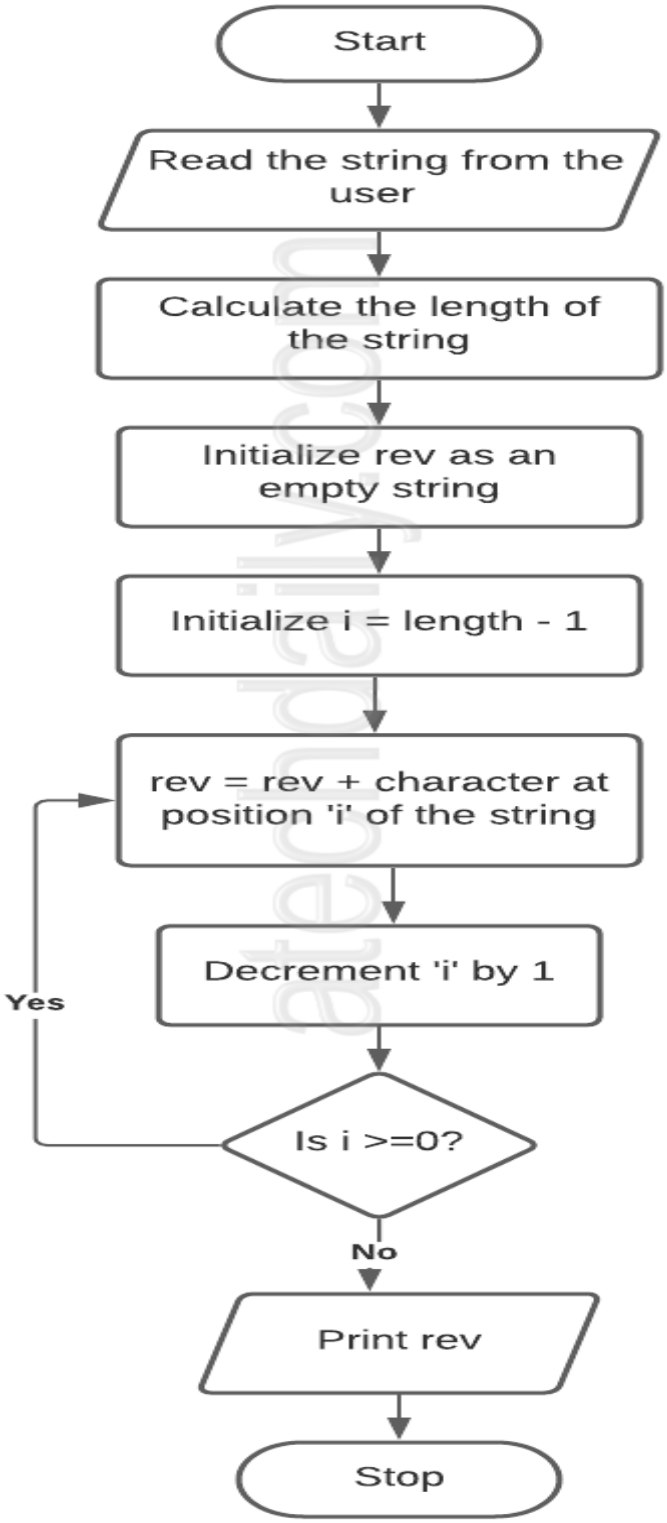
AIM:

To write a python program to take a string and append a backward copy of that string making a palindrome.

ALGORITHM:

Step 1: Start
Step 2: Read the string from the user
Step 3: Calculate the length of the string
Step 4: Initialize rev = " " [empty string]
Step 5: Initialize i = length - 1
Step 6: Repeat until i>=0:
6.1: rev = rev + Character at position 'i' of the string
6.2: i = i – 1
Step 7: Print rev
Step 8: Stop

FLOW CHART:


Start
Read the string from the user
Calculate the length of the string
Initialize rev as an empty string
Initialize i = length - 1
rev = rev + character at position 'i' of the string
Decrement 'i' by 1
Yes
Is i >=0?
No
Print rev
Stop

SOURCE CODE:

```
number=int(input('enter the string:'))
string=str(number)
rev_string=string[::-1]
print('reversed string:',rev_string)
if string==rev_string:
    print('string is a palindrome')
else:
    print('string is not a palindrome')
```

OUTPUT:

```
enter the string:1231
reversed string: 1321
string is not a palindrome
(or)
enter the string:1331
reversed string: 1331
string is a palindrome
```

Exercise -6 Python Sequences

1. **Write a program to create dictionary and display its key alphabetically**

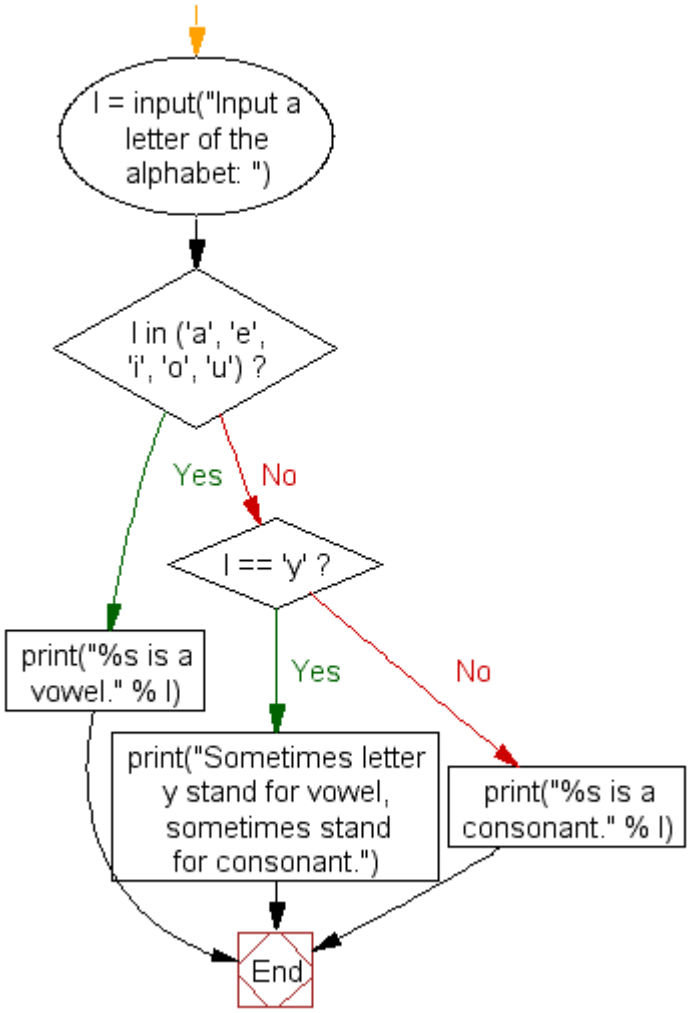
AIM:

To write a python program to create dictionary and display its keys alphabetically

ALGORITHM:

Step1: Start
Step2: Create the dictionary values
Step3: Sorted the values of the dictionary items
Step4: Sorted the values
Step5: Stop

FLOW CHART:



SOURCE CODE:

```
d = {2: 3, 1: 89, 4: 5, 3: 0}

od = sorted(d.items())

print(od)
```

OUTPUT:

```
[(1, 89), (2, 3), (3, 0), (4, 5)]
```

2. **Write a program to take a dictionary as input and return one as output, but the values are the keys and vice versa**

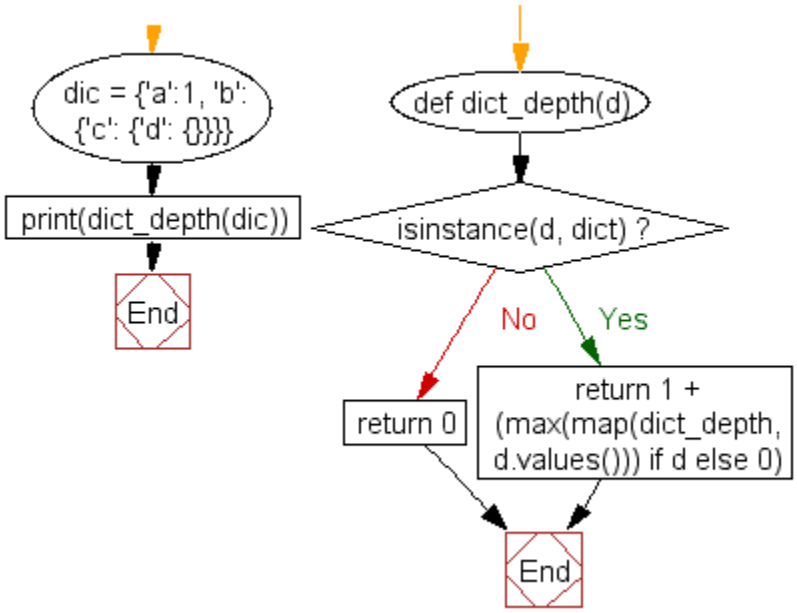
AIM:

To write a program to take a dictionary as input and return one as output, but the values are the keys and vice versa.

