



**NADAR SARASWATHI COLLEGE OF ENGINEERING &  
TECHNOLOGY**

**Vadapudupatti, Theni - 625531**

**DEPARTMENT OF SCIENCE AND HUMANITIES**

**Regulation 2021**

**GE3171**

**PROBLEM SOLVING AND PYTHON  
PROGRAMMING LAB MANUAL**

**I Year / I Sem**

**Prepared By**

**S.Abirami Kayathiri,  
AP/AI&DS**

## **VISION**

- ☐ Place for Technology Revolution

## **MISSION**

- ☐ Promote and undertake all – inclusive developments.
- ☐ Develop high quality technical education with academic excellence and innovative research with ethics.
- ☐ Create an atmosphere where teacher enjoys facilitation and learners(students) enjoy learning through foster innovation.
- ☐ Collaborate with industry and academic to meet the changing needs of society.

## **DEPARTMENT OF SCIENCE AND HUMANITIES**

## **VISION**

## **MISSION**

## **PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

1. To enrich the knowledge of students to get graduation in the field of computer science and engineering and pursue professional development in the software field or as entrepreneurs.
2. To participate in research and thrive in multi-disciplinary, systems oriented work environment in team work with professional ethics.
3. To contribute the solution to the complex problems as a professional engineer for society and strive to promote a practice of integrity, tolerance, leadership and respect in the work place.

## **PROGRAM OUTCOMES (POs)**

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **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.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **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 practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write

effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **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.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

#### **PROGRAM SPECIFIC OBJECTIVES (PSOs)**

1. To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.
2. To apply software engineering principles and practices for developing quality software for scientific and business applications.
3. To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

## SYLLABUS

GE3171      **PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY**      **L T P C**  
**0 0 4 2**

### COURSE OBJECTIVES:

- To understand the problem solving approaches.
- To learn the basic programming constructs in Python.
- To practice various computing strategies for Python-based solutions to real world problems.
- To use Python data structures - lists, tuples, dictionaries.
- To do input/output with files in Python.

### EXPERIMENTS:

**Note: The examples suggested in each experiment are only indicative. The lab instructor is expected to design other problems on similar lines. The Examination shall not be restricted to the sample experiments listed here.**

1. Identification and solving of simple real life or scientific or technical problems, and developing flow charts for the same. (Electricity Billing, Retail shop billing, Sin series, weight of a motorbike, Weight of a steel bar, compute Electrical Current in Three Phase AC Circuit, etc.)
2. Python programming using simple statements and expressions (exchange the values of two variables, circulate the values of n variables, distance between two points).
3. Scientific problems using Conditionals and Iterative loops. (Number series, Number Patterns, pyramid pattern)
4. Implementing real-time/technical applications using Lists, Tuples. (Items present in a library/Components of a car/ Materials required for construction of a building –operations of list & tuples)
5. Implementing real-time/technical applications using Sets, Dictionaries. (Language, components of an automobile, Elements of a civil structure, etc.- operations of Sets & Dictionaries)
6. Implementing programs using Functions. (Factorial, largest number in a list, area of shape)
7. Implementing programs using Strings. (reverse, palindrome, character count, replacing characters)
8. Implementing programs using written modules and Python Standard Libraries (pandas, numpy, Matplotlib, scipy)
9. Implementing real-time/technical applications using File handling. (copy from one file to another, word count, longest word)
10. Implementing real-time/technical applications using Exception handling. (divide by zero error, voter's age validity, student mark range validation)
11. Exploring Pygame tool.
12. Developing a game activity using Pygame like bouncing ball, car race etc.

### COURSE OUTCOMES:

On completion of the course, students will be able to:

- CO1: Develop algorithmic solutions to simple computational problems
- CO2: Develop and execute simple Python programs.
- CO3: Implement programs in Python using conditionals and loops for solving problems.
- CO4: Deploy functions to decompose a Python program.
- CO5: Process compound data using Python data structures.
- CO6: Utilize Python packages in developing software applications.

**TEXT BOOKS:**

1. Allen B. Downey, "Think Python: How to Think like a Computer Scientist", 2nd Edition, O'Reilly Publishers, 2016.
2. Karl Beecher, "Computational Thinking: A Beginner's Guide to Problem Solving and Programming", 1st Edition, BCS Learning & Development Limited, 2017.

**REFERENCES:**

1. Paul Deitel and Harvey Deitel, "Python for Programmers", Pearson Education, 1<sup>st</sup> Edition, 2021.
2. G Venkatesh and Madhavan Mukund, "Computational Thinking: A Primer for Programmers and Data Scientists", 1<sup>st</sup> Edition, Notion Press, 2021.
3. John V Guttag, "Introduction to Computation and Programming Using Python: With Applications to Computational Modeling and Understanding Data", Third Edition, MIT Press, 2021.
4. Eric Matthes, "Python Crash Course, A Hands - on Project Based Introduction to Programming", 2<sup>nd</sup> Edition, No Starch Press, 2019.
5. <https://www.python.org/>
6. Martin C. Brown, "Python: The Complete Reference", 4<sup>th</sup> Edition, Mc-Graw Hill, 2018.

**TOTAL: 60 PERIODS.**

<b>Ex No 1</b>	<b>Identification and solving of simple real life (or) scientific (or) technical problems and developing flow chart the same (Electricity Billing, Retail shop billing, Sin series, weight of a motorbike, Weight of a steel bar, compute Electrical Current in Three Phase AC Circuit, etc.)</b>
<b>Ex No 1) a</b>	<b>Flow chart for electricity billing</b>

**Aim:**

To develop the flowchart to calculate the electricity bill.

**Algorithm:**

**Step1:** Start

**Step2:** Get start reading and end reading from the user.

**Step3:** Subtract start reading from end reading to get the number of units consumed by the user.

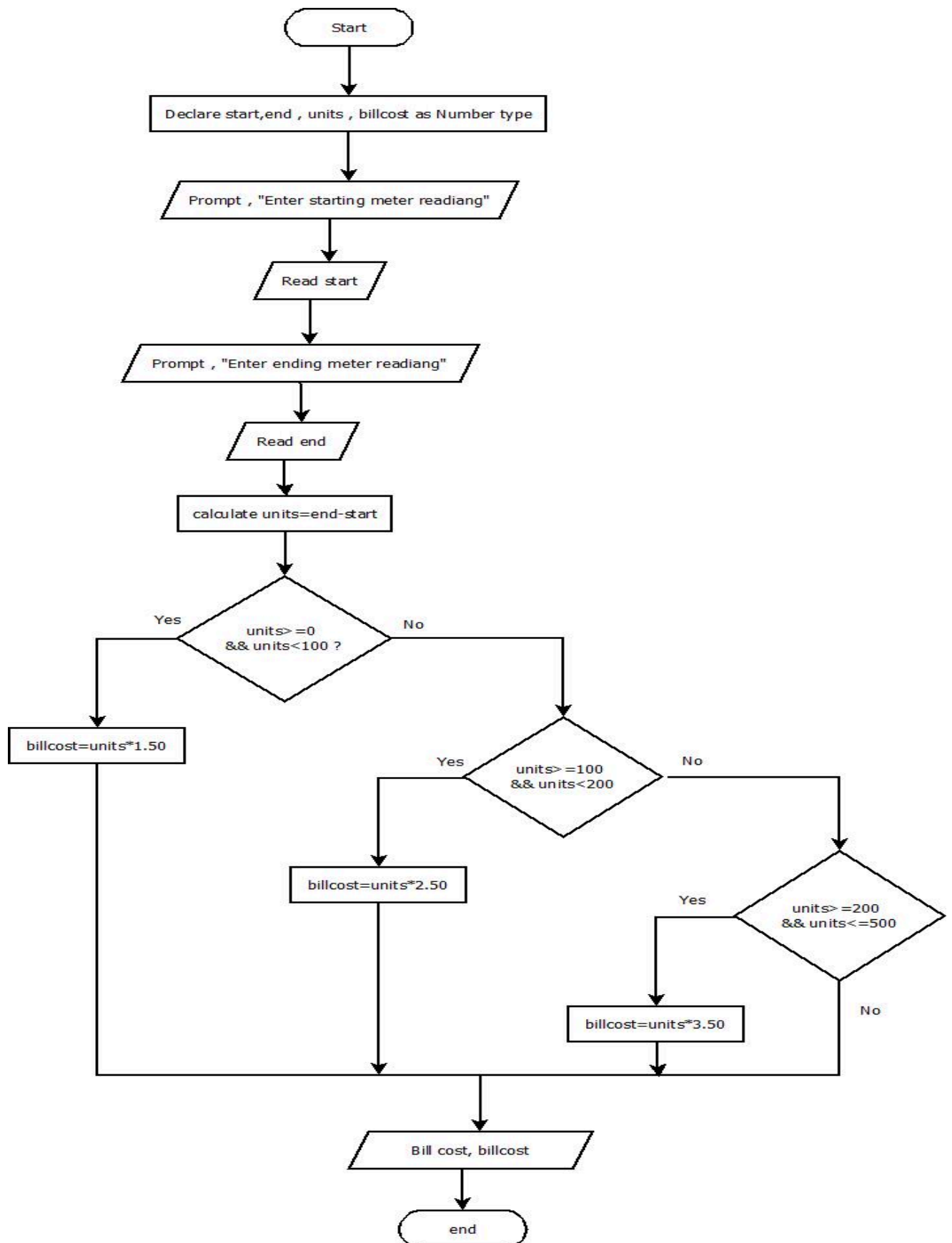
**Step4:** If the units consumed are greater than zero and less than or equal to 100, then multiply the units with Rs.1.50 to get the bill cost.