ALGORITHM:

Step1: Start
Step2: Create the dictionary values
Step3: Sorted the values of the dictionary items
Step4: Return the dictionary values as input and return the value as one input vice versa
Step5: Stop

FLOW CHART:



SOURCE CODE:

```
n=int(input("Input a number:"))
d=dict()
for x in range(1,n+1):
    d[x]=x*x
print(d)
```

OUTPUT:

```
Input a number:25
```

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225, 16: 256, 17: 289, 18: 324, 19: 361, 20: 400, 21: 441, 22: 484, 23: 529, 24: 576, 25: 625}

Exercise 7 – Files

1. **Write a program to compare two text files. If they are different, give the line and column numbers in the files where the first difference occurs**

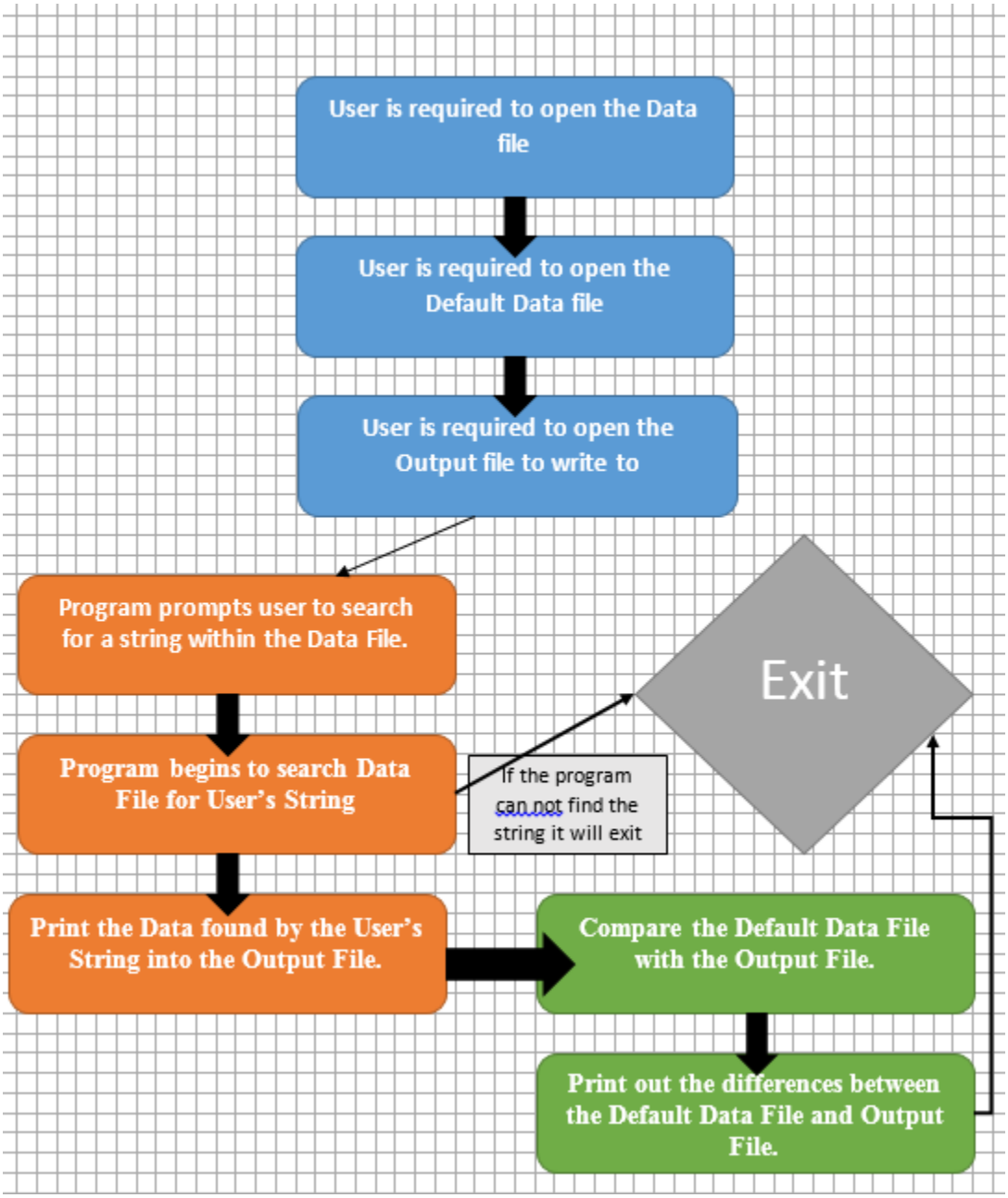
AIM:

To Write a Python program to compare two text files. If they are different, give the line and column numbers in the files where the first difference occurs.

ALGORITHM:

Step1: Start
Step2: Take the text file and write whatever user wants the data
Step3: User is required to open the file
Step4: Compare the data from one file to another file.
Step5: Print the data step by step
Step6: Print the same data as the user wants
Step7: Stop

FLOW CHART:



SOURCE CODE:

```
f1=open("file1.txt","r")
f2=open("file2.txt","r")
for line1 in f1:
    for line2 in f2:
        if line1==line2:
            print("SAME\n")
```

```
        else:
            print(line1+line2)
        break
f1.close()
f2.close()
```

My file1.txt

1. for
2. foo
3. in
4. bar
5. print
6. foo

My file2.txt

1. while
2. foo
3. <
4. bar
5. print
6. foo

OUTPUT:

1. for
2. while
3.
4. SAME
5.
6. in
7. <
8.
9. SAME
10.
11. SAME
12.
13. SAME

2. **Write a program to compute the number of characters, words and lines in a file**

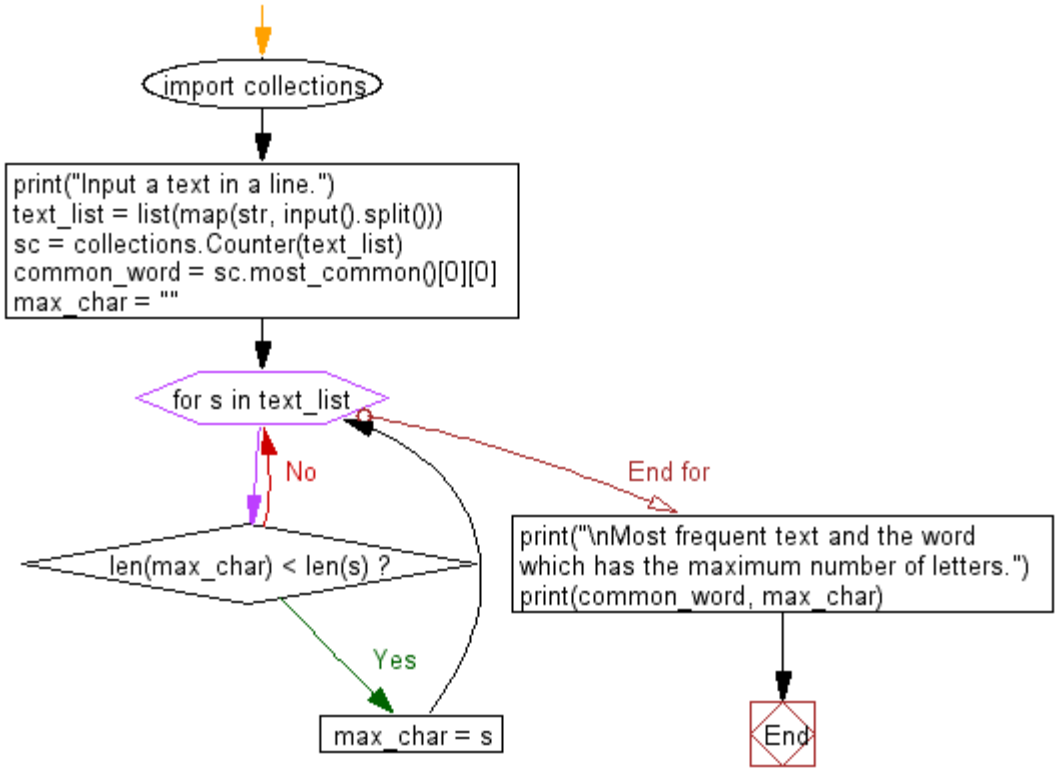
AIM:

To write a Python program that to compute the number of characters, words and lines in a file.