**Step5:** If the units consumed are greater than 100 and less than or equal to 200, then multiply the units with Rs.2.50 to get the bill cost.

**Step6:** If the units consumed are greater than 200 and less than or equal to 300, then multiply the units with Rs.3.50 to get the bill cost.

**Step7:** Print the bill cost.

**Step8:** Stop.



**Result:**

Thus the flowchart for calculating electricity bill has been developed successfully.



**Aim:**

To develop the flowchart for retail shop billing.

**Algorithm:**

**Step1:** Start

**Step2:** Get the number of items, quantity of items, price of items.

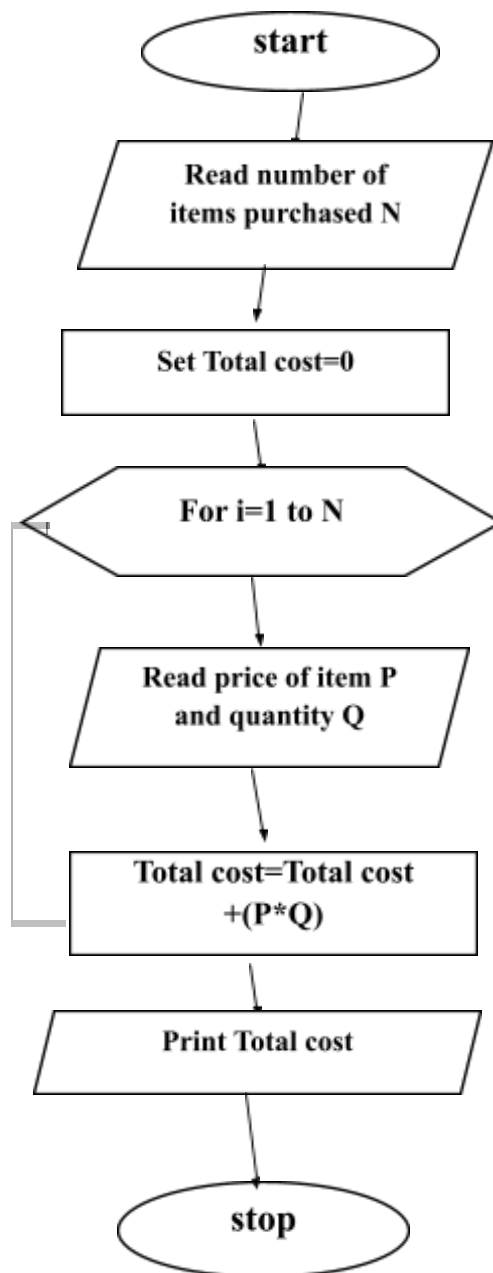
**Step3:** Set total cost as 0

**Step4:** Calculate the total cost by using this formula

$\text{total cost} = (\text{quantity of items} * \text{price of items}) + \text{total cost}.$

**Step5:** Print the total cost.

**Step6:** Stop.



**Result:**

Thus, the flowchart for retail shop billing has been developed successfully.

**Aim:**

To develop the flowchart for calculating sin series using Taylor's formula.

**Algorithm:**

**Step1:** Start

**Step2:** Read the value of x in degrees, no. of terms to be calculated from user.

**Step3:** Calculate sin series using Taylor's formula  $\sin(x) = x/1! - x^3/3! + x^5/5! - \dots$

**Step4:** Convert x into radian using the formula  $x * \pi / 180$ .

**Step5:** Assign x to temporary variable t.

**Step6:** Assign x to sum.

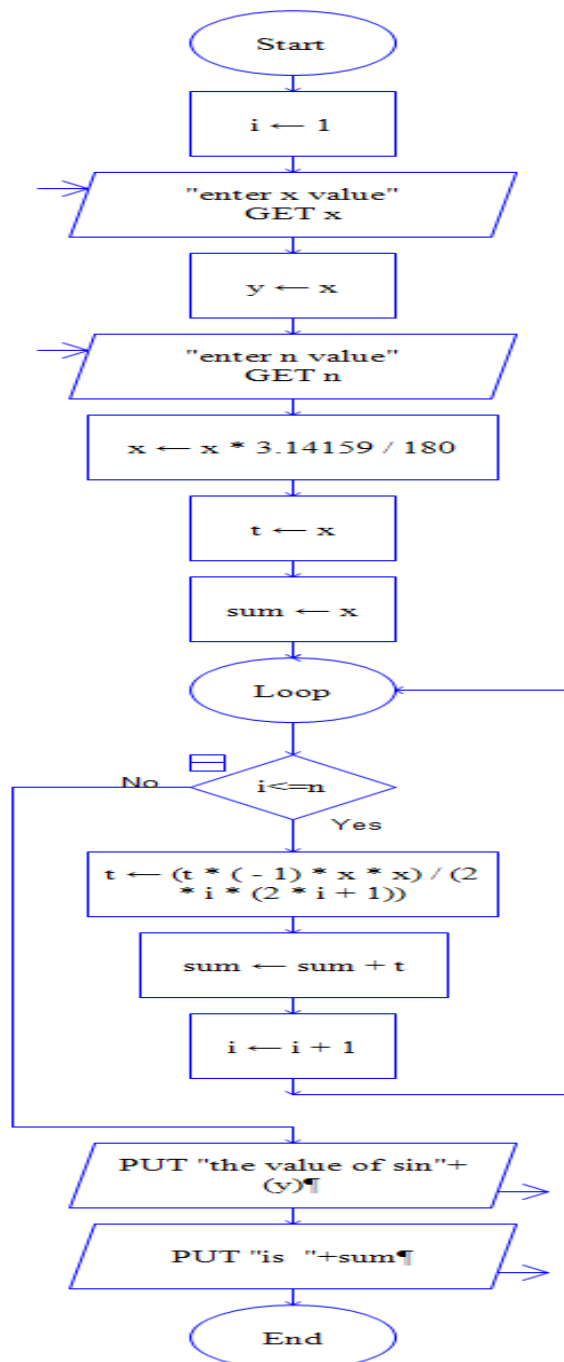
**Step7:** Initialize for loop with i= 1 to N

**Step8:** Calculate t using this formula  $t = t * \frac{(-1)^i * x^2}{2 * i * (2 * i + 1)}$

**Step9:** Calculate sum by using this formula  $\text{sum} = \text{sum} + t$

**Step10:** Print the value sum to the user

**Step11:** Stop



### Result:

Thus, the flowchart for calculating sin series using Taylor's formula has been developed successfully.

**Aim:**

To develop the flowchart for calculating the weight of steel bar.