ALGORITHM:

Step1: Start
Step2: Open the text file and write the file whatever the user wants
Step3: Check whatever the lines, words, characters are to be count
Step4: Print the numbers of lines, words, characters that the user has to be given as input
Step5: Stop

FLOW CHART:



SOURCE CODE:

```
file = open("sample.txt", "r")

number_of_lines = 0

number_of_words = 0

number_of_characters = 0

for line in file:
```

```
  line = line.strip("\n")
won't count \n as character
  words = line.split()
  number_of_lines += 1
  number_of_words += len(words)
  number_of_characters += len(line)
file.close()
print("lines:", number_of_lines, "words:", number_of_words, "characters:",
number_of_characters)
```

OUTPUT:

```
lines: 3 words: 5 characters: 29
```

Exercise -8 Files

1. **Write a function ball collide that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding.**

   Hint: Represent a ball on a plane tuple of (x,y,r), r being the radius

   If (distance between two balls centres) <= (sum of radii) then (they are colliding)

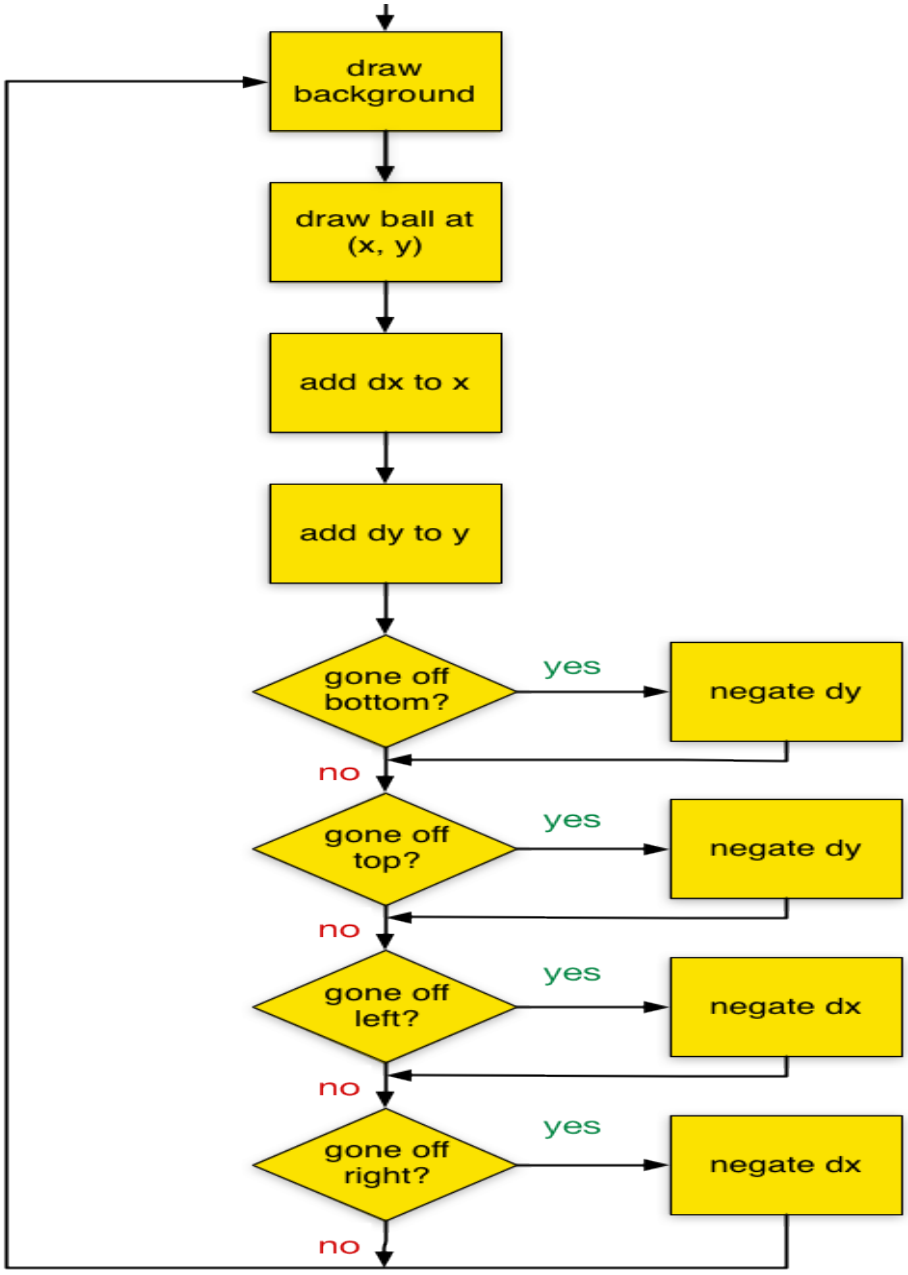
AIM:

To Write a Python function ball collide that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding.

ALGORITHM:

Step1: Start
Step2: Import math module
Step3: Read the values of x1,y1,r1,x2, y2and r2
Step4: Calculate the distance using the formulae **math.sqrt((x2 - x1)**2 + (y2 - y1)**2)** and store the result in distance
Step5: Print distance
Step6: Stop.

FLOW CHART:



SOURCE CODE:

```
import math
def ball_collide(x1,y1,r1,x2,y2,r2):
    dist=math.sqrt((x2-x1)**2+(y2-y1)**2);
    print("Distance b/w two balls:",dist)
    center=dist/2;
    print("Collisison point", center);
```

```
    r=r1+r2;
    print("Sum of radius",r)
    if(center<=r):
        print("They are Colliding")
        return True;
    else:
        print("Not Colliding")
        return False;
c= ball_collide(4,4,3,2,2,3)
print(c)
c= ball_collide(100,200,20,200,100,10)
print(c)
```

OUTPUT:

```
Distance b/w two balls: 2.8284271247461903
Collisison point 1.4142135623730951
Sum of radius 6
They are Colliding
True
Distance b/w two balls: 141.4213562373095
Collisison point 70.71067811865476
Sum of radius 30
Not Colliding
False
```

2. Find mean, median, mode for the given set of numbers in a list

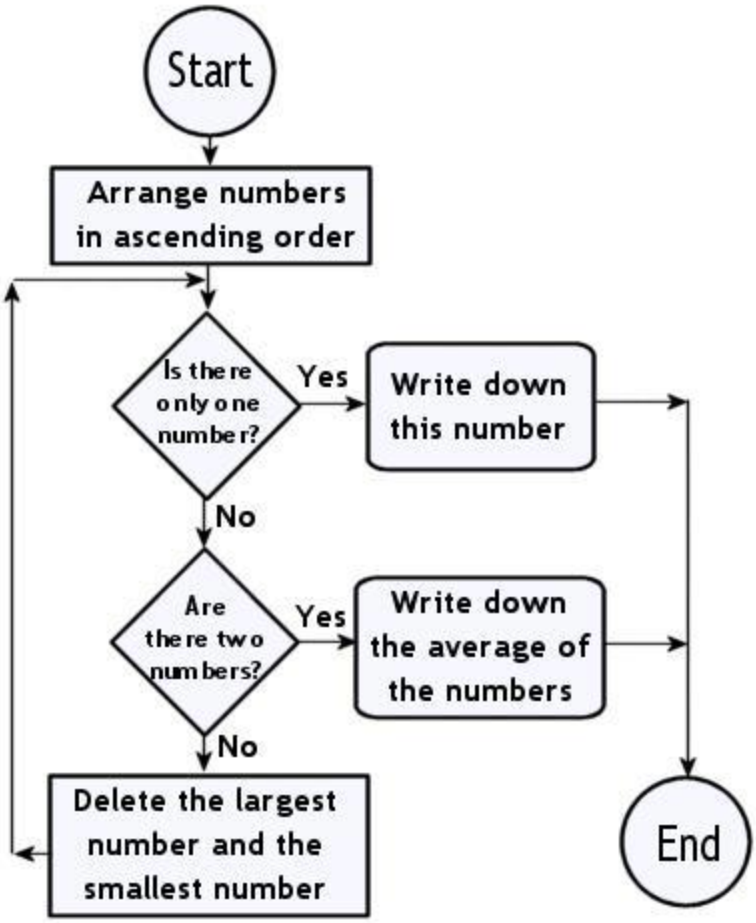
AIM:

To write a Python program that to find mean, median, mode for the given set of numbers in a list.