**Algorithm:**

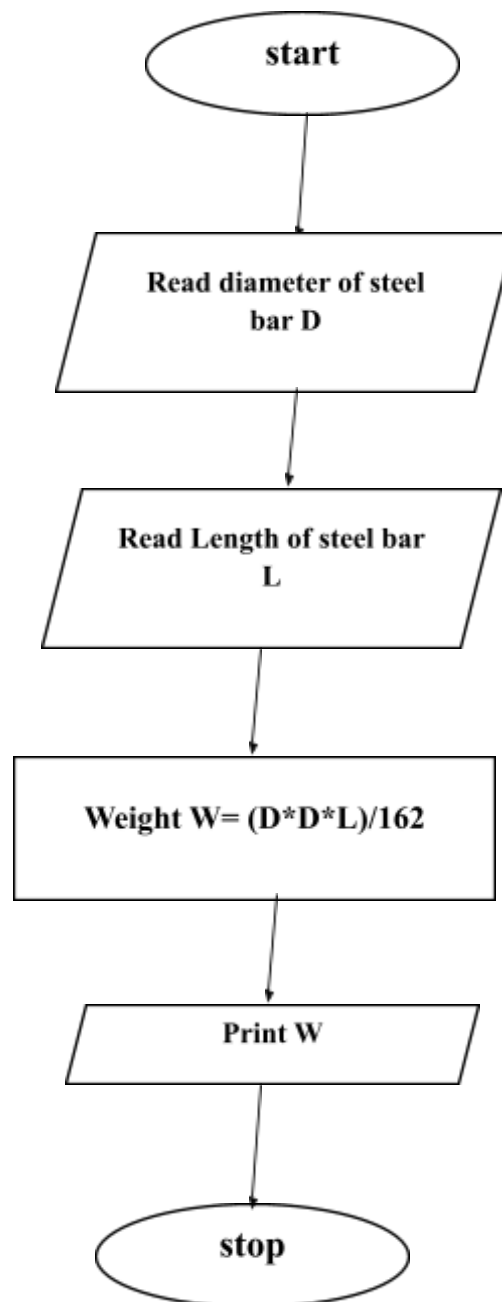
**Step1:** Start

**Step2:** Get the length and diameter from the user.

**Step3:** Calculate the weight of steel bar using this formula  $W = D^2L/162$

**Step4:** Print the weight of steel bar

**Step5:** Stop

**Result:**

Thus, the flowchart for calculating weight of steel bar has been developed successfully.

Ex No 1) e	Flowchart to compute electric current in three phase AC circuit
------------	---

**Aim:**

To develop the flowchart to compute electric current in three phase AC circuit.

**Algorithm:**

**Step1:** Start

**Step2:** Get the power, voltage, factor

**Step3:** Calculate the Ampere by using this formula  $\text{ampere} = (\text{KW} * 1000) / (\text{E} * 1.73 * \text{pf})$

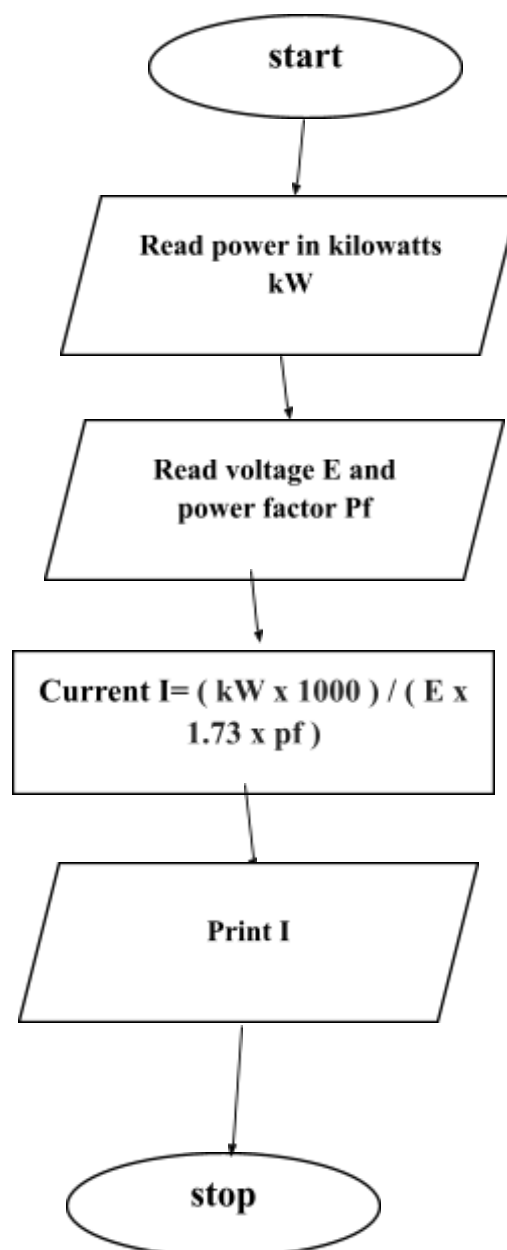
kW- Power in kilowatts

E- Voltage

Pf- Power factor

**Step4:** Print the current in Ampere

**Step5:** Stop



**Result:**

Thus, the flowchart to compute electric current in three phase AC circuit has been developed successfully.

<b>Ex No 2</b>	<b>Python programming using simple statements and expressions (exchange the values of two variables, circulate the values of n variables, distance between two points).</b>
<b>Ex No 2) a</b>	<b>Program to Exchange the values of two variables</b>

**Aim:**

To develop the python program for exchanging the values of two variables.

**Algorithm:**

**Step1:** Start

**Step2:** Read the value of a, b, c

**Step3:** Assign a to temporary variable c

**Step4:** Assign a=b

**Step5:** Assign b=a

**Step6:** Print the exchanged the values of a and b

**Step7:** Stop

**Program:**

```
a = int(input("Enter a value "))
b = int(input("Enter b value "))
c = a
a = b
b = c
print("a=",a,"b=",b)
```

**Output:**

```
Enter a value 5
Enter b value 8
a=8
b=5
```

**Exchange the values of two variables without using temporary variable**

**Program:**

```
a=int(input("Enter value of first variable: "))
b=int(input("Enter value of second variable: "))
a=a+b
b=a-b
a=a-b
print("a is:",a," b is:",b)
```

**Output:**

```
Enter value of first variable: 3
Enter value of second variable: 5
a is: 5 b is: 3
```

**Result:**

Thus, the python program for exchanging the values of two variables has been executed successfully.

**Aim:**

To develop the python program to circulate the values of n variables.

**Algorithm:**

**Step1:** Start

**Step2:** Read the numbers to be stored in the list

**Step3:** Initialize for loop with i=1 to len(a) and increment by 1

**Step4:** The elements in list are circulated using slicing concept

**Step5:** Print the circulated value in list

**Step6:** Stop

**Program:**

```
a=list(input("enter the list"))
print(a)
for i in range(1,len(a),1):
print(a[i:]+a[:i])
```

**Output:**

```
enter the list '1234'
['1', '2', '3', '4']
['2', '3', '4', '1']
['3', '4', '1', '2']
['4', '1', '2', '3']
```

**Result:**

Thus, the program for circulating the values of n variables has been executed successfully.

**Aim:**

To develop the python program to calculate the distance between two points

**Algorithm:**

**Step1:** Start

**Step2:** Get the value for  $X_1$ ,  $X_2$ ,  $Y_1$  and  $Y_2$  from the user

**Step3:** import math. sqrt

**Step4:** The distance between two points are calculated using the formula  
 $\text{distance} = \sqrt{((X_2 - X_1)^2 + (Y_2 - Y_1)^2)}$

**Step5:** Print the distance between two points

**Step6:** Stop

**Program:**

```
import math
x1=int(input("enter x1"))
y1=int(input("enter y1"))
x2=int(input("enter x2"))
y2=int(input("enter y2"))
distance =math.sqrt((x2-x1)**2)+((y2-y1)**2)
print("The distance between two points is: ",distance)
```

**Output:**

```
enter x1      7
enter y1      6
enter x2      5
enter y2      7
The distance between two points is 2.5
```

**Result:**

Thus, the python program to calculate the distance between two points has been executed successfully



<b>Ex. No: 3</b>	<b>Scientific problems using Conditionals and Iterative loops. (Number eries, Number Patterns, pyramid pattern)</b>
<b>Ex. No: 3a</b>	<b>Program to display the Fibonacci sequence up to nth term</b>

**Aim:**

To develop a python program to display the Fibonacci sequence up to nth term.

**Algorithm:**

**Step 1:** Start

**Step 2:** Get the value of the number of terms from the user and store it in the variable named "nterms"

**Step 3:** Assign  $n1=0$ ,  $n2=1$  and  $count=0$

**Step 4:** If the value of nterms is less than or equal to 1 then print "Please enter a positive integer"

**Step 5:** If the value of nterms is equal to 1 then print the Fibonacci sequence as zero

**Step 6:** Else, initialize a while loop with the condition ( $count < nterms$ )

**Step 7:** If the condition is true print  $n1$

**Step 8:** Then assign  $nth=n1+n2$

**Step 9:** Assign  $n1=n2$  and  $n2=nth$  to update the values

**Step 10:** Assign  $count = count+1$

**Step 11:** Repeat the while loop until the condition gets false

**Step 12:** Stop

**Program:**

```
nterms = int(input("How many terms? "))
```

```
# first two terms
```

```
n1, n2 = 0, 1
```

```
count = 0
```

```
# check if the number of terms is valid
```

```
if nterms <= 0:
```

```
    print("Please enter a positive integer")
```

```
# if there is only one term, return n1
```

```
elif nterms == 1:
```

```
    print("Fibonacci sequence upto",nterms,":")
```

```
    print(n1)
```

```
# generate fibonacci sequence
```

```
else:
```

```
    print("Fibonacci sequence:")
```

```
    while count < nterms:
```

```
        print(n1)
```

```
        nth = n1 + n2
```

```
        # update values
```

```
n1 = n2
n2 = nth
count += 1
```

**Output**

How many terms? 7

Fibonacci sequence:

0  
1  
1  
2  
3  
5  
8

**Result:**

Thus, the python program to display the Fibonacci sequence up to nth term has been developed and executed successfully.

<b>Ex. No: 3b</b>	<b>Program to print numbers in diamond pattern</b>
-------------------	--

**Aim:**

To develop a python program to print numbers in diamond pattern.

**Algorithm:**

**Step 1:** Start

**Step 2:** Define the sub function pattern(n) and read the value of n

**Step 3:** Assign  $k=2*n-2$  and  $x=0$

**Step 4:** For the upper half of the diamond pattern initialize a for loop with  $i=0$  to n

**Step 5:** Assign  $x=x+1$

**Step 6:** Initialize a nested for loop with  $j=0$  to k

**Step 7:** Print the spaces until the loop until the loop reaches its end value

**Step 8:** Assign  $k=k-1$

**Step 9:** Initialize a nested for loop with  $j=0$  to  $i+1$

**Step 10:** Print the x values. Print the spaces after the x values and carriage return

**Step 11:** Assign  $k=n-2$

**Step 12:** Assign  $x=n+2$

**Step 13:** For the lower half of the diamond pattern initialize another for loop with  $i=n$  to -1, increment by -1

**Step 14:** Assign  $x=x-1$

**Step 15:** Initialize a nested for loop with  $j=k$  to 0, increment by -1

**Step 16:** Print the spaces until the loop until the loop reaches its end value

**Step 17:** Assign  $k=k+1$

**Step 18:** Initialize another nested for loop with  $j=0$  to  $i+1$

**Step 19:** Print the  $x$  values. Print the spaces after the  $x$  values and carriage return

**Step 20:** Invoke the sub function `pattern(n)` with  $n=5$

**Step 21:** Stop

**Program:**

```
def pattern(n):
    k = 2 * n - 2
    x = 0
    for i in range(0, n):
        x += 1
        for j in range(0, k):
            print(end=" ")
        k = k - 1
        for j in range(0, i + 1):
            print(x, end=" ")
        print("\r")
    k = n - 2
    x = n + 2
    for i in range(n, -1, -1):
        x -= 1
        for j in range(k, 0, -1):
            print(end=" ")
        k = k + 1
        for j in range(0, i + 1):
            print(x, end=" ")
        print("\r")
    pattern(5)
```

**Output:**

```
      1
     2 2
    3 3 3
   4 4 4 4
  5 5 5 5 5
 6 6 6 6 6 6
 5 5 5 5 5
 4 4 4 4
 3 3 3
 2 2
 1
```

**Result:**

Thus, a python program for printing numbers in diamond pattern has been developed and executed successfully.

<b>Ex. No: 3c</b>	<b>Program to print numbers in pyramid pattern</b>
-------------------	--

**Aim:**

To develop a python program to print umbers in pyramid pattern.

**Algorithm:**

**Step 1:** Start

**Step 2:** Get the value of numbers of rows from the user as integer and store it in the variable named "rows"

**Step 3:** Assign  $k=0$ ,  $count=0$  and  $count1=0$

**Step 4:** Initialize a for loop with  $i=1$  to  $rows+1$

**Step 5:** Initialize a nested for loop with  $space=1$  to  $((rows-i) + 1)$  to print the space

**Step 6:** Print the spaces and print another space after every space

**Step 7:** Assign  $count=count+1$

**Step 8:** Initialize a while loop when  $k$  is not equal to  $((2*i)-1)$

**Step 9:** If  $count$  is less than or equal to  $(rows-1)$  then print  $(i + k)$

**Step 10:** Assign  $count=count+1$

**Step 11:** Else, assign  $count1=count1+1$

**Step 12:** Print  $(i + k - (2*count1))$  and print space after each and every number

**Step 13:** Assign  $k=k+1$

**Step 14:** Assign  $count1=count=k=0$

**Step 15:** Repeat the for loop until the end value is reached

**Step 16:** Stop

**Program:**

```
rows = int(input("Enter number of rows: "))
k = 0
count=0
count1=0
for i in range(1, rows+1):
    for space in range(1, (rows-i)+1):
        print(" ", end="")
        count+=1

    while k!=((2*i)-1):
        if count<=rows-1:
            print(i+k, end=" ")
            count+=1
        else:
            count1+=1
            print(i+k-(2*count1), end=" ")
            k += 1

    count1 = count = k = 0
    print()
```

**Output:**

```
1
2 3 2
3 4 5 4 3
4 5 6 7 6 5 4
```

5 6 7 8 9 8 7 6 5

**Result:**

Thus, the python program for printing numbers in pyramid pattern is developed and executed successfully.

<b>Ex. No: 4</b>	<b>Implementing real-time/technical applications using Lists, Tuples. (Items present in a library/Components of a car/Materials required for construction of a building –operations of list &amp; tuples)</b>
<b>Ex. No: 4) a</b>	<b>Python program to implement Operations in List</b>

**Aim:**

To develop a python program to implement operations in list.

**Algorithm:**

**Step 1:** Start

**Step 2:** Get a list L1 from the user

**Step 3:** Call the subfunction display() and pass the list L1 to the subfunction

**Step 4:** Define the subfunction display().

**Step 5:** In the definition of the subfunction display(L2), print the options to be selected by the user.

**Step 6:** Get an option from the user as integer and store it in a variable named “option”.

**Step 7:** If option==1 then print the total number of books in the list using len(L2) function and go to step 5.

**Step 8:** If option==2 then get the name off the book to be appended and store it in a temporary variable. Append the book using list\_name.append(item) and print the list after appending. And go to step 5.

**Step 9:** If option==3 then get the name of the book to be inserted and get the index position where it should be inserted and store them in some temporary variables. Insert the book in the list using list\_name.insert(index, item). Print the list after inserting and go to step 5.

**Step 10:** If option==4 then get the name of the book whose index should be displayed. Find the index position using list\_name.index(item). Print the index position of the book and go to step 5.

**Step 11:** If option==5 then get the size of the new list to be added and get the name of the items in the list separated by space. Add the new list of books to the existing list using list\_name.extend(newlist). Print the list after extending and go to step 5.

**Step 12:** If option==6 then pop a book which is in the end of the list from the list using list\_name.pop() and print the item popped. Print the list after popping and go to step 5.

**Step 13:** If option==7 then get the book name to be removed and remove the book from the list using list\_name.remove(item). Print the list after removing and got to step 5.

**Step 14:** If option==8 then reverse the list using list\_name.reverse(). Display the list after reversing and go to step 5.