ALGORITHM:

Step1: Start
Step2: Given the list of numbers as the user wants
Step3:Print Mean, Median and Mode
Step4: Stop

FLOW CHART:



SOURCE CODE:

```
from statistics import mean, median, mode

i=[15,18,2,36,12,78,5,6,9,18]

print("Mean",mean(i))

print("Median",median(i))
```

```
print("Mode",mode(i))
```

OUTPUT:

```
Mean 19.9
Median 13.5
Mode 18
```

3. **Write a simple functions max2() and min2() that take two items and return the larger and smaller item, respectively. They should work arbitrary Python objects. For example max2(4,8) and min2(4,8) would each return 8 and 4 respectively**

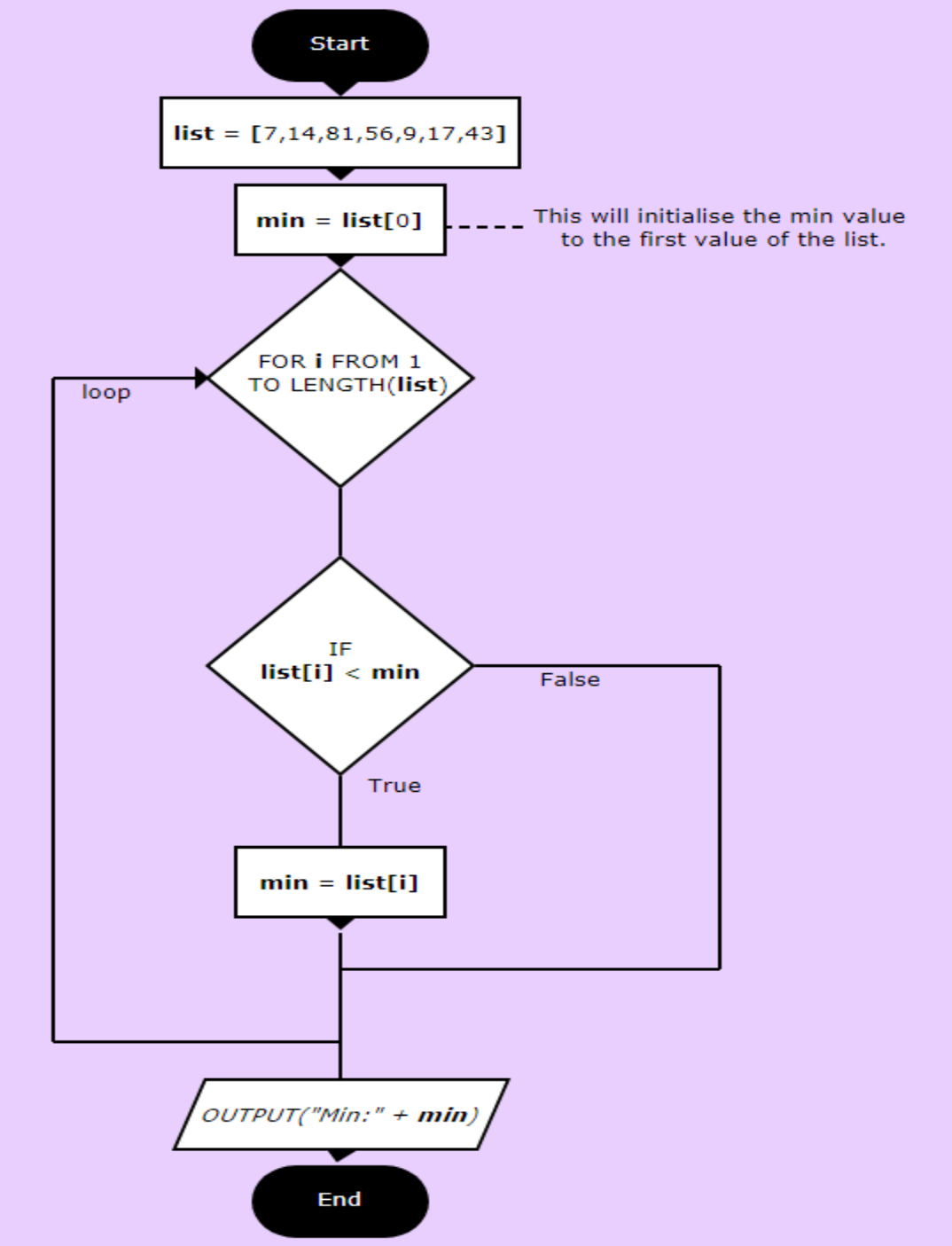
AIM:

To write a simple Python program functions max2() and min2() that take two items and return the larger and smaller item, respectively**.**

ALGORITHM:

Step1: Start
Step2: Give the values the user wants
Step3: Select the max and min values
Step4: Print max and min values
Step5: Stop

FLOW CHART


Start
list = [7,14,81,56,9,17,43]
min = list[0]
This will initialise the min value to the first value of the list.
FOR i FROM 1 TO LENGTH(list)
loop
IF list[i] < min
False
True
min = list[i]
OUTPUT("Min:" + min)
End

SOURCE CODE:

```
from operator import lt
def max2(num1,num2):
    ele=lt(num1,num2)
    if ele==True:
        return num2
    else:
        return num1
    def min2(num1,num2):
        ele=lt(num1,num2)
        if ele==True:
            return num1
        else:
            return num2
```

OUTPUT:

4,8

max value:8

min value:4

Exercise -9 Continued

1. **Write a function nearly equal to test whether two strings are nearly equal. Two strings a and b are nearly equal when a can be generated by a single mutation on b**

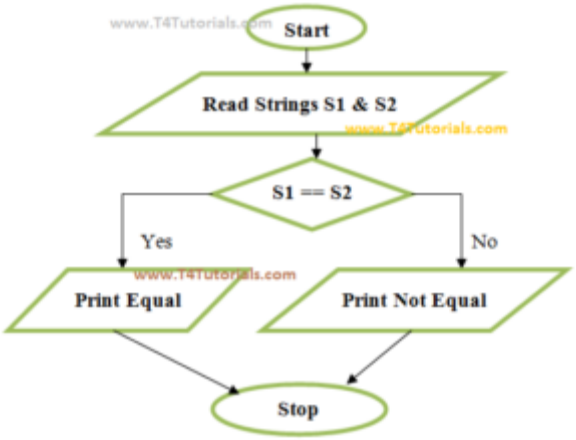
AIM:

To write a python function nearly equal to test whether two strings are nearly equal that two strings a and b are nearly equal when a can be generated by a single mutation on b

ALGORITHM:

Step1: Start
Step2: To compare two strings
Step3: To compare two strings with form values enter by user
Step4: To compare two strings with the database.
Step5: Stop

FLOW CHART:



SOURCE CODE:

```
def nearly_equal(str1,str2):
    count=0
    i=j=0
    while(i<len(str1) and j<len(str2)):
        if(str1[i]!=str2[j]):
            count=count+1
            if(len(str1)>len(str2)):
                i=i+1
            elif(len(str1)==len(str2)):
```

```
            pass
        else:
            i=i-1
    if(count>1):
        return False
    i=i+1
    j=j+1
  if(count<2):
        return True

str1=input("Enter first string::\n")
str2=input("Enter second string::\n")
boolean=nearly_equal(str1,str2)
if(boolean):
  print("Strings are nearly equal.")
else:
  print("Strings are not equal.")
```

OUTPUT:

```
Enter first string::
RISE
Enter second string::
RISEE
Strings are nearly equal.
```