**Step 15:** If option==9 then sort the ietms using list\_name.sort(). Print the list after sorting and go to step 5.

**Step 16:** If option==10 then clear the list using list\_name.clear(). Print the cleared list and do to step 5.

**Step 17:** If option==11 then exit from the subfunction and terminate the program.

**Step 18:** Else, print “The option is invalid”.

**Step 19:** Stop

**Program**

```

def display(L2):
    print("1.Total number of books \t 2.Append\t 3.Insert\t4.Display the position of book in list\n5.To add list of books\t6.Pop\t 7.Remove\t8.Reverse \t 9.Sort the books\t10.Clear\n11.Exit")
    option=int(input("enter your option"))
    if option==1:
        print ("Total number of books are",len(L2))
        display(L2)
    elif option==2:
        temp=input("Enter the book to be appended")
        L2.append(temp)
        print ("after append",L2)
        display(L2)
    elif option==3:
        temp=input("Enter the book to be inserted")
        i=int(input("Enter the index position"))
        L2.insert(i,temp)
        print ("after insertion",L2)
        display(L2)
    elif option==4:
        temp=input("Enter the book name")
        c=L2.index(temp)
        print ("Position of the book is",c)
        display(L2)
    elif option==5:
        n=int(input("Enter the size of list"))
        L3 = list(str(num) for num in input("Enter the list items separated by space ").strip().split())[:n]
        L2.extend(L3)
        print ("After extend", L2)
        display(L2)
    elif option==6:
        p=L1.pop()
        print ("Item popped:", p)
        print ("list after popping", L2)
        display(L2)
    elif option==7:
        temp=input("Enter the book name to be removed")
        L2.remove(temp)
        print ("After removing",L2)
        display(L2)
    elif option==8:
        L1.reverse()
        print ("After reversing:",L2)
        display(L2)
    elif option==9:
        L1.sort()
        print ("sorted list: ", L1)
        display(L2)

```

```

elif option==10:
    L1.clear()
    print ('All cleared:', L1)
    display()
elif option==11:
    exit()
else:
    print("Invalid option")
    display(L2)
L1=['DBMS','OS','PQT','IRT']
display(L1)

```

### Result:

Thus, the python program for implementing the operations of list has been developed and executed successfully.

<b>Ex. No: 4) b</b>	<b>Python program to implement Operations in tuples</b>
---------------------	---

### Aim:

To develop the python program for implementing operations in tuples

### Algorithm:

**Step1:** Start.

**Step2:** To create a tuple syntax: Tuple=(element1, element2) or tuple=tuple(data\_items).

**Step3:** To access items in tuple syntax: Tuple[index].

**Step4:** Concatenation operation on tuples is performed by using the operator + . syntax: tuple1+tuple2.

**Step5:** Nesting operation on tuples is done by using “,” syntax: Tuple3= (Tuple1, Tuple2).

**Step6:** Slicing operation on tuples is done by using “:” and giving the range of elements to be sliced syntax: Tuple [start: end].

**Step7:** To find length of Tuples len () is used.

**Step8:** As tuple have immutable property, changes cannot be done in tuples

**Step9:** Deletion of a tuple can be done by del ().

**Step10:** Membership test on tuples: in → True if value is found.

not in → True if value is not found.

Syntax: (item in tuple), (item not in tuple)

**Step11:** In built in functions we use the following functions to find the maximum, minimum, sum, sorted of the tuples: max (), min (), sum (), sorted ()

**Step12:** Use of tuples as keys in dictionary syntax: tuple[(‘key’)] = ‘value’

**Step13:** Tuple packing and unpacking: Assigning multiple values into a tuple is packing. use ‘\*’ to unpack tuples. Syntax: tuple=(“value1”, “value2”), (key1,key2)=tuple

**Step14:** To iterating over tuple elements syntax: for iterator in tuple



## Step15: Stop

### Program:

#### Creating Tuples:

```
type1 = (1, 3, 4, 5, 'test')
print (type1)
type2= 1, 4, 6, 'exam', 'rate'
print(type2)
```

#### Accessing Items in a Tuple

```
access_tuple = ('a', 'b', 1, 3, [5, 'x', 'y', 'z'])
print(access_tuple[0])
print(access_tuple[4][1])
access_tuple = ('a', 'b', 1, 3)
print(access_tuple[-1])
access_tuple = ('a', 'b', 1, 3, 5, 'x', 'y', 'z')
print(access_tuple[2:5])
```

#### Concatenation Operation on Tuples

```
Tuple1 = (1, 3, 4)
Tuple2 = ('red', 'green', 'blue')
print (Tuple1 + Tuple2)
```

#### Nesting Operation on Tuples

```
Tuple1 = (1, 3, 4)
Tuple2 = ('red', 'green', 'blue')
Tuple3 = (Tuple1, Tuple2)
print (Tuple3)
```

#### Slicing Operation on Tuples

```
Tuple1 = (1, 3, 4, 'test', 'red')
Sliced=(Tuple1[2:])
print (Sliced)
```

#### Finding length of Tuples

```
Tuple1 = (1, 3, 4, 'test', 'red')
print(len(Tuple1))
```

#### Changing a Tuple(Throws error)

```
Tuple1 = (1, 3, 4, 'test', 'red')
Tuple1[1] =4
```

**if the item in tuple itself is a mutable data type like a list, its nested items can be changed.**

```
Tuple1 = (1, 3, 4, 'test', 'red')
Tuple1[1] =4
```

#### Deleting a Tuple

```
Tuple1 = (1, 3, 4, 'test', 'red')
del (Tuple1)
print (Tuple1)
```

#### Membership Test on Tuples

```
Tuple1 = (1, 3, 4, 'test', 'red')
print (1 in Tuple1)
print (5 in Tuple1)
```

### **Inbuilt functions for Tuples**

```
tuple1 = (1, 2, 3, 6)
print(max(tuple1))
print(sum(tuple1))
print(sorted(tuple2))
```

### **Use of Tuples as keys in Dictionaries**

```
tuplekey = {}
tuplekey[('blue', 'sky')] = 'Good'
tuplekey[('red', 'blood')] = 'Bad'
print(tuplekey)
```

### **Tuple Packing and Unpacking**

```
person = ("Salman", '5 ft', "Actor")
(name, height, profession) = person
print(name)
print(height)
print(profession)
```

### **Iterate through a Tuple**

```
my_tuple = ("red", "Orange", "Green", "Blue")
# iterating over tuple elements
for colour in my_tuple:
    print(colour)
```

### **Result:**

Thus, the python program to implement operations in tuples has been developed and executed successfully.

<b>Ex No 5</b>	<b>Implementing real time/technical applications using sets, dictionaries (Language, component of an automobile, Elements of a civil structures, etc. operations of sets &amp; dictionaries)</b>
<b>Ex No 5) a</b>	<b>Python program to implement Operations in sets</b>

**Aim:**

To develop the python program to implement operations in sets.

**Algorithm:**

**Step1:** Start

**Step2:** Get two sets of items from the user and call the subfunction display (L1, L2) by passing the L1, L2 to the subfunction.

**Step3:** In the definition of the subfunction, print the options to be selected by the user. The options are operations in sets like union, intersection, set difference and symmetric difference.

**Step4:** If option==1, using the '|' (vertical bar) operator to find the union of the sets.  
Syntax:(set\_name1 | set\_name2) and go to step3.

**Step5:** If option==2, perform the intersection operation using '&'symbol. Syntax:  
(set\_name1&set\_name2) and go to step3.

**Step6:** If option==3, using '-' operator, find the difference of the sets,  
Syntax:(set\_name1-set\_name2) and go to step3.

**Step7:** If option==4, using the '^' symbol to find the symmetric difference of the sets  
Syntax:(set\_name1^set\_name2)

**Step8:** Stop

**Program:**

```
def display(L1,L2):
    print("1.Union\t 2.Intersection\t3.Set difference\t4.Symmetric difference")
    option=int(input("Enter a option"))
    if (option==1):
        print("Union of L1 and L2 is ",L1 | L2)
        display(L1,L2)
    elif (option==2):
        print("Intersection of L1 and L2 is ",L1 & L2)
        display(L1,L2)
    elif (option==3):
        print("Difference of L1 and L2 is ",L1 - L2)
        display(L1,L2)
    elif (option==4):
        print("Symmetric difference of L1 and L2 is ",L1 ^ L2)
```

```

display(L1,L2)

L1 = {'Pitch', 'Syllabus', 'Script', 'Grammar', 'Sentences'};
L2 = {'Grammar', 'Syllabus', 'Context', 'Words', 'Phonetics'};

display(L1,L2)

```

### Result:

Thus, the program to implement operations in sets has been developed and executed successfully.

<b>Ex No 5) b</b>	<b>Python program to implement Operation in dictionary</b>
-------------------	--

### Aim:

To develop the python program to implement operation in dictionary

### Algorithm:

**Step1:** To Create a dictionary use the Syntax: `dictionary_name = {"key": "value"}` and print the dictionary.

**Step2:** To add a new element in the dictionary using this Syntax: `dictionary_name[key] = "new_value"` and print the dictionary after adding a new element

**Step3:** To change an element in the dictionary using this Syntax: `dictionary_name ["existing key"] = "new_value"` and print the dictionary after changing the value.

**Step4:** To access an element in the dictionary using this Syntax: `dictionary_name.get("key")`.

**Step5:** To remove an element from the dictionary using this Syntax: `dictionary_name.pop("key")` and print the dictionary after removing.

**Step6:** To find the length of the dictionary use the Syntax: `len(dictionary_name)` and print the length of the dictionary.

**Step7:** To print all the keys in the dictionary use the Syntax: `print(dictionary_name.keys())`

**Step8:** To perform for loop in dictionary to find the set of squares, initialize an empty set and use the Syntax: `for iterator in range to initialize for loop`. To find the dictionary of squares use the Syntax: `dictionary_name[iterator]=iterator*iterator` and print the dictionary.

**Step9:** To find the dictionary of squares by using equivalent code use the Syntax: `dictionary_name= {x:x*x for x in range}` and print the dictionary

**Step10:** To perform if conditionals to print the odd number in the dictionary use the Syntax: `{x:x*x for x in range () if x%2==1}` and print the dictionary of odd number.

**Step11:** To perform membership operation in dictionary use the syntax: `(value in dictionary_name)`, `(value not in dictionary_name)`, the output of this operation is Boolean values like True or False.

**Step12:** To perform iteration in dictionary, initialize a for loop and use the syntax: for iterator in dictionary\_name, dictionary\_name[iterator] and print the iterated values.

**Step13:** To sort the items in dictionary, use the syntax: sorted(dictionary\_name) and print the sorted dictionary.

**Program:**

**#Creating Dictionary**

```
college = {  
    "name": "NSCET",  
    "code": "INDIA",  
    "id": "9210"  
}
```

```
print(college)
```

**#Adding new element**

```
college["location"] = "IBP"
```

```
print(college)
```

**#Changing element**

```
college["location"] = "Theni"
```

```
print(college)
```

**#Accessing elements in dictionary**

```
print(college.get('name'))
```

**# Removing elements from a dictionary**

```
college.pop("code")
```

```
print(college)
```

**# Length of dictionary**

```
print(len(college))
```

**#get all the keys in the dictionary**

```
print(college.keys())
```

**#for loop in dictionary**

```
squares = {}
```

```
for x in range(6):
```

```
    squares[x] = x*x
```

```
print(squares)
```

**#equivalent code**

```
squares = {x: x*x for x in range(6)}
```

```
print(squares)
```

**#Dictionary operations with if conditionals**

```
odd_squares = {x: x*x for x in range(11) if x % 2 == 1}
```

```
print(odd_squares)
```

**# Membership Test for Dictionary Keys**

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
```

```
print(1 in squares)
```

```
print(2 not in squares)
```

**# Iterating through a Dictionary**

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
```

```
for i in squares:
```

```
    print(squares[i])
```

**#Print sorted elements**

```
print(sorted(squares))
```

**Result:**

Thus, the python program to implement operation of dictionary has been developed and executed successfully.

<b>Ex No:6</b>	<b>Implementing programs using Functions. (Factorial, largest number in a list, area of shape)</b>
<b>Ex No:6.a</b>	<b>Python Program to find factorial using functions</b>

**Aim:**

To develop the program to find factorial using functions

**Algorithm:**

Step 1: Start

Step 2: Get the Number from user, which the factorial should be calculated

Step 3: Call function factorial ()

Step 4: Define the sub function factorial ()

Step 5: If the value of x is then the sub function returns to the main function

Step 6: else the function factorial is called recursively to calculate the factorial value of x

Step 7: The final value is returned to the main function

Step 8: Stop

**Program:**

```
def factorial(x):  
    if x == 1:  
        return 1  
    else:  
        return (x * factorial(x-1))  
num = int(input("Enter a number: "))  
result = factorial(num)  
print("The factorial of", num, "is", result)
```

**Result:**

Thus the python program to find factorial using function has been developed and executed successfully.

<b>Ex No:6.b</b>	<b>Program to find largest number in a list using functions</b>
------------------	---

**Aim :**

To develop the python program to find largest number in a list using function.

**Algorithm:**

Step 1: Start

Step 2: Initialize the empty list

Step 3: Get the number of elements from the user

Step 4: Enter the numbers

Step 5: Call the sub function maximum

Step 6: Define the sum function maximum()

Step 7: Assign max=[0]

Step 8: Initialize for loop for I in 0 to l

Step 9: If i is greater than max. then assign max=i

Step 10 : The maximum number is displayed to use

**Program:**

```
def maximum(L):  
    max=L[0]  
    for i in L:  
        if i > max:  
            max = i  
    print(max)
```

```

L = [ ]
size = int(input('How many elements you want to enter? '))
print('Enter the numbers')
for i in range(size):
    data = int(input())
    L.append(data)
maximum(L)

```

**Result:**

Thus the program to find largest number in a list using function has been developed and executed successfully.

<b>Ex No:6.c</b>	<b>Program to find area of shape using functions</b>
------------------	--

**Aim:**

To develop the python program to find area of shape using functions.

**Algorithm:**

- Step 1: Start
- Step 2: Get the radius value from the user
- Step 3: Call sub function area ()
- Step 4: Define the sub function area ()
- Step 5: Area of circles calculated using formula  $3.14*r$
- Step 6: Return the result to the main function
- Step 7: Call the function area()
- Step 8: The final result s returned to the main function then point the circle of area

**Program:**

```

def area(r):
    result=3.14*r*r
    return result
radius=int(input("Enter the radius"))
a=area(radius)
print("The of circle with radius %d is: %.2f" % (radius ,a))

```

**Result:**

Thus the python program to find area of shape using function has been developed and executed successfully

<b>Ex No:7</b>	<b>Implementing programs using Strings. (reverse, palindrome, character count, replacing characters)</b>
<b>Ex No:7.a</b>	<b>Python program to reverse a string</b>

**Aim:**

To develop a python program to reverse a string.



**Algorithm:**

Step-1-Start

Step-2-Get the string from user

Step-3-Calculate the length of the string len(s)

Step-4-The string is displayed in reverse order by using slicing concept

Step-5-Print (rev)

Step-6-Stop

**Program:**

```
s= input("enter the string")
length=len(s)
rev=s[length::-1]
print (rev)
```

**Result:**

Thus the program to reverse a string has been developed and executed successfully.

<b>Ex No:7.b</b>	<b>Python program to check palindrome</b>
------------------	---

**Aim:**

To develop a python program to check the given string is palindrome or not.

**Algorithm:**

Step-1-Start

Step-2-Get the string from user

Step-3-Calculate the length of the string using len ()

Step-4-The string is displayed in reverse order by using slicing concept

Step-5-Original string is compared with reverser string

Step-6-If it is equal, then the given string is palindrome

Step-7-Else it is not equal then the given string is not a palindrome

Step-8-Stop

**Program:**

```
s= input("enter the string")
length=len(s)
rev=s[length::-1]
if s==rev:
    print("The string is palindrome")
else:
    print("The string is not palindrome")
```

**Result:**

Thus the program to check the given string is palindrome has been developed and executed successfully.

<b>Ex No:7.c</b>	<b>Python program to count the character in string</b>
------------------	--

**Aim:**

To develop a python program to count the character in string

**Algorithm:**

Step-1-Start

Step-2-Get the string from user  
 Step-3-Initialize count=0  
 Step-4-Initialize for i in range(0, len(s))  
 Step-5-If the string s[i] is not in the empty string  
 Step-6-Count is incremented by count +1  
 Step-7-Print the total number of characters in a string  
 Step-8-Stop

**Program:**

```
s = input("Enter the string")
count = 0;
for i in range(0, len(s)):
    if(s[i] != ' '):
        count = count + 1
print("Total number of characters in a string: ",count)
```

**Result:**

Thus the python program to count the character in string has been developed and executed successfully.

<b>Ex No:7.d</b>	<b>Python program to replace character in string</b>
------------------	--

**Aim:**

To develop a python program to replace a character in string

**Algorithm:**

Step-1-Start  
 Step-2-Read the given string from the user  
 Step-3-Get the replaced character from the user and stored in Ch  
 Step-4-The character Ch in string str1 is replaced by the character in new Ch by using replace ()  
 and replaced string is assigned to str2  
 Step-5-Print the original string and replaced character in the string  
 Step-6-Stop

**Program:**

```
str1 = input(" Enter String : ")
ch = input(" Enter the character to be replaced : ")
newch = input(" Enter the New Character : ")
str2 = str1.replace(ch, newch)
print("Original String : ", str1)
print("character replaced String : ", str2)
```

**Result:**

Thus the python program to replace a character in string has been developed and executed successfully.