2. Write a function to find dups to find all the duplicates in the list

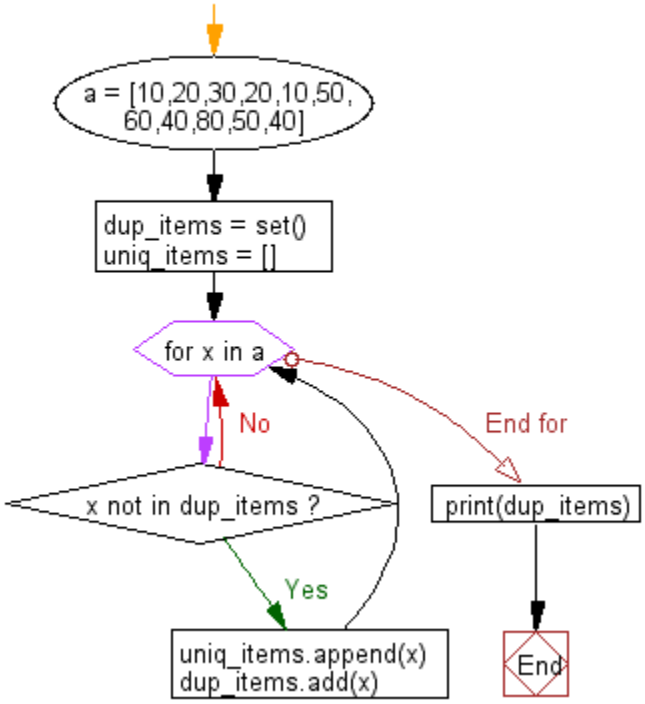
AIM:

To write a python function to find dups to find all the duplicates in the list

ALGORITHM:

Step1: Start
Step2: Enter the Integers whatever the user wants
Step3: Enter the Duplicate elements for the list
Step4: Select the dups in the duplicate elements
Step5: To find the duplicate elements in the list
Step6: Print the dups in the duplicate elements
Step7: Stop

FLOW CHART:



SOURCE CODE:

```
def dups(numlist):
    duplicates={}
    for ele in numlist:
        c=numlist.count(ele)
        if(c>=2):
            duplicates[ele]=c
```

```
    print("Duplicate elements are:\n",duplicates)
    return
numlist=[]
n=int(input("Enter number of elements to be insert:\n"))
for i in range(n):
    ele=int(input("Enter element"))
    numlist.append(ele)
dups(numlist)
```

OUTPUT:

```
Enter number of elements to be insert:
5
Enter element2
Enter element4
Enter element2
Enter element3
Enter element3
Duplicate elements are:
 {2: 2, 3: 2}
```

3. Write a function unique to find all the unique elements in a list

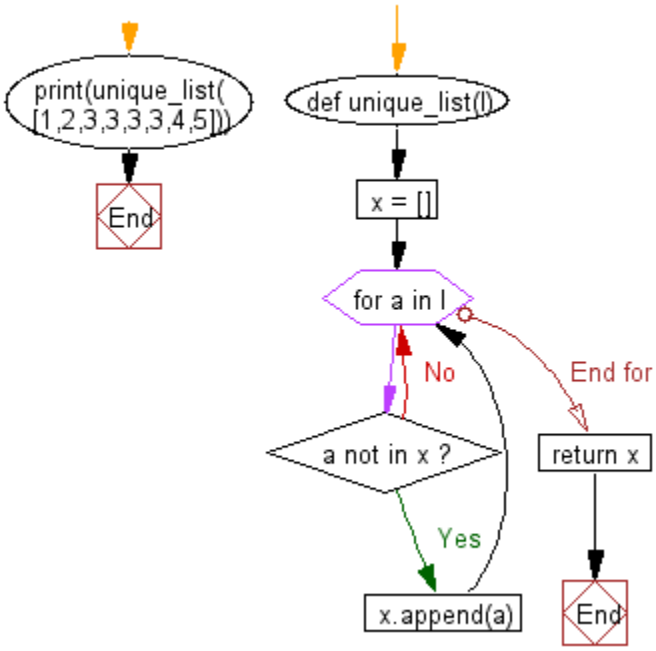
AIM:

To write a python function unique to find all the unique elements in a list

ALGORITHM:

Step1: Start
Step2: Enter the unique elements in the list
Step3: Define unique elements in the list
Step4: Append the unique elements in the list
Step5: Print the unique elements in the list
Step6: Stop

FLOW CHART:



SOURCE CODE:

```
def unique(numlist):
    uniqueele=[]
    for ele in numlist:
        c=numlist.count(ele)
        if(c==1):
            uniqueele.append(ele)
    print("Unique elements are:\n",uniqueele)
    return


numlist=[]
```

```
n=int(input("Enter number of elements to be insert:\n"))
for i in range(n):
    ele=int(input("Enter element"))
    numlist.append(ele)
unique(numlist)
```

OUTPUT:

```
Enter number of elements to be insert:
5
Enter element2
Enter element3
Enter element4
Enter element3
Enter element5
Unique elements are:
 [2, 4, 5]
```

Exercise -10 Functions – Problem Solving

1. **Write a function to cumulative_product to compute cumulative product of a list of numbers**

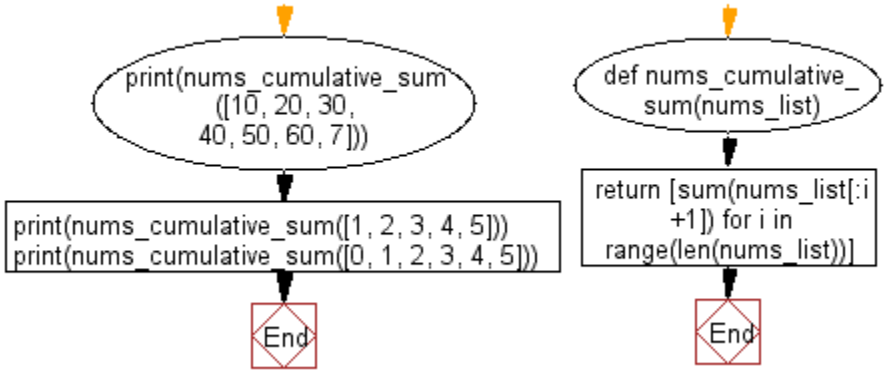
AIM:

To write a python function to cumulative_product to compute cumulative product of a list of numbers.

ALGORITH:

Step1: Start
Step2: Enter the elements whatever the user wants
Step3: Cumulative product of an elements
Step4: Append the elements
Step5: Print the cumulative_product of the elements
Step6: Stop

FLOW CHART:



SOURCE CODE:

```
def cumulative_product(numlist):
    product=1
    cp=[]
    for ele in numlist:
        product=product*ele
        cp.append(product)
    return cp


numlist=[]
n=int(input("Enter number of elements to be insert:"))
for i in range(n):
    ele=int(input("Enter element:"))
```

```
    numlist.append(ele)
cp=cumulative_product(numlist)
print("Cumulative product of list elements is ",cp)
```

OUTPUT:

```
Enter number of elements to be insert:4
Enter element:3
Enter element:2
Enter element:4
Enter element:1
Cumulative product of list elements is  [3, 6, 24, 24]
```

2. Write a function reverse to reverse a list. Without using the reverse function

AIM:

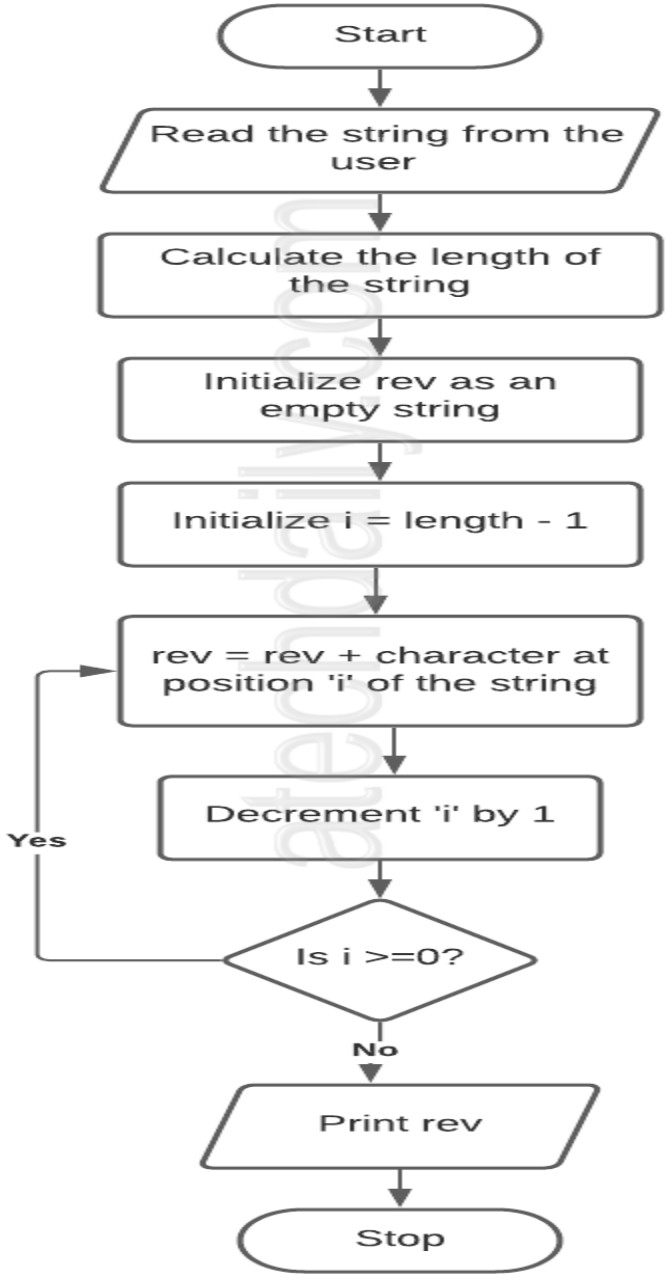
To write a python function that reverse to reverse a list without using the reverse function.