<b>Ex.No:8.</b>	<b>Implementing programs using written modules and Python Standard Libraries (pandas,numpy. Matplotlib, scipy)</b>
<b>Ex.No:8.a</b>	<b>Python program to reshape matrix using numpy:</b>

**Aim:**

To develop a python program to reshape matrix using Numpy.

**Numpy:**

It is a general purpose array-processing package.It provides a high-performance multidimensional array object and tools for working with these.

**Algorithm:**

Step-1:Start.  
 Step-2:Import Numpy module.  
 Step-3:Two dimensional array is assigned to a.  
 Step-4:Print a.  
 Step-5:Reshape the array into a 3\*2 matrix using reshape() function.  
 Step-6:Print the reshaped array a.  
 Step-7:Stop.

**Program:**

```
import numpy as np
a=np.array([(8,9,10),(11,12,13)])
print(a)
a=a.reshape(3,2)
print(a)
```

**Result:**

Thus the python program to reshape matrix using numpy has been developed and executed successfully.

<b>Ex.No:8.b.</b>	<b>Python program to calculate permutation using scipy:</b>
-------------------	---

**Aim:**

To develop the program to calculate permutation using scipy.

**Scipy:**

Scipy is a free and open-source python library used for scientific computing and technical computing.

**Algorithm:**

Step-1:Start.  
 Step-2:Import perm from scipy.  
 Step-3:Calculate the permutation of the number using perm(n,k) function and store it in a variable 'per'.  
 Step-4:Print per.  
 Step-5:Stop.

**Program:**

```
from scipy.special import perm
#find permutation of 5,2 using perm(n,k) function
per=perm(5,2,exact=True)
print(per)
```

**Result:**

Thus the program to calculate permutation using scipy have been developed and executed successfully.

<b>Ex.No:8.c.</b>	<b>Python program to plot a histogram of marks obtained by students in a class using matplotlib.</b>
-------------------	--

**Aim:**

To develop a python program to plot a histogram of marks obtained by students in a class using matplotlib.

**Matplotlib:**

Matplotlib is a plotting library for the python programming language and its numerical mathematic extension numpy.

**Algorithm:**

Step-1:Start.

Step-2: Import pyplot as plt from matplotlib library.  
 Step-3: Import numpy module as np.  
 Step-4: Set the subplot() value as (1,1).  
 Step-5: Convert the list into an array using array() and store it in 'a'.  
 Step-6: Plot the histogram using hist() with arguments(a,bins=[0,25,50,75,100]).  
 Step-7: Set the title of the histogram as "Histogram of results" using set\_title().  
 Step-8: Set the categories names in x-axis as [0,25,50,75,100] using set\_xticks().  
 Step-9: Set the label name of x-axis as 'marks' using set\_xlabel().  
 Step-10: Set the label name of y-axis as 'No. of students' using set\_ylabel().  
 Step-11: Display the histogram using show().  
 Step-12: Stop.

**Program:**

```
from matplotlib import pyplot as plt
import numpy as np
fig,ax=plt.subplots(1,1)
a=np.array([22,87,5,43,56,73,55,54,1,20,51,5,79,31,27])
ax.hist(a,bins=[0,25,50,75,100])
ax.set_title("Histogram of result")
ax.set_xticks([0,25,50,75,100])
ax.set_xlabel("Marks")
ax.set_ylabel("no.of students")
plt.show()
```

**Result:**

Thus the python program to plot a histogram of marks obtained by students in a class using matplotlib have been developed and executed successfully.

<b>Ex.No:8.d.</b>	<b>Python program to split the following dataframe by school code and get mean, min, and max value of age for each school using Pandas.</b>
-------------------	---

**Aim:**

To develop a python program to split the data frame using pandas.

**Algorithm:**

Step-1: Start.  
 Step-2: Set the set\_option() as ('display max\_rows',none)  
 Step-3: Assign the student\_data values.  
 Step-4: Print student\_data in the table format.  
 Step-5: Using group by() function ,group the student\_code with mean,max,min value of age.And store it in a variable named grouped\_single.  
 Step-6: Print grouped\_single.  
 Step-7: Stop.

**Program:**

**Test Data:**

	school	class	name	date_Of_Birth	age	height	weight	address
S1	s001	V	Alberto Franco	15/05/2002	12	173	35	street1
S2	s002	V	Gino Mcneill	17/05/2002	12	192	32	street2
S3	s003	VI	Ryan Parkes	16/02/1999	13	186	33	street3
S4	s001	VI	Eesha Hinton	25/09/1998	13	167	30	street1
S5	s002	V	Gino Mcneill	11/05/2002	14	151	31	street2
S6	s004	VI	David Parkes	15/09/1997	12	159	32	street4

```
import pandas as pd
```

```

pd.set_option('display.max_rows',None)
#pd.set_option('display.max_columns',None)
student_data=pd.DataFrame({'school_code':['s001','s002','s003','s001','s002','s004'],'class':['V','VI','VI','V','VI','V'],'name':['Albert Franco','Gino Mcnaill','Ryan Parkes','Eesha Hinton','Gino Mcnaill','DavidParkes'],'date_of_birth':['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'],'age':[12,12,13,13,14,12],'height':[173,192,186,167,151,159],'weight':[35,32,33,30,31,32],'address':['Street1','Street2','Street3','Street1','Street2','Street4']},index=['s1','s2','s3','s4','s5','s6'])
print("Original DataFrame:")
print(student_data)
print("\n Mean,Min and Max value of age for each value of the school:")
grouped_single=student_data.groupby('school_code').agg({'age':['mean','min','max']})
print(grouped_single)

```

### Result:

Thus the python program to split the data frame using pandas have been developed and executed successfully.

<b>Ex.No:9.</b>	<b>Implementing real-time/technical applications using File handling. (copy from one file to another, word count, longest word)</b>
<b>Ex.No:9.a</b>	<b>Python program to copy from one file to another</b>

### Aim:

To Develop the Python Program to Copy from One file to another

### Algorithm:

- Step 1: Start
- Step 2: Open a file"file.txt" in write in for made using open() & assign it to f.
- Step 3: Using write () function. Add content to the text file.
- Step 4: Open the "file 2 txt" as f.
- Step 5: Open the "file 2 txt" as f in write mode .
- Step 6: for each line in f, write ach line n f, using write() function.
- Step 7: Using +2 read() display the content "file 2 txt"
- Step 8: Stop

### Program:

```

f = open("file1.txt", "wt")
f.write("Python is an interpreted high-level general-purpose programming language.")

with open("file1.txt") as f:
    with open("file2.txt", "w") as f1:
        for line in f:
            f1.write(line)
f2=open("file2.txt","rt")
print("File copied successfully.The contents are:\n",f2.read())

```

### Result:

Thus the Python Program to Copy from One file to another file has been developed and executed successfully.

<b>Ex.No:9.b.</b>	<b>Python program to count words in a file</b>
-------------------	--

**Aim:**

To Develop the Python Program to Count words in a file

**Algorithm:**

Step 1: Start

Step 2: Open a file "demofile.txt" in write in for mode x & assign it to f.

Step 3: Using write () function. write content to the text file.

Step 4: Close f using f.close()

Step 5: Open the "demofile.txt" in read mode & assign it to f

Step 6: Using f in read() the contents of file is assigned to a variable named data.

Step 7: Using str.split() each word of the file is splinted s assign to variable named words

Step 8: Find the length of the longest word using len(max(words ,key: len) assign it to variable max\_len

Step 9: for each in words,check the condition le(word)=max x len f +ve then assigned longest\_word=word

Step 9: Print longest\_word

Step 10: Stop

**Program:**

```
f = open("demofile.txt", "wt")
```

```
f.write("Python is an interpreted high-level general-purpose programming language.")
```

```
f = open("demofile.txt", "rt")
```

```
data = f.read()
```

```
words = data.split()
```

```
print("The total number of words are:",len(words))
```

```
f.close()
```

**Result:**

Thus the Python Program to count the number of words in s file has been developed and executed successfully.

<b>Ex.No:9.c.</b>	<b>Python program to find longest word in a file</b>
-------------------	--

**Aim:**

To develop the python Program to find longest word in a file

**Algorithm:**

Step 1: Start

Step 2: Open a file "demofile.txt" in write in for mode x & assign it to f.

Step 3: Using write () function. write content to the text file.

Step 4: Close f using f.close()

Step 5: Open the "demofile.txt" in read mode & assign it to f

Step 6: Using f in read() the contents of file is assigned to a variable named data.

Step 7: Using str.split() each word of the file is splinted s assign to variable named words

Step 8: Find the length of the longest word using len(max(words ,key: len) assign it to variable max\_len

Step 9: for each in words,check the condition le(word)=max x len f +ve then assigned longest\_word=word

Step 9: Print longest\_word

Step 10: Stop

**Program:**

```
f = open("demofile.txt", "wt")
```

```

f.write("Python is an interpreted high-level general-purpose programming language.")
f.close()
fin = open("demofile.txt","r")
str = fin.read()
words = str.split()
max_len = len(max(words, key=len))
for word in words:
    if len(word)==max_len:
        longest_word =word

print("The longest word in the file is",longest_word)

```

**Result:**

Thus the python program to find longest word in a file has been developed and executed successfully.

<b>Ex.No:10.</b>	<b>Implementing real-time/technical applications using Exception handling. (divide by zero error, voter's age validity, student mark range validation)</b>
<b>Ex.No:10.a</b>	<b>Python program to handle divide by zero error using Exceptional handling</b>

**Aim:**

To develop a python program to handle divide by zero error using exceptional handling.

**Algorithm:**

Step 1: Start

Step 2: Read two numbers a and b calculate c by dividing a by b if no error then go to step 5.

Step 3: if you get value error then print "you have entered get wrong data" and go to step 6.

Step 4: if you zero division error, then print "divide by zero division!!!" and go to step 6

Step 5: Else print c

Step 6: Stop

**Program:**

try :

```

a=int(input("Enter value of a: "))

```

```

b=int(input("Enter value of b: "))

```

```

c=a/b

```

except ValueError:

```

print("You have entered wrong data")

```

except ZeroDivisionError:

```

print("Divide by Zero Error!!!")

```

else:

```

print("The result: ",c)

```

**Result:**

Thus the python program to handle divide by zero error has been developed and executed successfully.

<b>Ex.No:10.b</b>	<b>Python program to validate voter's age using Exceptional handling</b>
-------------------	--

**Aim:**

To develop a python program to validate voter's age using exceptional handling.

**Algorithm:**

Step 1: Start

Step 2: Read the value of age from the user if the value error occurs then goto step 5  
 Step 3: Check the condition age>18, if true then print “eligible to vote” and goto step 6  
 Step 4: Else print “not eligible to vote” and goto step 6  
 Step 5: if value error occurs then print “age must be a valid number”  
 Step 6: Stop

**Program:**

```
try:
    age=int(input("Enter your age"))
    if age>=18:
        print("Eligible to vote")
    else:
        print("Not eligible to vote")
except:
    print("age must be a valid number")
```

**Result:**

Thus the python program to validate voter’s age using exceptional handling has been developed and executed successfully.

<b>Ex.No:10.c</b>	<b>Python program to validate student mark range using Exceptional handling</b>
-------------------	---

**Aim:**

To develop a python program to a validate students mark range using exceptional handling.

**Algorithm:**

Step 1: Start  
 Step 2: Read the value of mark from the user.if value error occurs then goto step 5  
 Step 3:Check the condition mark>0 and mark <100, if true then the print “valid mark” and goto step6  
 Step 4: Else print “Invalid mark” and goto step6  
 Step 5: if value error occurs, then print “mark must be a valid number”  
 Step 6: Stop

**Program:**

```
try:
    mark=int(input("Enter your mark"))
    if mark>0 and mark<=100:
        print("Valid Mark")
    else:
        print("Invalid mark")
except:
    print("Mark must be a valid number")
```

**Result:**

Thus the python program to validate student mark range using exception handling has been developed and executed successfully.

<b>Ex.No:11</b>	<b>Exploring Pygame</b>
-----------------	-------------------------

**Aim:**

To study the steps to Pygame.

**Introduction to Pygame**

**Python PyGame** library is used to create video games. This library includes several modules for playing sound, drawing graphics, handling mouse inputs, etc. It is also used to create client-side applications that can be wrapped in standalone executables.



## Installing pygame on Windows

### Step 1: Check for Python Installation

In order to install Pygame, Python must be installed already in your system. To check whether Python is installed or not in your system, open the command prompt and give the command as shown below.

#### Command Prompt

```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python --version
Python 3.9.0
```

### Step 2: Check for PIP installation

PIP is a tool that is used to install python packages. PIP is automatically installed with Python 2.7.9+ and Python 3.4+. Open the command prompt and enter the command shown below to check whether pip is installed or not.

#### Select Command Prompt

```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DELL>pip --version
pip 20.2.3 from c:\python39\lib\site-packages\pip (python 3.9)

C:\Users\DELL>
```

### Step 3: Install Pygame

To install Pygame, open the command prompt and give the command as shown below:

pip install pygame

#### Command Prompt

```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python --version
Python 3.9.0

C:\Users\DELL>pip --version
pip 20.2.3 from c:\python39\lib\site-packages\pip (python 3.9)

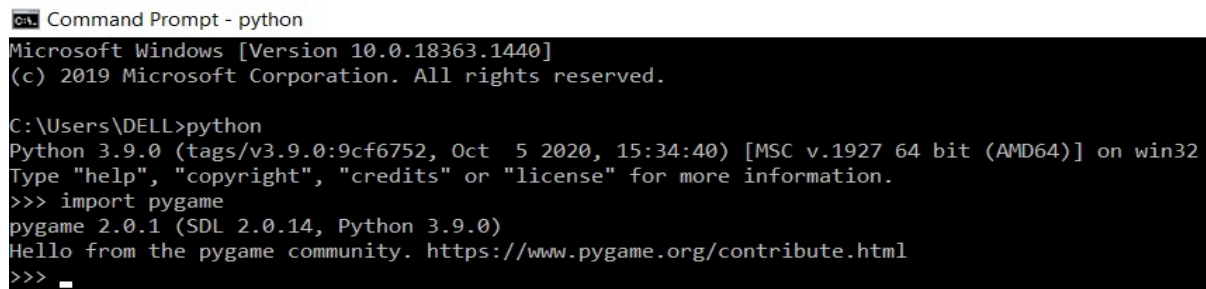
C:\Users\DELL>pip install pygame
Collecting pygame
  Downloading pygame-2.0.1-cp39-cp39-win_amd64.whl (5.2 MB)
    |#####| 5.2 MB 148 kB/s
Installing collected packages: pygame
Successfully installed pygame-2.0.1
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\python39\python.exe -m pip install --upgrade pip' command

C:\Users\DELL>
```

Pygame is successfully installed as shown in the image above.

#### Step 4: Check Whether PyGame is Working or not

Now open a new terminal and import the Pygame library to see whether it is working fine or not in our system. The library is imported successfully means we got success.



```
Command Prompt - python
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import pygame
pygame 2.0.1 (SDL 2.0.14, Python 3.9.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
>>> _
```

In this way, we can install the pygame module in Python.

#### Result:

Thus the Pygame has been installed successfully.

Ex.No:12	Developing a game activity using Pygame like bouncing ball.
----------	---

#### Aim:

To develop a game activity to bounce a ball using pygame.

#### Algorithm:

- Step-1: Import pygame module.
- Step-2: Invoke pygame.init() to initialize all imported pygame modules..
- Step-3: Assign the height and width for the panel.
- Step-4: Assign the speed for the ball movement.
- Step-5: Load the image of ball.
- Step-6: invoke move() and assign the direction for ball movement.
- Step-7: invoke pygame.display.flip() to update the full display Surface to the screen

#### Program:

```
import sys, pygame
pygame.init()
size = width, height = 800, 400
speed = [1, 1]
background = 255, 255, 255
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Bouncing ball")
ball = pygame.image.load("Star.png")
ballrect = ball.get_rect()
while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    ballrect = ballrect.move(speed)
    if ballrect.left < 0 or ballrect.right > width:
```

```
    speed[0] = -speed[0]
    if ballrect.top < 0 or ballrect.bottom > height:
        speed[1] = -speed[1]
    screen.fill(background)
    screen.blit(ball, ballrect)
    pygame.display.flip()
```

**Result:**

Thus the a game activity to bounce a ball using pygame has been developed and execute successfully.