ALGORITHM:

Step 1: Start
Step 2: Read the string from the user

Step 3: Calculate the length of the string
Step 4: Initialize rev = " " [empty string]
Step 5: Initialize i = length - 1
Step 6: Repeat until i>=0:
6.1: rev = rev + Character at position 'i' of the string
6.2: i = i – 1
Step 7: Print rev
Step 9: Stop

FLOW CHART:



SOURCE CODE:

```
def reverse(numlist):
    i=0
    j=len(numlist)-1
    while(i<=j):
        temp=numlist[i]
        numlist[i]=numlist[j]
```

```
        numlist[j]=temp
        i=i+1
        j=j-1
    return
numlist=[]
n=int(input("Enter number of elements to be insert:"))
for i in range(n):
    ele=int(input("Enter element:"))
    numlist.append(ele)
reverse(numlist)
print("After reverse list elements are:\n",numlist)
```

OUTPUT:

```
Enter number of elements to be insert:4
Enter element:2
Enter element:3
Enter element:1
Enter element:5
After reverse list elements are:
 [5, 1, 3, 2]
```

3. **Write function to compute gcd, lcm of two numbers. Each function shouldn’t exceed one line**

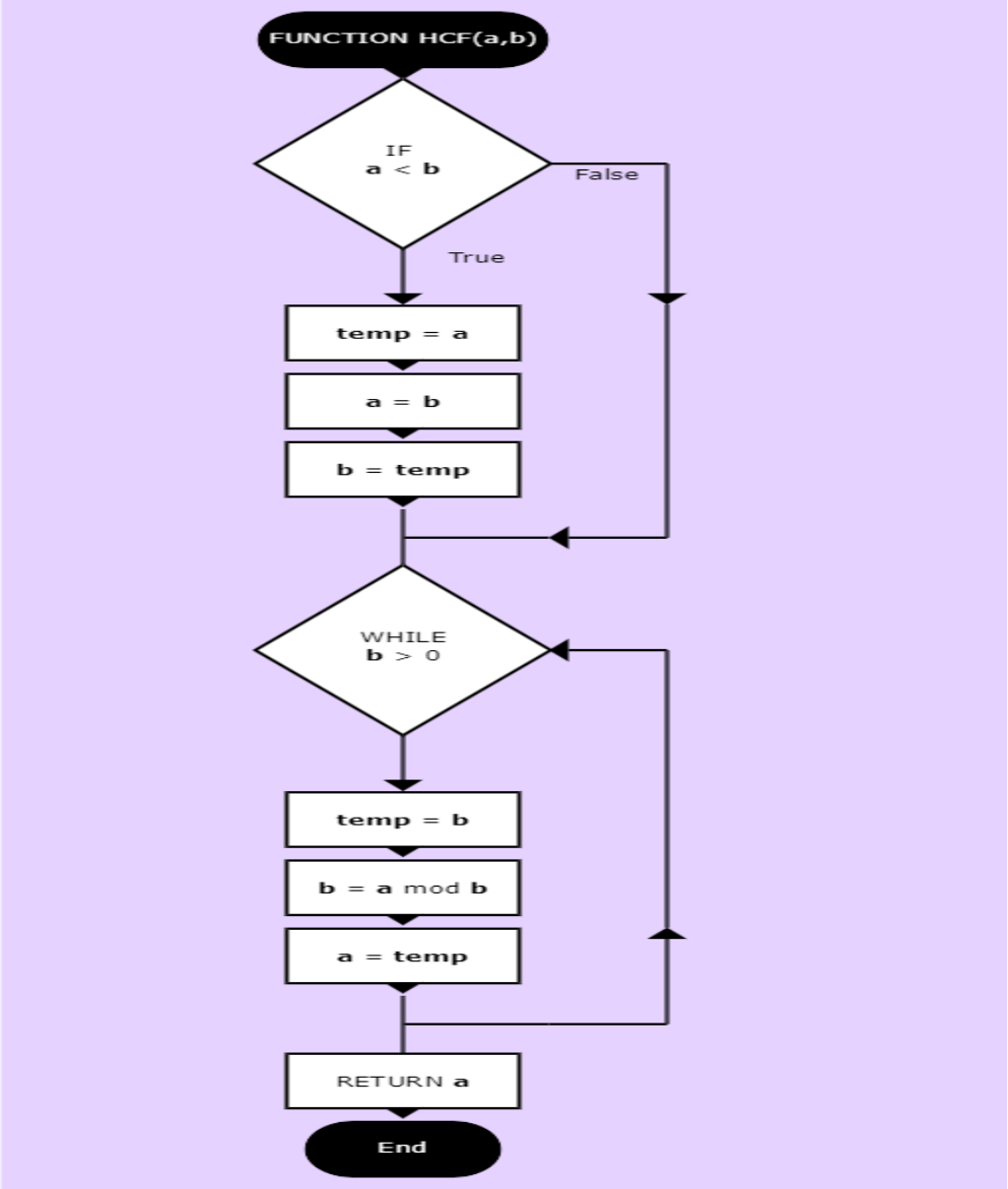
AIM:

To write a python function that to compute gcd, lcm of two numbers. Each function shouldn’t exceed one line.

ALGORITHM:

Step1: Start
Step2: Enter the numbers of GCD elements
Step3: Enter the numbers of LCM elements
Step4: Print GCD of two numbers
Step5: Print LCM of two numbers
Step6: Stop

FLOW CHART:



SOURCE CODE:

```
gcd=lambda a,b: a if b==0 else gcd(b,a%b)
lcm=lambda a,b: (a*b)/gcd(a,b)
num1=int(input("Enter first number:"))
num2=int(input("Enter second number:"))
```

```
print("LCM of %d and %d is %d"%(num1,num2,lcm(num1,num2)))
print("GCD of %d and %d is %d"%(num1,num2,gcd(num1,num2)))
```

OUTPUT:

```
Enter first number:8
Enter second number:24
LCM of 8 and 24 is 24
GCD of 8 and 24 is 8
```

(or)

SOURCE CODE:

```
n1 = int(input("Enter First number :"))
n2 = int(input("Enter Second number :"))
x = n1
y = n2
while(n2!=0):
    t = n2
    n2 = n1 % n2
    n1 = t
gcd = n1
print("GCD of {0} and {1} = {2}".format(x,y,gcd))
lcm = (x*y)/gcd
print("LCM of {0} and {1} = {2}".format(x,y,lcm))
```

OUTPUT:

```
Enter First number :8
Enter Second number :24
GCD of 8 and 24 = 8
LCM of 8 and 24 = 24.0
```

Exercise – 11 GUI, Graphics

1. **Write a GUI for an Expression Calculator usingtk**

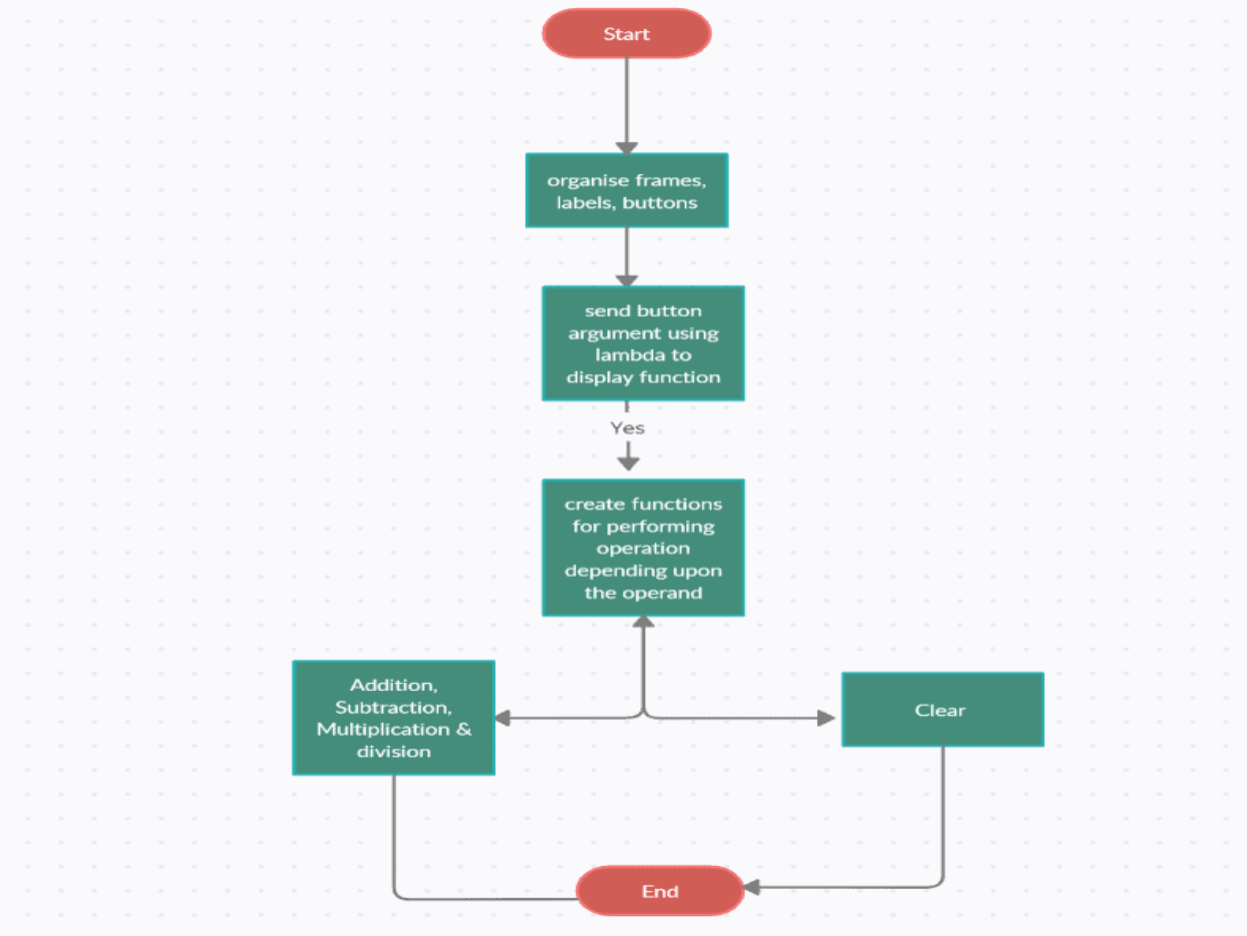
AIM:

To write a Python program for GUI for an Expression Calculator usingtk

ALGORITHM:

Step1: Start
Step2: Every time application will open on the bottom right side.
Step3: The application will have a grey background in the display & black background in the buttons.
Step4: The text color of the entire application will be white
Step5: A window will have a fixed size & can't be resized.
Step6: The calculator will have the basic functionality of addition, subtraction, multiplication & division.
Step7: Numbers will be displayed on the right side of the screen
Step8: Stop

FLOW CHART:



SOURCE CODE:

```
# Python program to create a simple GUI
# calculator using Tkinter
# import everything from tkinter module
from tkinter import *
# globally declare the expression variable
expression = ""
# Function to update expression
# in the text entry box
def press(num):
        # point out the global expression variable
        global expression
```

```
        # concatenation of string
        expression = expression + str(num)
        # update the expression by using set method
        equation.set(expression)
# Function to evaluate the final expression
def equalpress():
        # Try and except statement is used
        # for handling the errors like zero
        # division error etc.

        # Put that code inside the try block
        # which may generate the error
        try:
                global expression
                # eval function evaluate the expression
                # and str function convert the result
                # into string
                total = str(eval(expression))
                equation.set(total)
                # initialize the expression variable
                # by empty string
                expression = ""
        # if error is generate then handle
        # by the except block
        except:
                equation.set(" error ")
                expression = ""
# Function to clear the contents
```

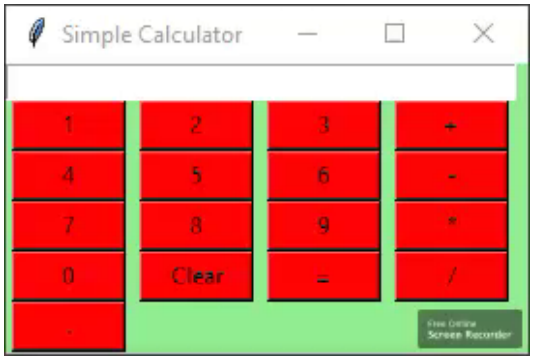
```
# of text entry box
def clear():
        global expression
        expression = ""
        equation.set("")
# Driver code
if __name__ == "__main__":
        # create a GUI window
        gui = Tk()

        # set the background colour of GUI window
        gui.configure(background="light green")
        # set the title of GUI window
        gui.title("Simple Calculator")
        # set the configuration of GUI window
        gui.geometry("270x150")
        # StringVar() is the variable class
        # we create an instance of this class
        equation = StringVar()
        # create the text entry box for
        # showing the expression .
        expression_field = Entry(gui, textvariable=equation)
        # grid method is used for placing
        # the widgets at respective positions
        # in table like structure .
        expression_field.grid(columnspan=4, ipadx=70)
        # create a Buttons and place at a particular
        # location inside the root window .
```

```
# when user press the button, the command or
# function affiliated to that button is executed .
button1 = Button(gui, text=' 1 ', fg='black', bg='red',
                                    command=lambda: press(1), height=1, width=7)
button1.grid(row=2, column=0)
button2 = Button(gui, text=' 2 ', fg='black', bg='red',
                                    command=lambda: press(2), height=1, width=7)
button2.grid(row=2, column=1)
button3 = Button(gui, text=' 3 ', fg='black', bg='red',
                                    command=lambda: press(3), height=1, width=7)
button3.grid(row=2, column=2)
button4 = Button(gui, text=' 4 ', fg='black', bg='red',
                                    command=lambda: press(4), height=1, width=7)
button4.grid(row=3, column=0)
button5 = Button(gui, text=' 5 ', fg='black', bg='red',
                                    command=lambda: press(5), height=1, width=7)
button5.grid(row=3, column=1)
button6 = Button(gui, text=' 6 ', fg='black', bg='red',
                                    command=lambda: press(6), height=1, width=7)
button6.grid(row=3, column=2)
button7 = Button(gui, text=' 7 ', fg='black', bg='red',
                                    command=lambda: press(7), height=1, width=7)
button7.grid(row=4, column=0)
button8 = Button(gui, text=' 8 ', fg='black', bg='red',
                                    command=lambda: press(8), height=1, width=7)
button8.grid(row=4, column=1)
button9 = Button(gui, text=' 9 ', fg='black', bg='red',
                                    command=lambda: press(9), height=1, width=7)
```

```
button9.grid(row=4, column=2)
button0 = Button(gui, text=' 0 ', fg='black', bg='red',
                                    command=lambda: press(0), height=1, width=7)
button0.grid(row=5, column=0)
plus = Button(gui, text=' + ', fg='black', bg='red',
                                command=lambda: press("+"), height=1, width=7)
plus.grid(row=2, column=3)
minus = Button(gui, text=' - ', fg='black', bg='red',
                                command=lambda: press("-"), height=1, width=7)
minus.grid(row=3, column=3)
multiply = Button(gui, text=' * ', fg='black', bg='red',
                                    command=lambda: press("*"), height=1, width=7)
multiply.grid(row=4, column=3)
divide = Button(gui, text=' / ', fg='black', bg='red',
                                    command=lambda: press("/"), height=1, width=7)
divide.grid(row=5, column=3)
equal = Button(gui, text=' = ', fg='black', bg='red',
                                command=equalpress, height=1, width=7)
equal.grid(row=5, column=2)
clear = Button(gui, text='Clear', fg='black', bg='red',
                                command=clear, height=1, width=7)
clear.grid(row=5, column='1')
Decimal= Button(gui, text='.', fg='black', bg='red',
                                    command=lambda: press('.'), height=1, width=7)
Decimal.grid(row=6, column=0)
# start the GUI
gui.mainloop()
```

Output:



2. Write a program to implement the following figures using turtle

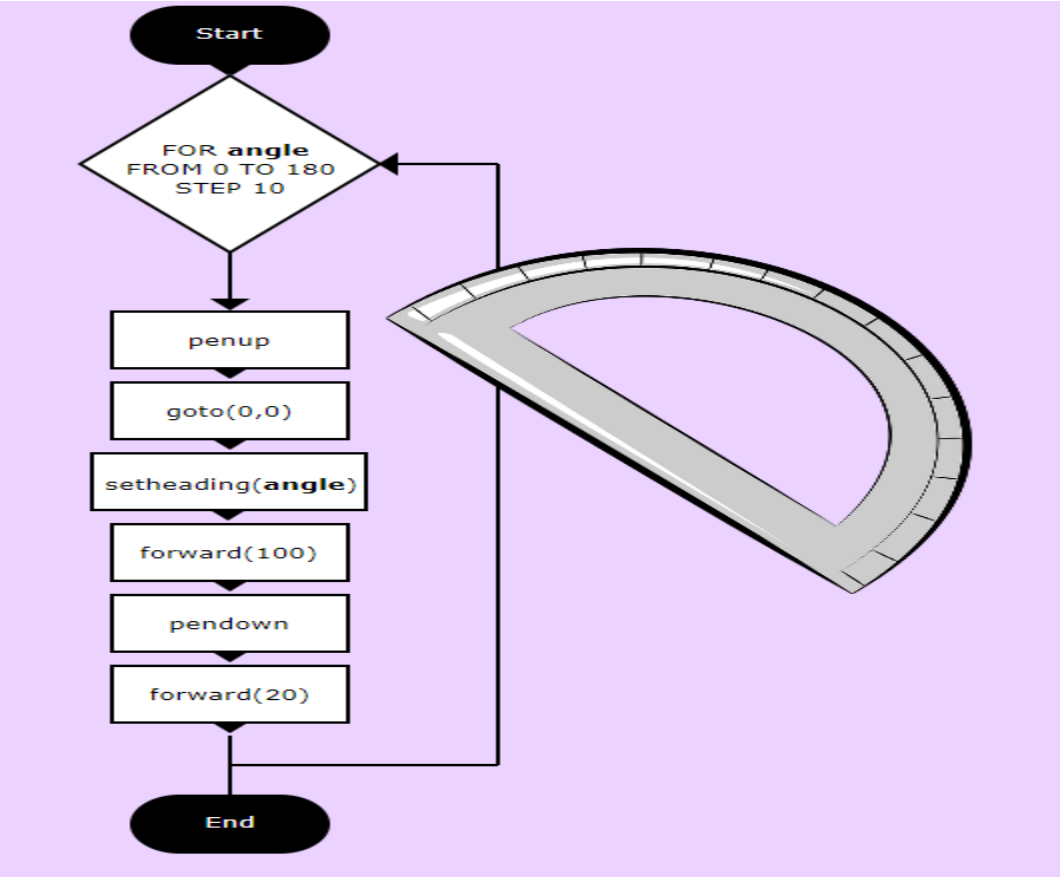
AIM:

To Write a Python program to implement the following figures using turtle.

ALGORITHM:

Step1: Start
Step2: Making an angle from 0 to 180
Step3: Penup
Step4: Goto (0,0)
Step5: Making an angle
Step6: Forward (100) of an turtle
Step7: Pendown
Ste8: Forward(20) of an turtle
Step9: Stop

FLOW CHART:



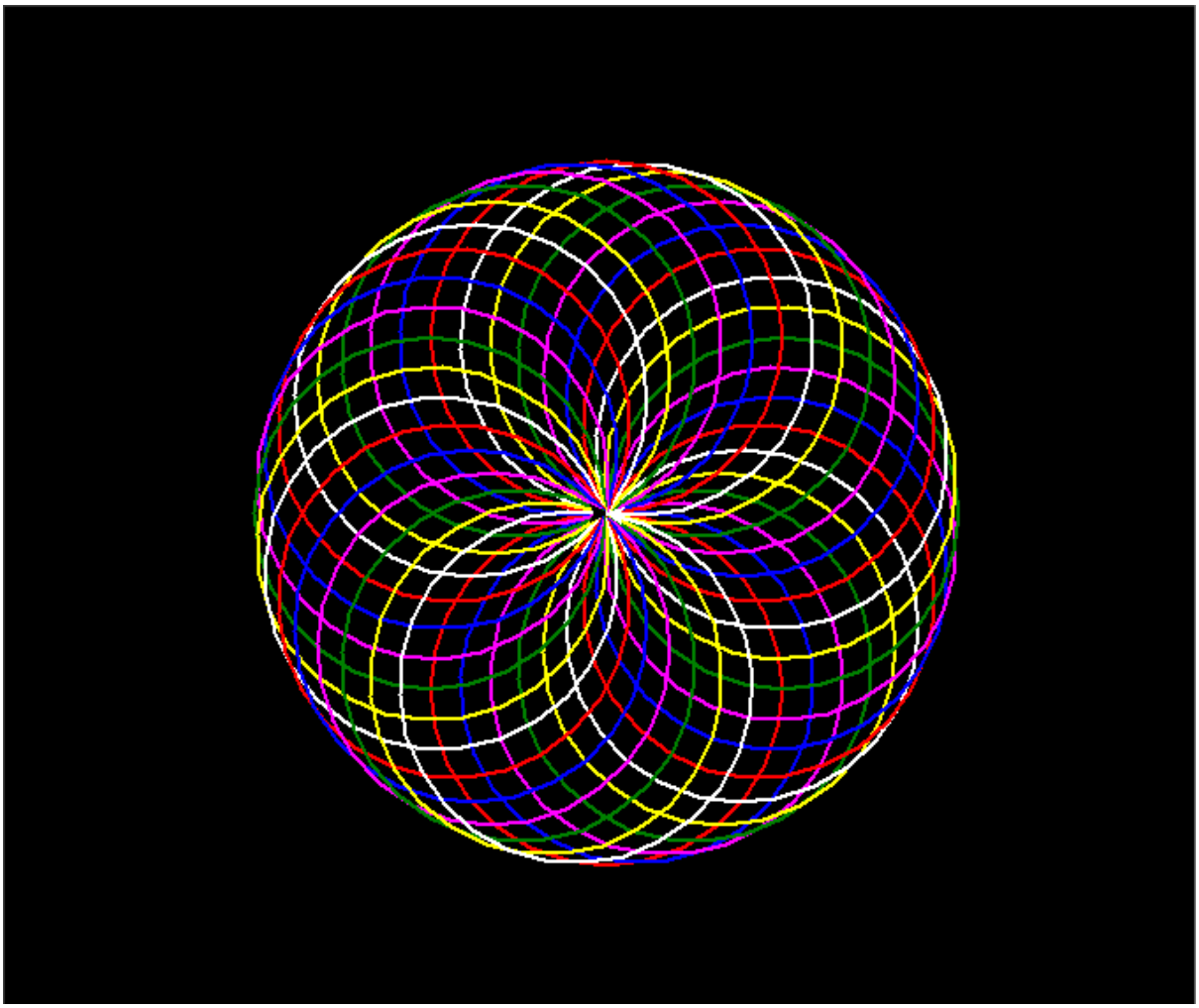
SOURSE CODE:

```
import turtle as tt

tt.bgcolor('black')
```

```
tt.pensize(2)
tt.speed(10)
for i in range(6):
        for color in ('red', 'magenta', 'blue',
                                'cyan', 'green', 'white',
                                'yellow'):
                tt.color(color)
                tt.circle(100)
                tt.left(10)
        tt.hideturtle()
```

OUTPUT:



Vibrate Circle Source Code:

```
import turtle
t = turtle.Turtle()
s = turtle.Screen()
s.bgcolor("black")
t.pencolor("red")
a = 0
b = 0
t.speed(0)
t.penup()
```

```
t.goto(0,200)
t.pendown()
while(True):
    t.forward(a)
    t.right(b)
    a+=3
    b+=1
    if b == 210:
        break
        t.hideturtle()

turtle.done()
```

OUTPUT: