Sri Eshwar College of Engineering

Kinathukadavu, Coimbatore-641202

(Approved By AICTE, New Delhi & Affiliated to Anna University, Chennai)

Department of Electronics and Communication Engineering

R19CS151-PYTHON PROGRAMMING

Sri Eshwar College of Engineering

An Autonomous Institution
Affiliated to Anna University

Kondampatti (Post), Vadasithur (Via), Coimbatore - 641 202.

Department of

....

Record work of .... Laboratory

Certified bonafide Record of work done by

Name : .... Roll No. : ....

Class : .... Branch : ....

Reg. No. : ....

Place : *HOD* *Faculty In charge*

Date :

University Register No. ....

Submitted for the University Practical Examination held on

....

Internal Examiner *External Examiner*

LIST OF EXPERIMENTS

EX. NO	DATE	NAME OF THE EXPERIMENTS	PAGE NO.	MARKS (75)	SIGNATURE OF FACULTY
1.		Develop flow charts and solve simple real-life or scientific or technical problems a. Computing Electrical Current in Three Phase AC circuits b. Retail shop billing c. Temperature control system d. Water level controller e. Traffic signal control f. Automatic washing machine control system g. Automatic street light control system h. Electricity Billing			
2.		Implementation of applications of statements and expressions a. Swap without a temporary variable b. Quadratic Equation c. Valid Palindrome d. Integer to Roman Letter			
3.		Implementation of Conditions and Iterative loops a. check whether an alphabet is a vowel or consonant b. sum of all even numbers from 0 to n c. the factorial of a number			
4.		Implementation of real-time/technical applications using Lists and Tuples a. Minimum Index Sum of Two Lists b. Concatenate two lists index-wise c. Tuple with the same product d. Copy specific elements from one Tuple to a new tuple			
5.		Implementation of real-time/technical applications using Set and Dictionaries a. Magic Dictionary b. Longest Word in Dictionary c. Set Mismatch d. Smallest Number in Finite Set			
6.		Implementation of Functions in the program a. Factorial b. Largest number in a list c. Area of shape			
7.		Implementation of Strings in the program a. Determine if string halves are alike b. Palindrome			

		c. Character count d. Replacing characters			
8.		Implementation of file-handling operations a. Copy from one file to another b. Word count c. Longest word.			
9.		Implementation of libraries (Pandas, NumPy, Matplotlib)			
10.		Implementation of applications of standard libraries a. Handle scalars to work on the NumPy array b. Insert values at random positions in an array c. Convert the index of a series into a column of a Data frame d. Combine many series to form a Data frame e. Get frequency counts of unique items of a series f. Union of two arrays g. Convert a NumPy array to a Data frame of a given shape h. Plotting datasets.)			
11.		Mini Project			
CONTENT BEYOND SYLLABUS					
12.		SCIKIT-LEARN			

Marks in Words:
Average :

SSIGNATURE OF THE FACULTY

MODULE - I	
Ex. No: 1	FLOWCHART TO SOLVE SIMPLE REAL LIFE / SCIENTIFIC / TECHNICAL PROBLEMS
Date:	

AIM:

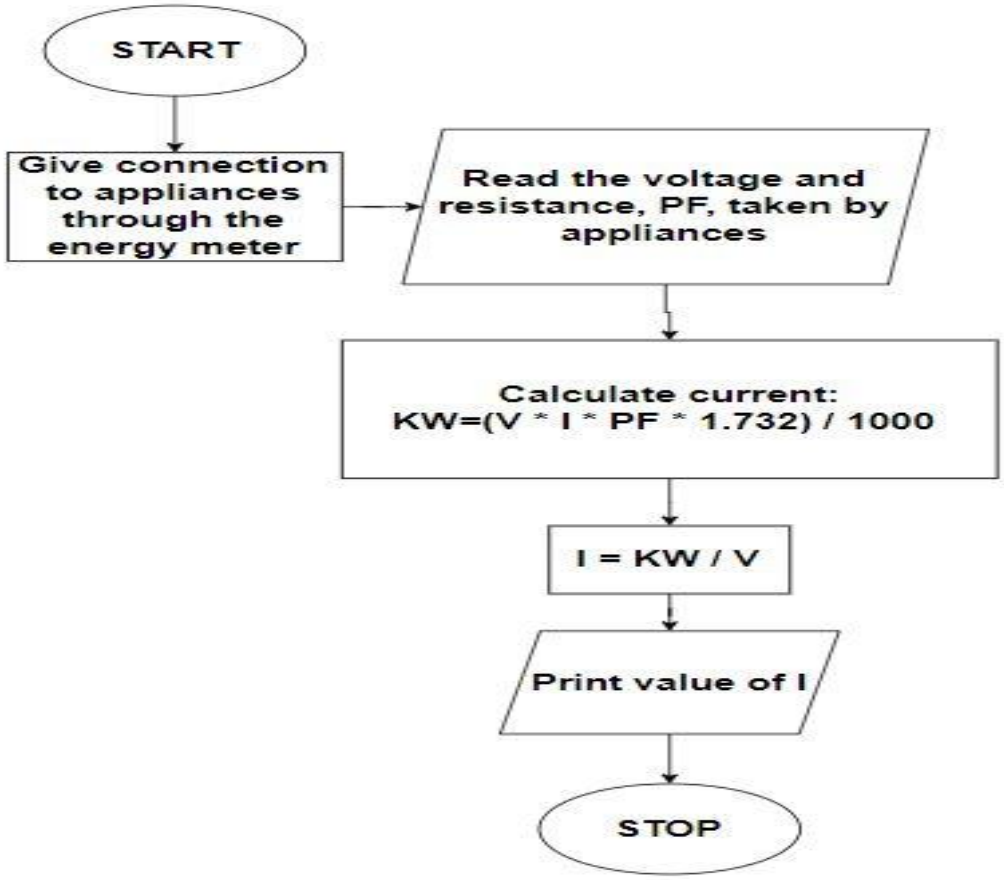
To develop a flow chart and solve simple real – life or scientific or technical problems

a) Computing Electrical Current in Three Phase AC circuit

ALGORITHM:

Step 1:Read the voltage, resistance, PF taken by appliances
Step 2: Calculate KW=(V*I*PF*1.732)/1.000
Step 3:Calculate I=KW/V
Step 4:Print I

FLOWCHART:


START
Give connection to appliances through the energy meter
Read the voltage and resistance, PF, taken by appliances
Calculate current: KW=(V * I * PF * 1.732) / 1000
I = KW / V
Print value of I
STOP

b) Retail Shop Billing

ALGORITHM:

Step 1: Check if the customer wants pay the bill.

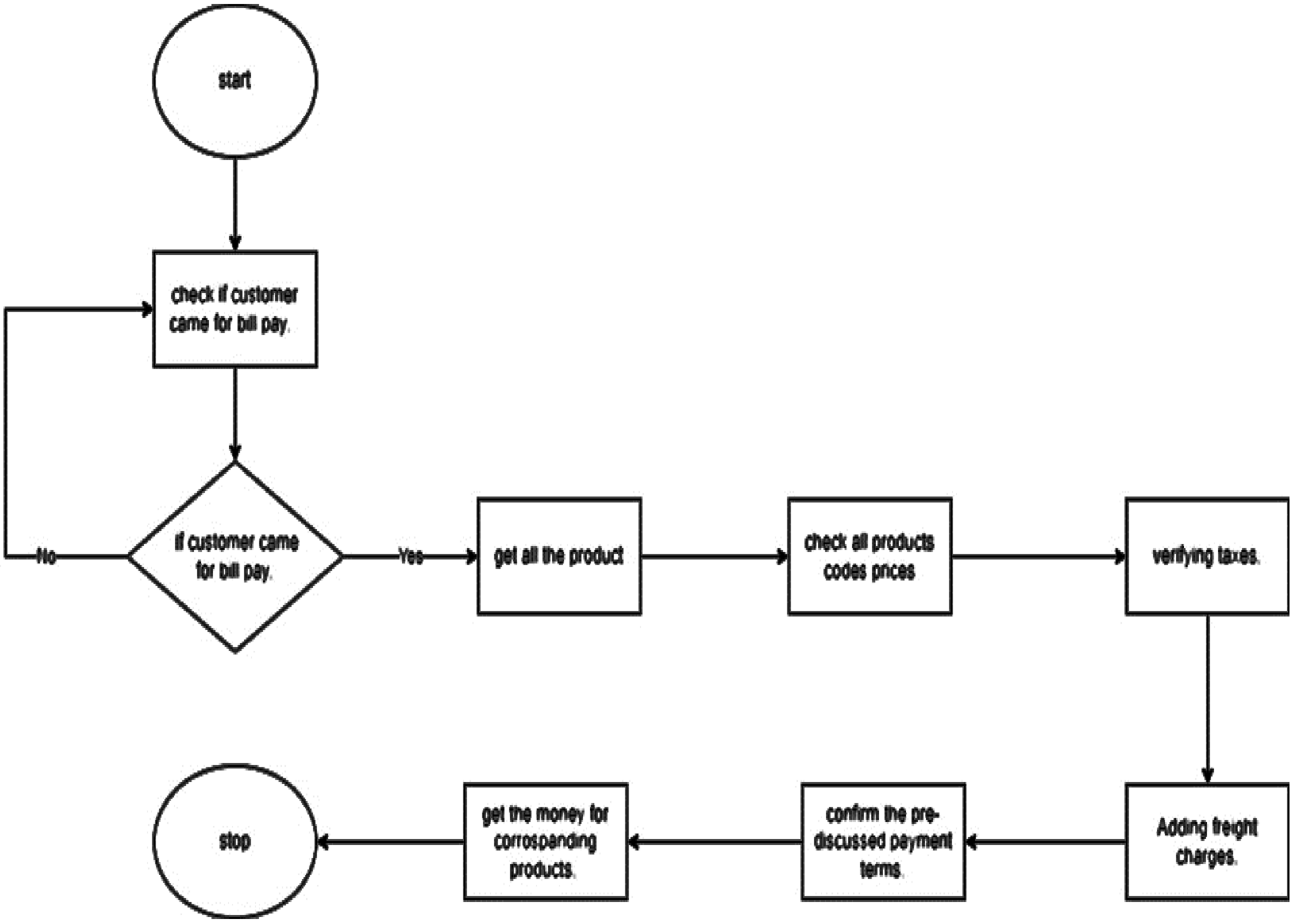
Step 2: If yes, then get all products

Step 3: Check the prices of Product and verify all taxes

Step 4: Confirm the Bill and print the Bill

Step 5: Get the Payment for the corresponding Bill.

FLOWCHART:

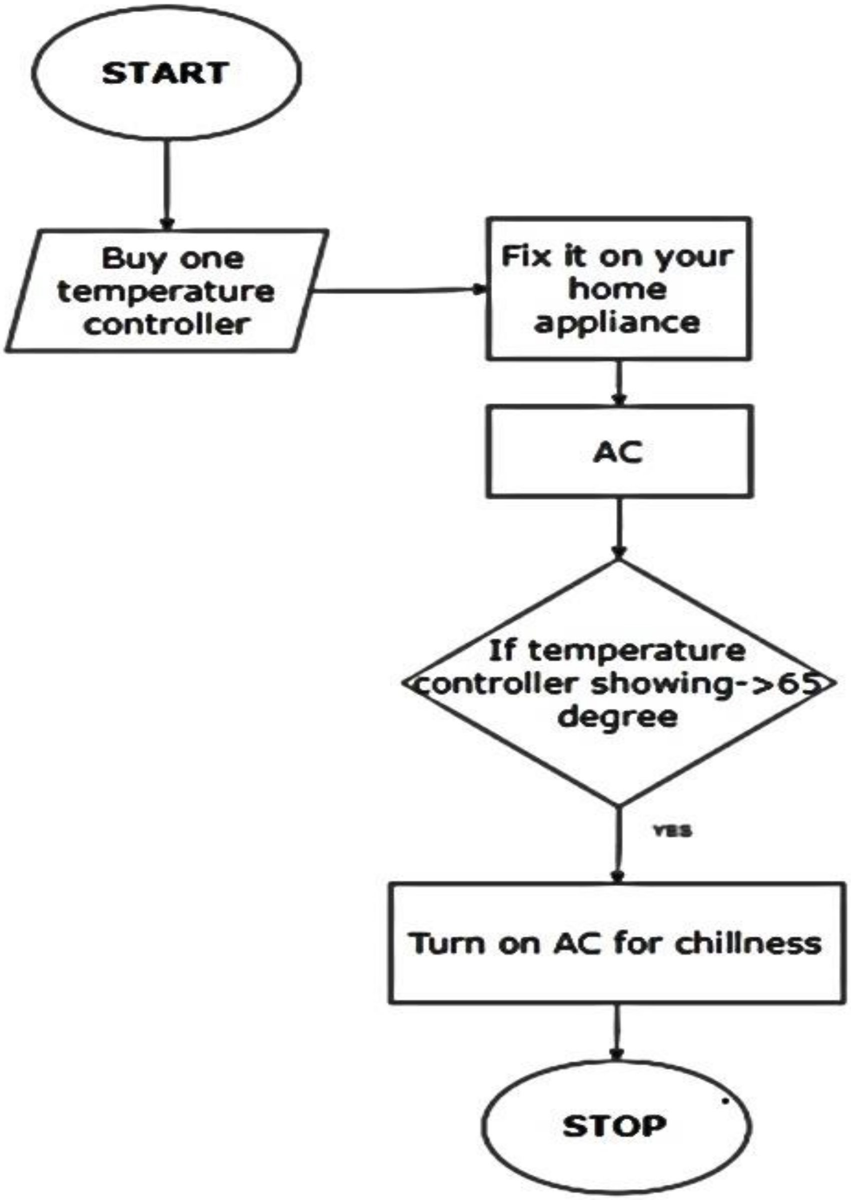

start
check if customer came for bill pay.
No
if customer came for bill pay.
Yes
get all the product
check all products codes prices
verifying taxes.
Adding freight charges.
confirm the pre-discussed payment terms.
get the money for corrospanding products.
stop

c) Temperature Control System

ALGORITHM:

Step 1: Read the Temperature
Step 2:If Temperature >65 degrees , then turn on AC for chillness
Step 3: If Temperature >15 degrees &< 65 degrees , then turn on Fan
Step 4:If Temperature <15 degrees then turn on the heater.
Step 5: End.

FLOWCHART:


START
Buy one temperature controller
Fix it on your home appliance
AC
If temperature controller showing->65 degree
YES
Turn on AC for chillness
STOP

d) Water Level Controller

ALGORITHM:

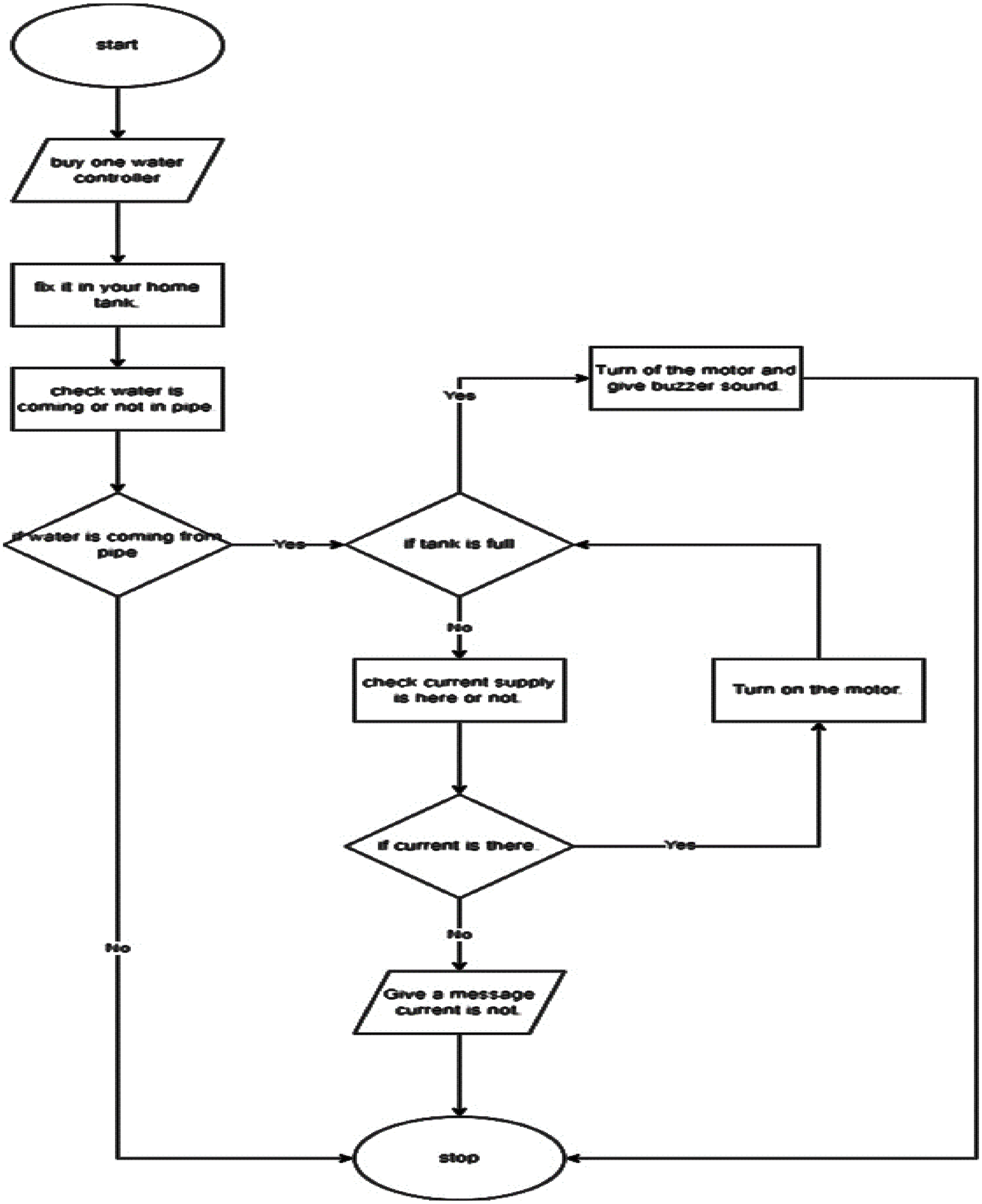
Step 1: Check if water connection is proper and water is flowing.

Step 2:Check if tank is FULL, then turn OFF Motor and give buzzer sound.

Step 3:else check the current supply, if yes, turn ON Motor

Step 4:If no power, then give alert “NO POWER”.

FLOWCHART:

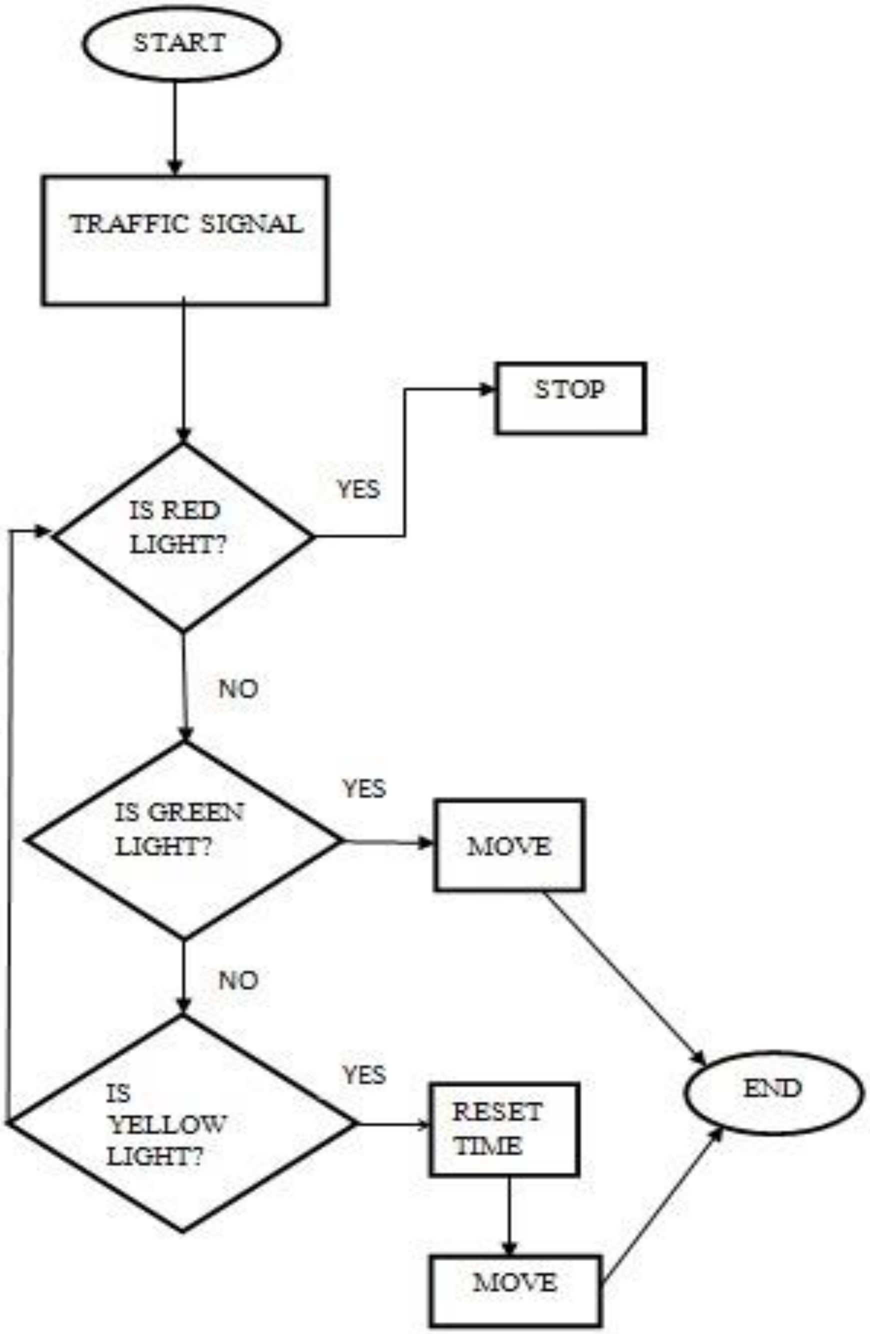

start
buy one water controller
fix it in your home tank.
check water is coming or not in pipe
if water is coming from pipe
Yes
No
if tank is full
Yes
Turn of the motor and give buzzer sound.
No
check current supply is here or not.
Turn on the motor.
if current is there.
Yes
No
Give a message current is not.
stop

e) **Traffic Signal Control**

ALGORITHM:

Step 1: Read the Traffic Signal
Step 2: If signal is Red , Command STOP
Step 3: If signal is Green, Command MOVE
Step 4:If signal is yellow, Reset Time and then Move
Step 5: End

FLOWCHART:

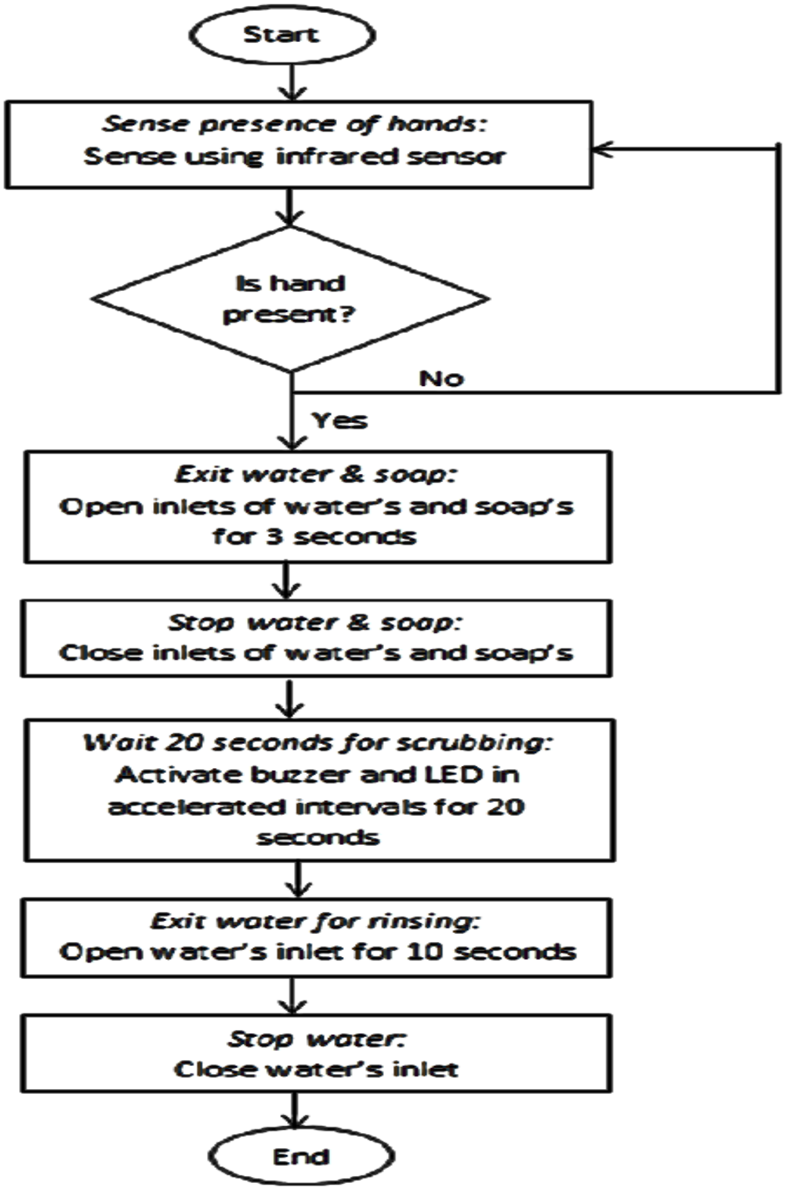


f) Automatic Washing Machine Control System

ALGORITHM:

Step 1: Start
Step 2: Sense the presence of hand using infrared sensor
Step 3: Fill water until certain level and add required soap solution.
Step 4: Close the water valve and soap.
Step 5: Activate the buzzer and LED After Scrubbing
Step 6: Remove the soapy water and open water valve for rinsing
Step 7: Close water valve and rinse.
Step 8: STOP

FLOWCHART:


Start
Sense presence of hands:
Sense using infrared sensor
Is hand present?
No
Yes
Exit water & soap:
Open inlets of water's and soap's for 3 seconds
Stop water & soap:
Close inlets of water's and soap's
Wait 20 seconds for scrubbing:
Activate buzzer and LED in accelerated intervals for 20 seconds
Exit water for rinsing:
Open water's inlet for 10 seconds
Stop water:
Close water's inlet
End

g) **AUTOMATIC STREET LIGHT CONTROL SYSTEM:**

ALGORITHM:

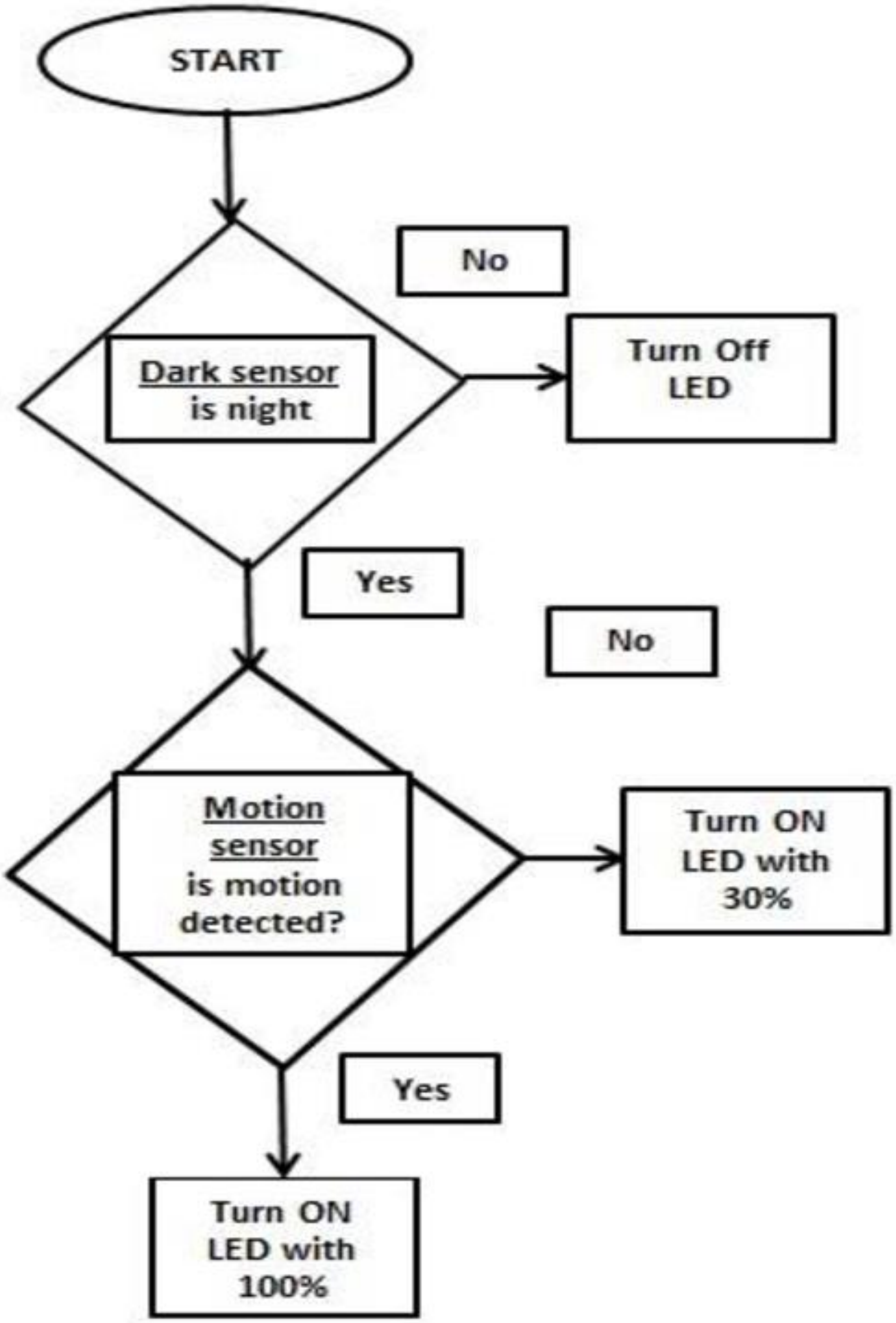
Step 1: Sense the Darkness with Sensor

Step 2: If Sensor detects Night, Turn ON Light, else Turn OFF Light

Step 3: If Motion Detected, then Turn ON LED with 100%.

Step 4: If Motion Detected is at Distance, then Turn ON LED with 30%.

FLOWCHART:

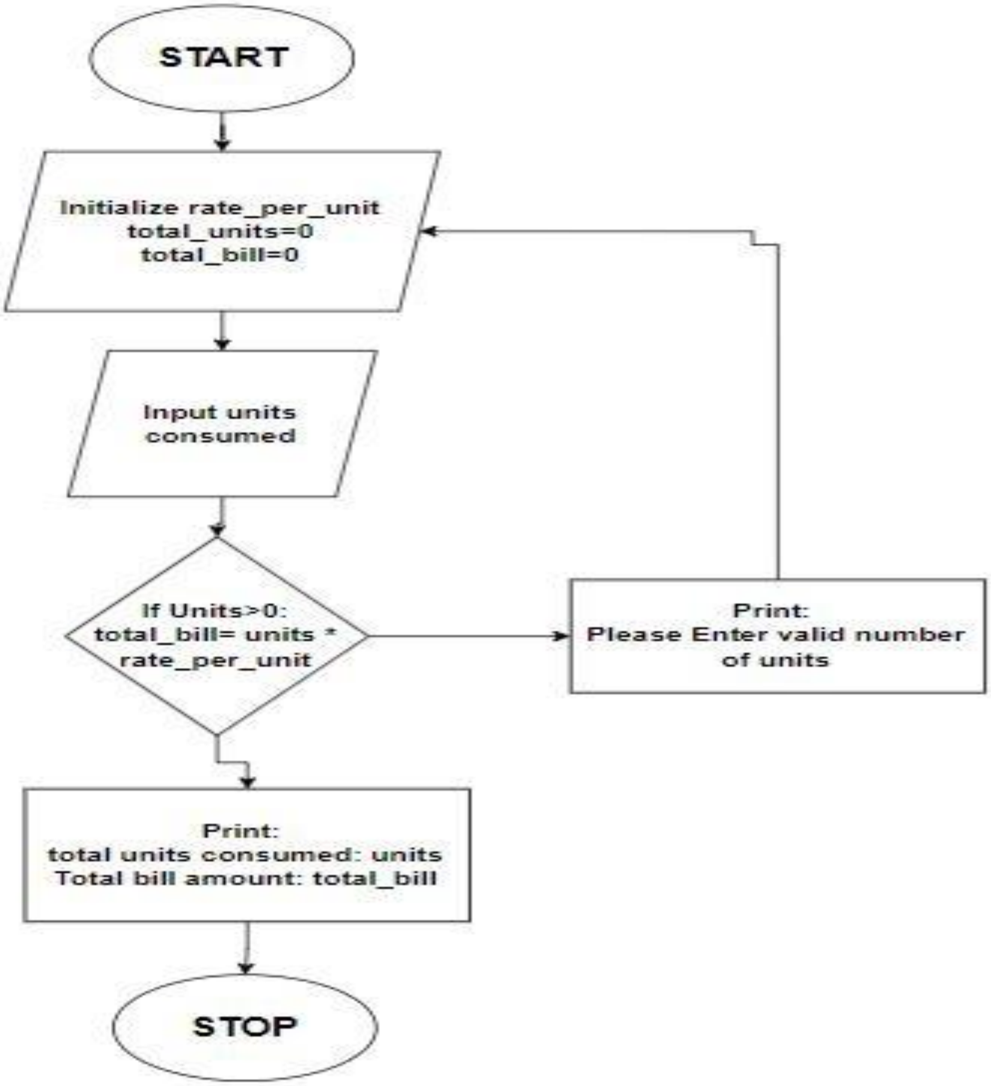


h) Electricity Billing

ALGORITHM:

```
Step 1: Initialize rate_per_unit to your fixed rate for electricity consumption.
Step 2. Initialize total_units to 0 and also Initialize total_bill to 0.
Step 3. Read input for units consumed (units).
Step 4. If units is greater than 0, then:
Step 5. Calculate total_bill = units * rate_per_unit.
Step 6. Display "Total units consumed: units", and "Total bill amount: total_bill".
Step 7. Else, Display "Invalid input:"
```

FLOWCHART:



RESULT:

Thus, the flow charts is developed and solved for the simple real-life or scientific or technical problems.

MODULE – II

Ex. No: 2a	**SWAP WITHOUT A TEMPORARY VARIABLE**
Date:	

AIM:

To write a python program to swap without a temporary variable.

ALGORITHM:

Step 1: Read the values of the two variables, a and b.
Step 2: Print the values of a and b before swapping.
Step 3: Perform the swap:
Step 4: Assign the sum of a and b to variable a.
Step 5: Assign the difference between the new value of a and b to variable b.
Step 6: Assign the difference between the new value of a and b to variable a again.
Step 7: Print the values of a and b after swapping.

PROGRAM:

```
# Read the values of a and b
a = int (input (“Enter the value of a:”))
b = int (input (“Enter the value of b:”))
# Print the values before swapping
print (“Before swapping: a =”, a, “b =”, b)
# Swap without temporary variable
a = a + b
b = a - b
a = a - b
# Print the values after swapping
Print (“After swapping: a =”, a,“b =”, b)
```

OUTPUT:

```
Enter the value of a: 5
Enter the value of b: 8
 Before swapping: a = 5   b = 8
After swapping: a = 8   b = 5
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE – II	
Ex. No: 2b	

Date:	**QUADRATIC EQUATION**

AIM:

To write a python program to solve the Quadratic Equation.

ALGORITHM:

Step 1: Read the coefficients a, b, and c of the quadratic equation.
Step 2: Calculate the discriminant, given by the formula: discriminant = b^2 - 4ac.
Step 3: If the discriminant is greater than zero, calculate the two solutions:

- solution1 = (-b - sqrt(discriminant)) / (2a)
- solution2 = (-b + sqrt(discriminant)) / (2a)

Step 4: If the discriminant is equal to zero, calculate a single solution:

- solution = -b / (2a)

Step 5: If the discriminant is negative, print that the equation has no real solutions.
Step 6: Print the solutions if they exist.

PROGRAM:

```
# Read the coefficients of the quadratic equation
a = float (input (“Enter the coefficient a:”))
b = float (input (“Enter the coefficient b:”))
 c = float (input (“Enter the coefficient c:”))
# Calculate the discriminant
discriminant = b**2 - 4*a*c
 # Check the value of the discriminant
if discriminant > 0:
    # Calculate two solutions
    solution1 = (-b - (discriminant**0.5)) / (2*a)
     solution2 = (-b + (discriminant**0.5)) / (2*a)
 print (“Two solutions exist:”)

print (“Solution 1:”, solution1)
      print (“Solution 2:”, solution2)
elif discriminant == 0:
    # Calculate a single solution
    solution = -b / (2*a)
    print (“One solution exists:”)
    print (“Solution:”, solution)
else:
    print (“No real solutions exist.”)
```

OUTPUT:

```
Enter the coefficient a: 1
Enter the coefficient b: -4
Enter the coefficient c: 3
One solution exists:
Solution: 3.0
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE – II

Ex. No: 2c

Date:

VALID PALINDROME

AIM:

To write a python program to solve the valid palindrome.

ALGORITHM:

Step 1: Read the input string.
Step 2: Initialize two pointers, start and end, pointing to the beginning and end of the string, respectively.
Step 3: Repeat the following until start is less than or equal to end:

- Skip non-alphanumeric characters at start and end.
- If the characters at indices start and end are not equal, the string is not a valid palindrome. Print the result and exit.
- Increment start by 1 and decrement end by 1.

Step 4: If the loop completes without any mismatched characters, the string is a valid palindrome.
Step 5: Print the result.

PROGRAM:

```
# Read the input string
string = input (“Enter a string: ”)
# Initialize pointers
start = 0
end = len(string) - 1
# Check for palindrome
is_palindrome = True
while start <= end:
    if not string[start]. isalnum ():
        start += 1
        continue
    if not string[end]. isalnum ():
        end -= 1
        continue
    if string[start]. lower ()! = string[end]. lower ():
        is_palindrome = False
        break
    start += 1
    end -= 1
# Print the result
if is_palindrome:
    print(“The string is a valid palindrome”)
else:
    print(“The string is not a valid palindrome”)
```

OUTPUT:

```
Enter a string: A man, a plan, a canal: Panama
The string is a valid palindrome.
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - II	
Ex. No: 2d	**INTEGER TO ROMAN LETTER**
Date:	

AIM:

To write a python program to convert Integer to Roman Letter.

ALGORITHM:

Step 1: Read the input integer.
Step 2: Initialize an empty string to store the Roman numeral representation.
Step 3: Convert the input integer to its Roman numeral representation using ternary operators:

- Divide the input integer by 1000 using integer division (//). Append ‘M’ repeated by the quotient to the result string. Set the input integer to the remainder.
- Check if the input integer is greater than or equal to 900. If true, append ‘CM’ to theresult string and subtract 900 from the input integer.
- Check if the input integer is greater than or equal to 500. If true, append ‘D’ to theresult string and subtract 500 from the input integer.
- Check if the input integer is greater than or equal to 400. If true, append ‘CD’ to theresult string and subtract 400 from the input integer.
- Divide the input integer by 100 using integer division (//). Append ‘C’ repeated bythe quotient to the result string. Set the input integer to the remainder.
- Check if the input integer is greater than or equal to 90. If true, append ‘XC’ to theresult string and subtract 90 from the input integer.
- Check if the input integer is greater than or equal to 50. If true, append ‘L’to theresult string and subtract 50 from the input integer.
- Check if the input integer is greater than or equal to 40. If true, append ‘XL’to theresult string and subtract 40 from the input integer.
- Divide the input integer by 10 using integer division (//). Append ‘X’ repeated by thequotient to the result string. Set the input integer to the remainder.
- Check if the input integer is greater than or equal to 9. If true, append ‘IX’ to theresult string and subtract 9 from the input integer.
- Check if the input integer is greater than or equal to 5. If true, append ‘V’ to theresult string and subtract 5 from the input integer.
- Check if the input integer is greater than or equal to 4. If true, append ‘IV’ to theresult string and subtract 4 from the input integer.
- Append ‘I’ repeated by the input integer to the result string.

Step 4: Print the Roman numeral representation.

PROGRAM:

```
# Read the input integer
         num = int (input (“Enter an integer (1-3999):”))
       # Roman numeral conversion
         result = “ ”
      # Thousands
         result += “M” * (num // 1000)
      num %= 1000
      # Hundreds
        result +=”CM” if num >= 900 else “D” + “C” * ((num % 900) // 100)
if num >= 400 else “C” * (num //100)
num %= 100
# Tens
result += “XC” if num >= 90 else “L” + “X” * ((num % 90) // 10) if num >= 40 else “X” * (num // 10)
num %= 10
# Units
result += “IX”if num >= 9 else “V” + “I” * ((num % 9) // 1)if num >= 4 else “I” * num
# Print the Roman numeral representation
print (“Roman numeral:”, result)
```

OUTPUT

```
Enter an integer (1-3999): 1990
Roman numeral: MCMXC
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE – III	
Ex. No: 3a	**CHECK WHETHER AN ALPHABET IS A VOWEL OR CONSONANT**
Date:	

AIM:

To write a python program to check whether an alphabet is a vowel or consonant..

ALGORITHM:

Step 1: Read the input alphabet.
Step 2: Convert the input alphabet to lowercase.
Step 3: Check if the input alphabet is equal to any of the vowel characters (‘a’, ‘e’, ‘i’, ‘o’, ‘u’)

- solution1 = (-b - sqrt(discriminant)) / (2a)
- solution2 = (-b + sqrt(discriminant)) / (2a)

Step 4: If the input alphabet is equal to any vowel character, it is a vowel. Print the result.
Step 5: If the input alphabet is not equal to any vowel character, it is a consonant. Print the result.

PROGRAM:

```
# Read the input alphabet
alphabet = input (“Enter an alphabet:”)
# Convert the alphabet to lowercase
alphabet = alphabet.lower()
# Check if the alphabet is a vowel or consonant
if alphabet == “a” or alphabet == “e” or alphabet == “i” or alphabet == “o” or alphabet == “u”:
    print(“The alphabet is a vowel.”)
else:
    print(“The alphabet is a consonant.”)
```

OUTPUT:

```
Enter an alphabet: E
The alphabet is a vowel.
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - III	
Ex. No: 3b **Date:**	**SUM OF ALL EVEN NUMBERS FROM 0 TO N**

AIM:

To write a python program to sum of all even numbers from 0 to n.

ALGORITHM:

```
Step 1: Read the input number n.
Step 2: Initialize a variable sum to store the sum of even numbers, starting from 0.
Step 3: Initialize a variable i to 0.
Step 4: Repeat the following until I is less than or equal to n:
            • If I is an even number, add i to sum.
            • Increment i by 2 to move to the next even number.
Step 5: Print the value of sum.
```

PROGRAM:

```
# Read the input number n
n = int(input("Enter a number: "))
# Calculate the sum of even numbers
sum = 0
i = 0
while i<= n:
   if i % 2 == 0:
       sum += i
   i += 2
# Print the sum of even numbers
print ("Sum of even numbers from 0 to", n, ":", sum)
```

OUTPUT:

```
Enter a number: 10
Sum of even numbers from 0 to 10: 30
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - III

Ex. No: 3c

Date:

FACTORIAL OF A NUMBER

AIM:

To write a python program to find the factorial of a number.

ALGORITHM:

```
Step 1: Read the input number n.
Step 2: Initialize a variable factorial to 1.
Step 3: Repeat the following until n is greater than 1:
        • Multiply factorial by n.
        • Decrement n by 1.
Step 4: Print the value of factorial.
```

PROGRAM:

```
# Read the input number
n = int(input("Enter a number: "))
# Calculate the factorial
factorial = 1
while n > 1:
    factorial *= n
    n -= 1
# Print the factorial
print ("Factorial:", factorial)
```

OUTPUT:

```
Enter a number: 5
Factorial: 120
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE – IV	
Ex. No: 4a	**MINIMUM INDEX SUM OF TWO LISTS**
Date:	

AIM:

To write a python program using lists to find the minimum Index Sum of Two Lists.

ALGORITHM:

```
Step 1: Create two empty lists e and a.
Step 2: Iterate through the indices i in the range of the length of list1.
        • Iterate through the indices j in the range of the length of list2.
        • Check if the element at list1[i] is equal to the element at list2[j].
        • If they are equal, calculate the sum of I and j and append it to the e list.
        • Append the element at list1[i] to the a list
Step 3: find the minimum value t in the e list.
Step 4: create an empty list.
Step 5: Iterate through the indices c in the range of the length of e.
        • Check if the element at e[c] is equal to t.
        • If it is equal, append the element at a[c] to the i list.
Step 6: Print the i list.
```

PROGRAM:

```
list1= [“happy”, “sad”, “good”]
    list 2=[“sad”, “happy”, “good”]
    e=[]
    a=[]
    for i in range(0, len(list1)):
      for j in range(0, len(list1)):
        if(list1[i]==list2[j]):
e. append(i+j)
           e. append(list1[i])
    t=min(e)
i=[]
    for c in range(0, len(e)):
      if(e[c]==t):
i.append(a[c])
     print(i)
```

OUTPUT:

```
['happy', 'sad']
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - IV	
Ex. No: 4b	**CONCATENATE TWO LISTS INDEX - WISE**
Date:	

AIM:

To write a python program using lists to concatenate two lists index – wise.

ALGORITHM:

Step 1: Create an empty lists i.
Step 2: Iterate through the indices i in the range of the length of a.

- Concatenate the element at index i from a with the element at the same index from b.
- Append the concatenated string to the a list

Step 3: Print the i list.

PROGRAM:

```
        a=["hello","welcome","thankyou"]
b=["welcome","hello","thankyou"]
l=[]
for i in range(0,len(a)):
l.append(a[i]+b[i])
print(l)
```

OUTPUT:

```
        ['hellowelcome', 'welcomehello', 'thankyouthankyou']
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - IV	
Ex. No: 4c	**TUPLE WITH SAME PRODUCT**
Date:	

AIM:

To write a python program using Tuples with the same product.

ALGORITHM:

Step 1: Create an empty lists tuples_list to store the tuples.
Step 2: Iterate through the range of numbers from 1 to n (inclusive) with the variable

- Iterate through the range of numbers from a to n (inclusive) with the variable
- Check if the product of a and b is equal to n.
- If the condition is true, create a tuple (a, b) and append it to the tuples_list.

Step 3: Return the tuples_list.

PROGRAM:

```
def find_tuple_with_same_product(n):
tuples_list = []
for a in range(1, n + 1):
for b in range(a, n + 1):
if a * b == n:
tuples_list.append((a, b))
return tuples_list
number = 20
result = find_tuple_with_same_product(number)
print(f"tuples with the product {number}: {result}")
```

OUTPUT:

```
tuples with the product 20:[(1, 20), (2, 10), (4, 5)]
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - IV

Ex. No: 4d

Date:

COPY SPECIFIC ELEMENTS FROM ONE TUPLE TO A NEW TUPLE

AIM:

To write a python program using Tuples to copy specific elements from one tuple to a new tuple.

ALGORITHM:

Step 1: Create an empty list new_tuple to store the selected elements.
Step 2: Iterate through each index i in the indices_to_copy list..

- Access the element at index i from the original_tuple using original_tuple[i].
- Append the element to the new_tuple list.

Step 3: Convert the new_tuple list into a tuple using the tuple() function.
Step 4:Print the new_tuple.

PROGRAM:

```
original_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
indices_to_copy = [2, 4, 6]
new_tuple = tuple(original_tuple[i] for i in indices_to_copy)
print(new_tuple)
```

OUTPUT:

```
(3, 5, 7)
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - V	
Ex. No: 5a	**MAGIC DICTIONARY**
Date:	

AIM:

To write a python program to implement Magic Dictionary.

ALGORITHM:

Step 1: Create a class MagicDictionary with an empty dictionary self.dictionary as an instance variable.
Step 2: Implement the add_word method:

- Take a word as input.
- Get the length of the word.
- Check if the length exists as a key in the dictionary.
- If it doesn't exist, add a new key-value pair where the key is the length and the value is an empty list.
- Append the word to the list corresponding to the length in the dictionary.

Step 3: Implement the search method:

- Take a word as input.
- Get the length of the word.
- Check if the length exists as a key in the dictionary.
- If it doesn't exist, return False as there are no words of that length.
- Get the list of words of the same length from the dictionary.
- Iterate through each word in the list.
- initialize a variable diff_count to track the number of differing characters between the given word and the current word from the list.
- Iterate through each character index in the range of the length of the word.
- Compare the characters at the corresponding indices in both words.
- If the characters are different, increment diff_count by 1.
- If diff_count exceeds 1, break out of the loop to optimize the search.
- If diff_count is exactly 1, return True as a valid match is found
- If no valid match is found, return False.

Step 4:Create an instance of the MagicDictionary class called magic_dict.
Step 5:Use the add_word method to add words to the magic dictionary.
Step 6:Use the search method to search for words in the magic dictionary and print the results

PROGRAM:

```
class Magic Dictionary:
        def __init__(self):
      self.dictionary = {}
   def add_word(self, word):
      length = len(word)
      if length not in self.dictionary:
          self.dictionary[length] = []
      self.dictionary[length].append(word)
   def search(self, word):
      length = len(word)
      if length not in self.dictionary:
          return False
      words = self.dictionary[length]
      for w in words:
          diff_count = 0
          for i in range(length):
              if w[i] != word[i]:
                  diff_count += 1
              if diff_count> 1:
                  break
          if diff_count == 1:
              return True
      return False

# Example usage:
magic_dict = MagicDictionary()
magic_dict.add_word("hello")
magic_dict.add_word("world")
magic_dict.add_word("python")
magic_dict.add_word("magic")
print(magic_dict.search("hella"))  # True
print(magic_dict.search("hillo"))  # False
print(magic_dict.search("python"))  # False

print(magic_dict.search("magik"))  # True
```

OUTPUT:

```
True
True
False
True
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - V	
Ex. No: 5b	**LONGEST WORD IN DICTIONARY**
Date:	

AIM:

To write a python program to find the longest word in a Dictionary.

ALGORITHM:

Step 1: Define a function longest_word that takes a dictionary word_dict as input.
Step 2: Initialize a variable longest to an empty string.
Step 3: Iterate through the values of the dictionary using a loop.

- For each word in the dictionary values, check if its length is greater than the length of the current longest word.
- If the length of the word is greater, update the value of longest to the current word.

Step 4: After iterating through all the words, return the value of longest.
Step 5:In the main program:

- Create an empty dictionary dict1.
- Prompt the user to enter values for the dictionary using a loop.
- Assign each input value to the corresponding key in dict1.
- Call the longest_word function with dict1 as an argument and assign the result to ytlongest.
- Print the value of ytlongest as the longest word.

PROGRAM:

```
def longest_word(word_dict):

longest = "

for word in word_dict.values():

if len(word) > len(longest):

longest = word


return longest

# Example usage:

dict1 = {}

print("Enter the values for the dictionary:")

for i in range(0,5):

dict1[i]=input()
```

```
longest=longest_word(dict1)
print("Longest word:", longest)
```

OUTPUT:

```
Enter the values for the dictionary:
computer science
database
machine learning
artificial intelligence
cloud computing
Longest word: artificial intelligence
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE –V	
Ex. No: 5c	**SET MISMATCH**
Date:	

AIM:

To write a python program to find the set mismatch.

ALGORITHM:

Step 1: Create a function findErrorNums that takes a list nums as input.
Step 2: Get the length of the nums list and assign it to the variable n.
Step 3:Create an empty set num_set to store unique numbers.
Step 4: Initialize a variable duplicate to 0 to track the duplicate number.
Step 5:Calculate the total sum of all numbers in the nums list using the sum() function and assign it to the variable total_sum.
Step 6: Iterate through each number num in the nums list.

- Check if the current number num is already present in the num_set.
- If it is, assign the num value to the duplicate variable.
- Add the num to the num_set.

Step 7: Calculate the missing number using the formula: ((n * (n + 1)) // 2) - (total_sum - duplicate).
Step 8: Return a list containing the duplicate number and the missing number as [duplicate, missing].

PROGRAM:

```
def findErrorNums(nums):
n = len(nums)
num_set = set()
 duplicate = 0
total_sum = sum(nums)
for num in nums:


  if num in num_set:
 duplicate = num
num_set.add(num)
missing = ((n * (n + 1)) // 2) - (total_sum - duplicate)
return [duplicate, missing]


# Test cases
```

```
nums1 = [1, 2, 3, 3]
print(findErrorNums(nums1))  # Output: [2, 3]
nums2 = [1, 2, 2]
print(findErrorNums(nums2))  # Output: [1, 2]
```

OUTPUT:

```
[3, 4]
 [2, 3]
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE – V	
Ex. No: 5d	**SMALLEST NUMBER IN FINITE SET**
Date:	

AIM:

To write a python program to find the Smallest Number in finite set.

ALGORITHM:

Step 1: Create a set number_set with the given values.
Step 2: Find the smallest number in the set using the min() function and assign it to the variable smallest_number.
Step 3:Print the value of smallest_number along with an appropriate message.

PROGRAM:

```
number_set = {5, 2, 9, 1, 7}

smallest_number = min(number_set)

print("Smallest number:", smallest_number)
```

OUTPUT:

```
Smallest number: 1
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VI	
Ex. No: 6a	**FACTORIAL**
Date:	

AIM:

To write a python program to implement the functions in the factorial program

ALGORITHM:

```
Step 1: Start with a positive integer n.
Step 2: If n is 0 or 1, return 1 (as the factorial of 0 and 1 is defined as 1).
    Step 3:Otherwise, recursively calculate the factorial by multiplying n with the factorial of n - 1.
 Step 4:Repeat step 3 until the base case is reached.
```

PROGRAM:

```
 def factorial(n):
 if n == 0 or n == 1:
   return 1
else:
  return n * factorial(n - 1)
print(factorial(5))
```

OUTPUT:

```
120
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VI	
Ex. No: 6b	**LARGEST NUMBER IN A LIST**
Date:	

AIM:

To write a python program to implement the functions in the largest number in a list

ALGORITHM:

Step 1: Start with a list of numbers.
Step 2: Initialize a variable largest with the first element of the list.
Step 3:Iterate through each element in the list.
Step 4:If the current element is greater than largest, update largest to the current element.
Step 5: Repeat steps 3-4 for each element in the list.
Step 6:After iterating through all elements, largest will contain the largest number in the list.

PROGRAM:

```
def find_largest_number(numbers):
 if len(numbers) == 0:
return None
 largest = numbers[0]
for number in numbers:
if number > largest:
 largest = number
return largest
numbers = [5, 8, 2, 10, 3]
print(find_largest_number(numbers))
```

OUTPUT:

```
10
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VI

Ex. No: 6c

Date:

AREA OF SHAPE

AIM:

To write a python program to implement the functions in the largest number in a list

ALGORITHM:

Step 1: Start with the name of the shape and the required parameters for calculating its area.
Step 2: Convert the shape name to lowercase for case-insensitive comparison.
Step 3:If the shape is a circle, calculate the area using the formula: area = pi * radius^2, where pi is a constant and radius is the provided parameter.
Step 4:If the shape is a rectangle, calculate the area using the formula: area = length * width, where length and width are the provided parameters.
Step 5: If the shape is a triangle, calculate the area using the formula: area = 0.5 * base * height, where base and height are the provided parameters.
Step 6: If the shape is not recognized or supported, return None to indicate an invalid shape.

PROGRAM:

```
import math
def calculate_area(shape, *args):
shape = shape.lower()
if shape == "circle":
radius = args[0]
return math.pi * radius**2
elif shape == "rectangle":
length = args[0]
width = args[1]
 return length * width

elif shape == "triangle":
base = args[0]
height = args[1]
 return 0.5 * base * height
else:
return None
print(calculate_area("circle", 5))
print(calculate_area("rectangle", 4, 6))
print(calculate_area("triangle", 3, 8))
print(calculate_area("square", 5))
```

OUTPUT:

```
78.53981633974483
24
12.0
None (shape not supported)
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE – VII	
Ex. No: 7a	**DETERMINE IF STRING HALVES ARE ALIKE**
Date:	

AIM:

To write a python program to determine if string halves are alike.

ALGORITHM:

```
Step 1: Start with a string s.
Step 2: Create a set of vowels.
Step 3:Calculate the midpoint index of the string by dividing the length of s by 2.
Step 4:Split s into two halves, first_half and second_half, based on the calculated midpoint index.
Step 5:Count the number of vowels in first_half and second_half using a loop or list comprehension.
Step 6: Return True if the number of vowels in first_half is equal to the number of vowels in second_half, otherwise return False.
```

PROGRAM:

```
def halves_are_alike(s):
vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
mid = len(s) // 2
 first_half = s[:mid]
second_half = s[mid:]
first_half_vowel_count = sum(1 for char in first_half if char in vowels)
second_half_vowel_count = sum(1 for char in second_half if char in vowels)
return first_half_vowel_count == second_half_vowel_count
print(halves_are_alike("book"))
print(halves_are_alike("textbook"))
```

OUTPUT:

```
 True
`False
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VII	
Ex. No: 7b	PALINDROME
Date:	

AIM:

To write a python program to check the string is Palindrome

ALGORITHM:

Step 1: Start with a string s.
Step 2: Convert s to lowercase.
Step 3:Remove non-alphanumeric characters from s and store the result in alphanumeric_chars.
. Step 4: Create a reversed version of alphanumeric_chars and store it in reversed_chars.
Step 5:Compare alphanumeric_chars with reversed_chars.
Step 6: Return True if alphanumeric_chars is equal to reversed_chars, indicating a palindrome. Otherwise, return False.

PROGRAM:

```
def is_palindrome(s):
s = s.lower()
alphanumeric_chars = [char for char in s if char.isalnum()]
reversed_chars = alphanumeric_chars[::-1]
return alphanumeric_chars == reversed_chars
print(is_palindrome("level"))
print(is_palindrome("Hello"))
```

OUTPUT:

```
True
False
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VII	
Ex. No: 7c	**CHARACTER COUNT**
Date:	

AIM:

To write a python program to count the characters.

ALGORITHM:

Step 1: Start with a string s.
Step 2: Create an empty dictionary character_count to store the count of each character.
Step 3:Iterate through each character char in s.
. Step 4:If char is already a key in character_count, increment its value by 1
Step 5:Otherwise, add char as a key to character_count with a value of 1.
Step 6: Return character_count dictionary containing the count of each character in s.

PROGRAM:

```
def count_characters(s):
character_count = {}
for char in s:
if char in character_count:
character_count[char] += 1
else:
 character_count[char] = 1
return character_count
print(count_characters("hello"))
```

OUTPUT:

```
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VII

Ex. No: 7d

Date:

REPLACE CHARACTER

AIM:

To write a python program to replace the characters in a string.

ALGORITHM:

```
Step 1: Start with a string s, and two characters old and new.
Step 2: Use the replace() method of the string s to replace all occurrences of old with new.
Step 3:Return the modified string s.
```

PROGRAM:

```
def replace_characters(s, old, new):
 return s.replace(old, new)
print(replace_characters("Hello, World!", "o", "e"))
```

OUTPUT:

```
Helle, werld!
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE –VIII	
Ex. No: 8a	COPY FROM ONE FILE TO ANOTHER
Date:	

AIM:

To write a python program to copy from one file to another.

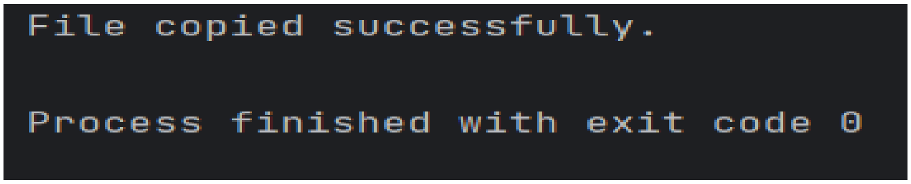
ALGORITHM:

Step 1: The file test. txt is opened using the open() function.
Step 2: Another file out. txt is opened using the open() function in the write mode using the f1 stream.
Step 3: Each line in the file is iterated over using a for loop (in the input stream).
Step 4: Each of the iterated lines is written into the output file.

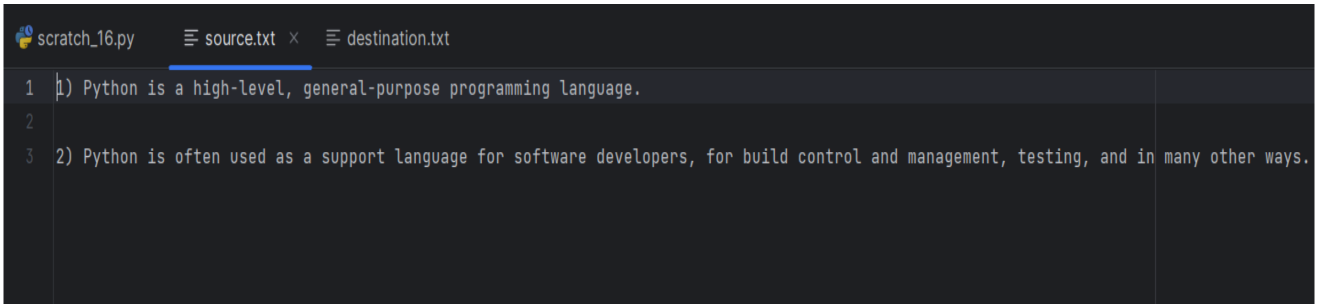
PROGRAM:

```
def copy_file(source_file, destination_file):
        try:
          with open(source_file, 'r') as source:
            with open(destination_file, 'w') as destination:
              for line in source:
                destination.write(line)
          print("File copied successfully.")
        except FileNotFoundError:
          print("Error: Source file not found.")
        except Exception as e:
          print("An error occurred:", str(e))
      # Example usage
      source_file = 'source.txt'
      destination_file = 'destination.txt'
      copy_file(source_file, destination_file)
```

OUTPUT:

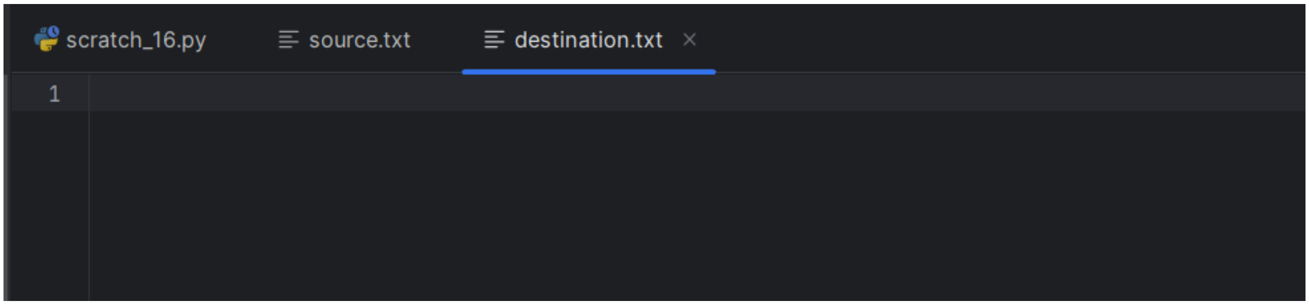


```
File copied successfully.

Process finished with exit code 0
```

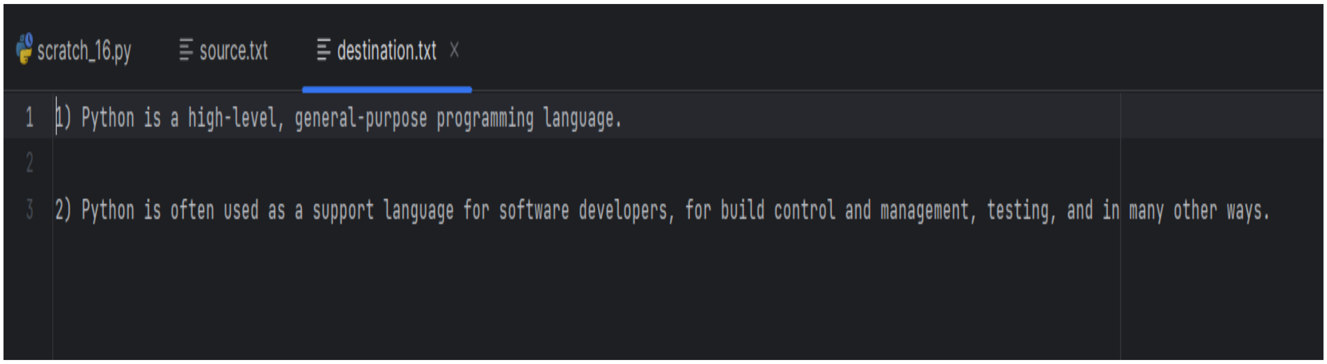


```
1) Python is a high-level, general-purpose programming language.

2) Python is often used as a support language for software developers, for build control and management, testing, and in many other ways.
```

Initial File



Final File



```
1) Python is a high-level, general-purpose programming language.

2) Python is often used as a support language for software developers, for build control and management, testing, and in many other ways.
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VIII	
Ex. No: 8b	**WORD COUNT**
Date:	

AIM:

To write a python program to count the word.

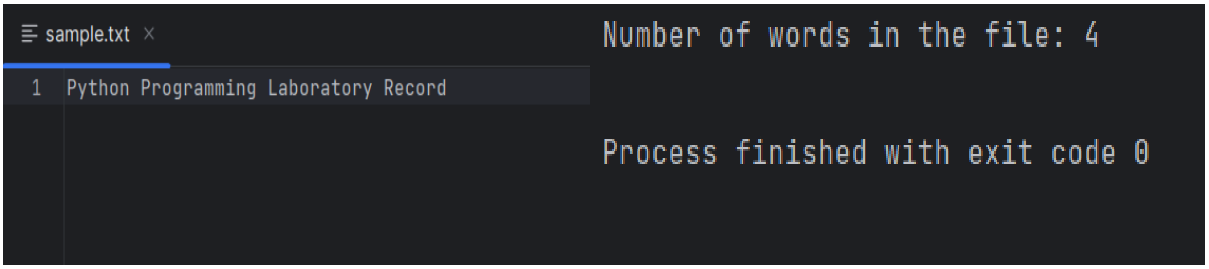
ALGORITHM:

Step 1: Initialize the input string
Step 2: Print the original string.
Step 3: Use the 'char. Split ()' method to count the number of spaces in the string and add 1 to it to get the count of words.
Step 4: Print the count of words.

PROGRAM:

```
def count_words(file_path):
    try:
        with open(file_path, 'r') as file:
            content = file.read()
            word_count = len(content.split())
            print("Number of words in the file:", word_count)
    except FileNotFoundError:
        print("Error: File not found.")
    except Exception as e:
        print("An error occurred:", str(e))

# Example usage
file_path = 'sample.txt'
count_words(file_path)
```

OUTPUT:



```
Python Programming Laboratory Record
```

```
Number of words in the file: 4

Process finished with exit code 0
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE - VIII	
Ex. No: 8c **Date:**	**LONGEST WORD IN A FILE**

AIM:

To write a python program to find the longest word in a file.

ALGORITHM:

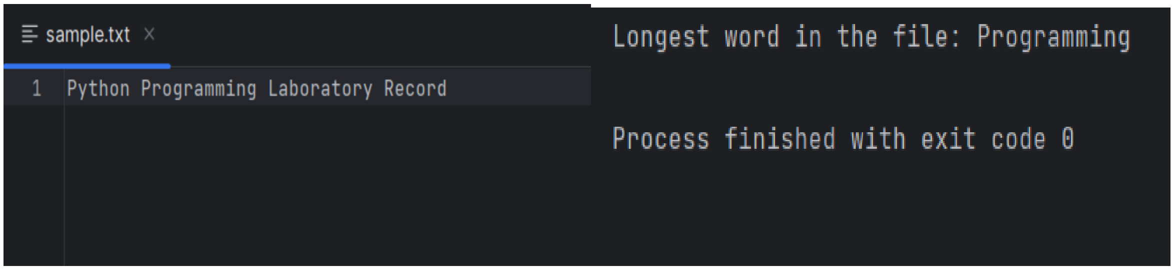
Step 1: Crcate a variable to read the text file data using the read() function (reads the specified number of bytes from the file and returns them. ...

Step 2: Find the length of the longest word using the len() (The number of items in an object is returned by the len() method.

PROGRAM:

```
def find_longest_word(file_path):
    try:
        with open(file_path, 'r') as file:
            content = file.read()
            words = content.split()
            longest_word = max(words, key=len)
            print("Longest word in the file:", longest_word)
    except FileNotFoundError:
        print("Error: File not found.")
    except Exception as e:
        print("An error occurred:", str(e))
# Example usage
file_path = 'sample.txt'
find_longest_word(file_path)
```

OUTPUT:



```
Python Programming Laboratory Record
```

```
Longest word in the file: Programming

Process finished with exit code 0
```

RESULT:

Thus, the python program is executed and successfully verified.

MODULE –IX	
Ex. No: 9 **Date:**	**PYTHON LIBRARIES (PANDAS / NUMPY / MATPLOTLIB)**

AIM

To implement the Libraries of Numpy, Pandas and MatPlotLib

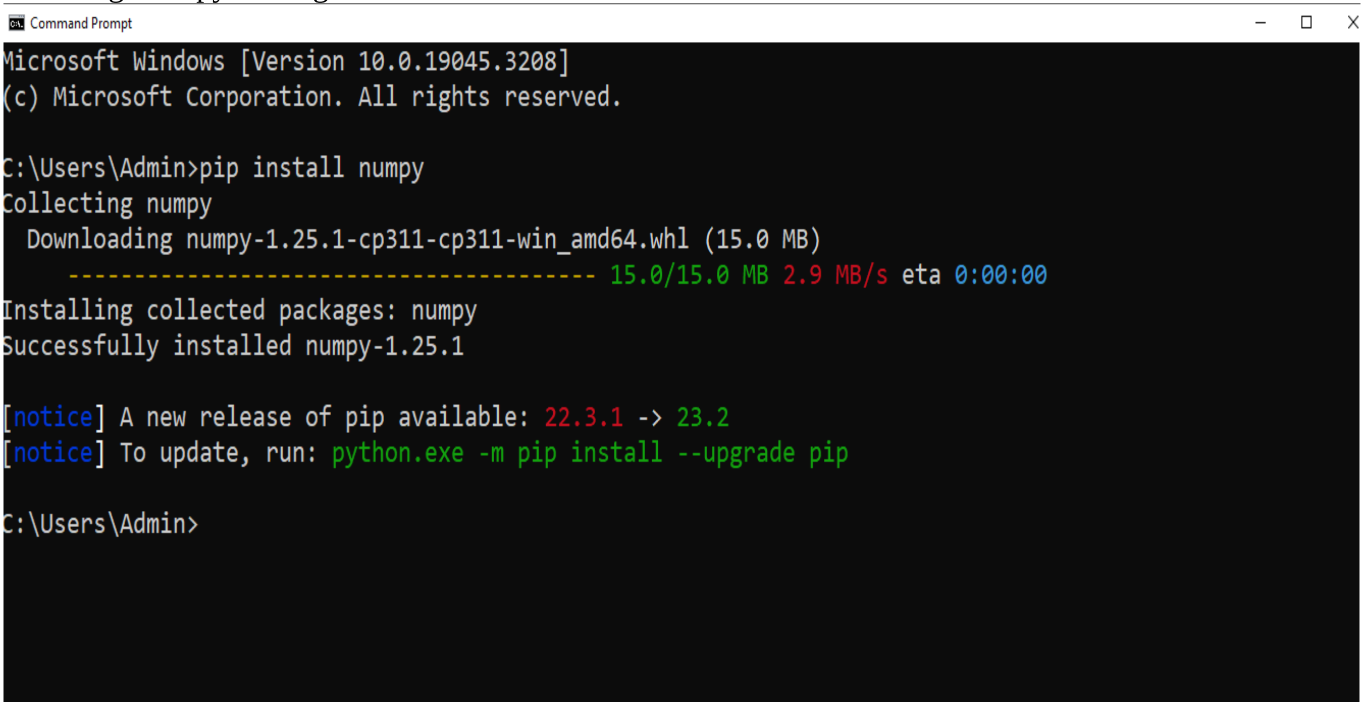
ALGORITHM

1. Install the Numpy Package in IDLE
2. Import the numpy package and create numpy arrays
3. Install the Pandas Pacakge in IDLE
4. Import the Pandas package and create series and dataframe
5. Install MatplotLib Package in IDLE
6. Import the MatplotLib Pyplot package and create simple plots.

PROGRAM

1. Working with Numpy

Installing Numpy Package



```
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>pip install numpy
Collecting numpy
  Downloading numpy-1.25.1-cp311-cp311-win_amd64.whl (15.0 MB)
     ---------------------------------------- 15.0/15.0 MB 2.9 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.25.1

[notice] A new release of pip available: 22.3.1 -> 23.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Admin>
```

Creating a simple array

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
```

```
print(arr)
```

OUTPUT

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Creating Different Arrays

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

OUTPUT

```
0
1
2
3
```

Array Slicing

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

OUTPUT

```
[2  3  4  5]
```

Array Reshaping

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(4, 3)

print(newarr)
```

OUTPUT

```
[[ 1  2  3]
```

```
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

Joining Arrays

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
```

OUTPUT

```
[1 2 3 4 5 6]
```

Array Split

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6])

newarr = np.array_split(arr, 3)

print(newarr)
[array([1, 2]), array([3, 4]), array([5, 6])]
```

Array Searching:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 4, 4])

x = np.where(arr == 4)

print(x)
```

OUTPUT

```
(array([3, 5, 6]),)
```

Array Sorting

```
import numpy as np

arr = np.array([3, 2, 0, 1])

print(np.sort(arr))
```

OUTPUT

```
[0 1 2 3]
```

Array Filtering

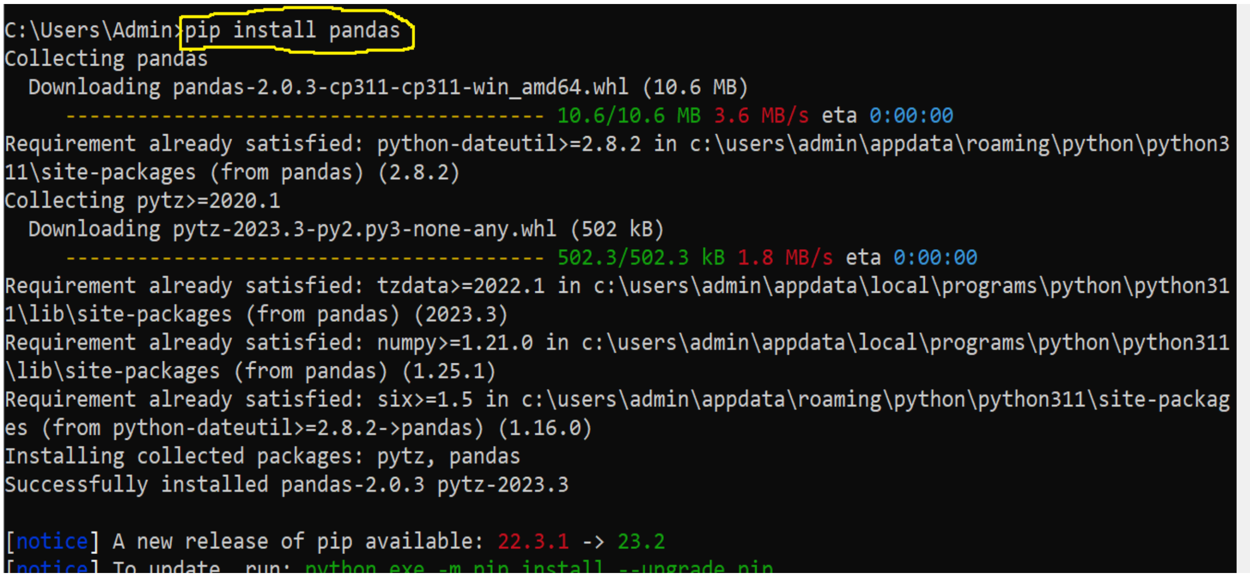
```
import numpy as np

arr = np.array([41, 42, 43, 44])

x = [True, False, True, False]

newarr = arr[x]

print(newarr)
```

OUTPUT

```
[41 43]
```

2. Working wiht PANDAS

Install Pandas in IDLE



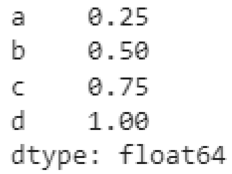
```
C:\Users\Admin>pip install pandas
Collecting pandas
  Downloading pandas-2.0.3-cp311-cp311-win_amd64.whl (10.6 MB)
     ---------------------------------------- 10.6/10.6 MB 3.6 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\admin\appdata\roaming\python\python3
11\site-packages (from pandas) (2.8.2)
Collecting pytz>=2020.1
  Downloading pytz-2023.3-py2.py3-none-any.whl (502 kB)
     ---------------------------------------- 502.3/502.3 kB 1.8 MB/s eta 0:00:00
Requirement already satisfied: tzdata>=2022.1 in c:\users\admin\appdata\local\programs\python\python31
1\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\admin\appdata\local\programs\python\python311
\lib\site-packages (from pandas) (1.25.1)
Requirement already satisfied: six>=1.5 in c:\users\admin\appdata\roaming\python\python311\site-packag
es (from python-dateutil>=2.8.2->pandas) (1.16.0)
Installing collected packages: pytz, pandas
Successfully installed pandas-2.0.3 pytz-2023.3

[notice] A new release of pip available: 22.3.1 -> 23.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Working with Pandas:

2.1 SERIES
CREATING A SERIES USING PANDAS

```
import pandas as pd
data = pd.Series([0.25, 0.5, 0.75, 1.0],
                 index=['a', 'b', 'c', 'd'])
data
```

OUTPUT



```
a    0.25
b    0.50
c    0.75
d    1.00
dtype: float64
```

Data Selection in Series

```
data['b']
```

OUTPUT

```
0.5
```

```
'a' in data
```

Output

```
True
```

```
data.keys()
```

OUTPUT

```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

```
data[0:2]
```

OUTPUT

```
a    0.25
b    0.50
dtype: float64
```

```
data[(data > 0.3) & (data < 0.8)]
b    0.50
c    0.75
dtype: float64
```

```
data = pd.Series(['a', 'b', 'c'], index=[1, 3, 5])
data
```

OUTPUT

```
1    a
3    b
5    c
dtype: object
```

```
data.loc[1:3]
```

OUTPUT

```
1    a
3    b
dtype: object
data.iloc[1:3]
```

Output

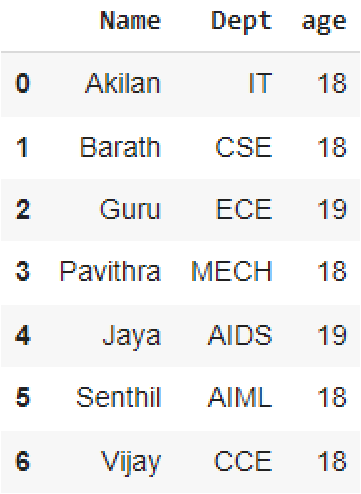
```
3    b
5    c
dtype: object
```

2.2 DATAFRAME

Creating a Dataframe a dictionary

```
import pandas as pd

dict={"Name":["Akilan","Barath","Guru","Pavithra","Jaya","Senthil","Vijay"],"Dept":["IT","CSE","ECE","MECH","AIDS","AIML","CCE"],"age":[18,18,19,18,19,18,18]}

df=pd.DataFrame(dict)
df
```

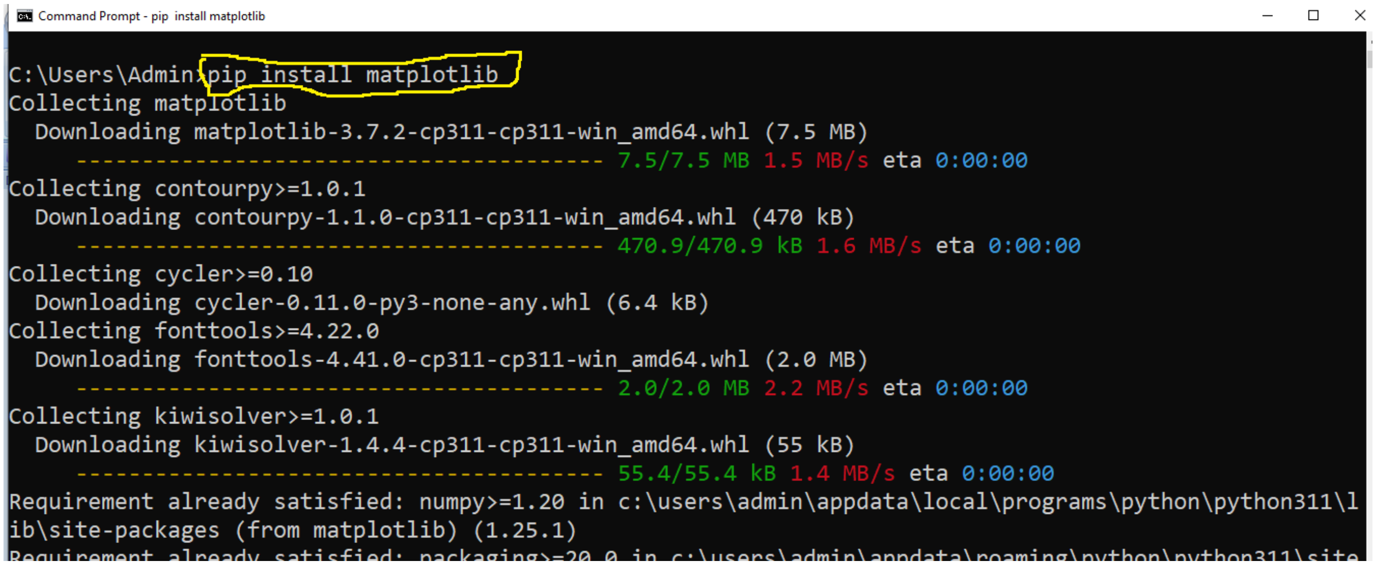
OUTPUT



	Name	Dept	age
0	Akilan	IT	18
1	Barath	CSE	18
2	Guru	ECE	19
3	Pavithra	MECH	18
4	Jaya	AIDS	19
5	Senthil	AIML	18
6	Vijay	CCE	18

3. MatplotLib

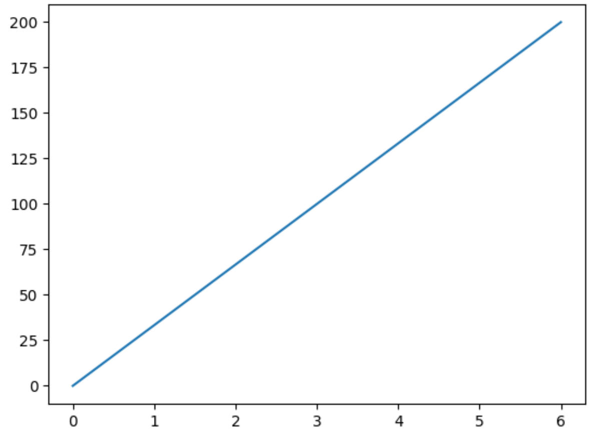
Installing MatplotLib



```
C:\Users\Admin>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.7.2-cp311-cp311-win_amd64.whl (7.5 MB)
     ---------------------------------------- 7.5/7.5 MB 1.5 MB/s eta 0:00:00
Collecting contourpy>=1.0.1
  Downloading contourpy-1.1.0-cp311-cp311-win_amd64.whl (470 kB)
     ---------------------------------------- 470.9/470.9 kB 1.6 MB/s eta 0:00:00
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.41.0-cp311-cp311-win_amd64.whl (2.0 MB)
     ---------------------------------------- 2.0/2.0 MB 2.2 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp311-cp311-win_amd64.whl (55 kB)
     ---------------------------------------- 55.4/55.4 kB 1.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.20 in c:\users\admin\appdata\local\programs\python\python311\l
ib\site-packages (from matplotlib) (1.25.1)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\appdata\roaming\python\python311\site
```

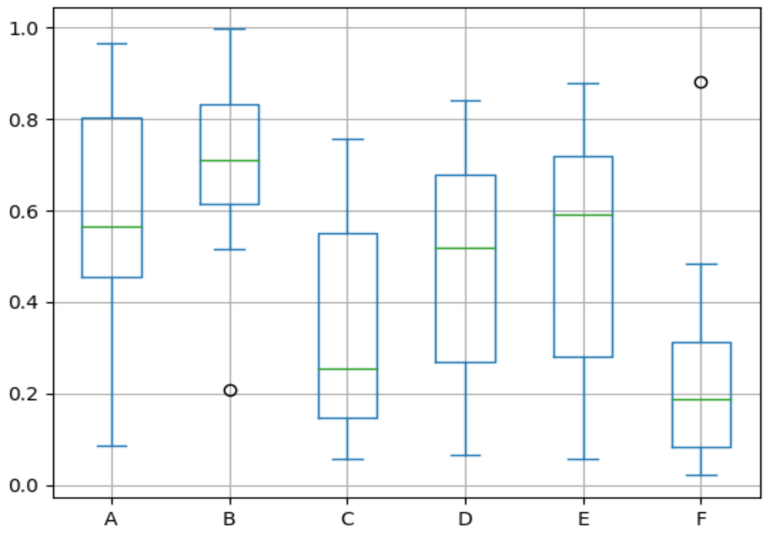
3.1 CREATE A LINE PLOT

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 200])

plt.plot(xpoints, ypoints)
plt.show()
```



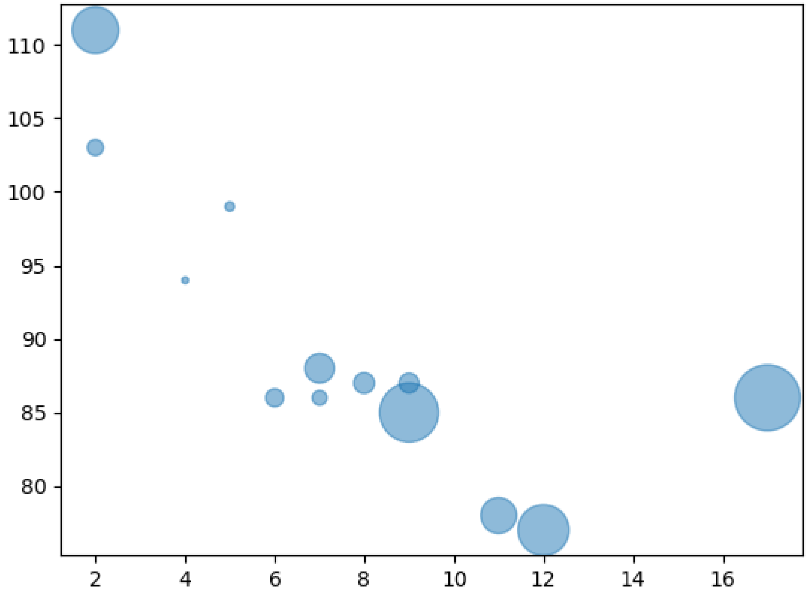
3.2 CREATING A BOX PLOT

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(8, 6), columns=['A', 'B', 'C', 'D', 'E','F'])
df.plot.box(grid='True')
```



3.3 CREATING A SCATTER PLOT

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes, alpha=0.5)
plt.show()
```



RESULT:

Thus the Libraries of Numpy, Pandas, and MatplotLIb are installed and implemented the arrays, series, dataframe and plot successfully.

MODULE – X	
Ex. No: 10 **Date:**	**IMPLEMENTATION OF APPLICATIONS OF STANDARD LIBRARIES**

AIM:

To implement different applications using the libraries Numpy, Pandas and MatplotLib

ALGORITHM:

1. Import Numpy Package
2. Create two different numpy arrays with different dimensions
3. Perform arithmetic operations on arrays.
4. Insert element in random position in an array.
5. Perform union of two numpy arrays.
6. Import Pandas package
7. Convert the numpy array into dataframe.
8. Create a Series, convert the series into dataframe
9. Create a dataframe multiple series.
10. Import the MatplotLib Pacakge
11. Convert a plot from a dataset.

PROGRAM:

1. Handling Scalar Values in Numpy Arrays

```
import numpy as np
a=np.array([[1,2,3],[1,2,1]])
b=np.array([1,2,4])
a+b
```

OUTPUT:

```
array([[2, 4, 7],

 [2, 4, 5]])
```

```
 import numpy as np
a=np.array([[1,2,3],[4,5,6]])
b=3
a+b
```

OUTPUT

```
array([[4, 5, 6],
       [7, 8, 9]])
```

```
 import numpy as np
a=np.array([[1,2],[1,2]])
b=np.array([[2,4],[3,4]])
a.dot(b)
```

OUTPUT

```
array([[ 8, 12],
       [ 8, 12]])
```

```
import numpy as np
a=np.array([[1,2,4],
            [1,2,5],
            [4,5,6]])
a[-1][-2]
```

OUTPUT

```
5
```

2. Insert values at random positions in an array

```
import random
# initializing list
test_list = [5, 7, 4, 2, 8, 1]
 # printing original list
```

```
print("The original list : " + str(test_list))
# initializing add list

add_list = ["Gfg", "Best", "CS"]
# initializing K
K = 3
for idx in range(K):
   # choosing index to enter element
    index = random.randint(0, len(test_list))

    # reforming list and getting random element to add
    test_list = test_list[:index] + [random.choice(add_list)] + test_list[index:]
# printing result
print("The created List : " + str(test_list))
```

OUTPUT

```
The original list : [5, 7, 4, 2, 8, 1]
The created List : [5, 'Gfg', 7, 4, 'Best', 2, 'CS', 8, 1]
```

3. Convert the index of a series into a column of a Dataframe

```
import pandas as pd

# Creating Series of
# programming languages
s = pd.Series(['C', 'C++', 'Java',
               'Python', 'Perl', 'Ruby',
               'Julia'])
s
df = s.to_frame().reset_index()

# show the dataframe
df
```

OUTPUT

```
     index       0
 0       0       C
 1       1     C++
 2       2    Java
 3       3  Python
 4       4    Perl
 5       5    Ruby
 6       6   Julia
```

4. Create DataFrame from multiple Series

```
import pandas as pd

# Create pandas Series
courses = pd.Series(["Spark","PySpark","Hadoop"])
fees = pd.Series([22000,25000,23000])
discount  = pd.Series([1000,2300,1000])

# Combine two series.
df=pd.concat([courses,fees],axis=1)

# It also supports to combine multiple series.
df=pd.concat([courses,fees,discount],axis=1)
print(df)
```

OUTPUT

```
       0      1     2
0   Spark  22000  1000
```

```
1  PySpark  25000  2300
2   Hadoop  23000  1000
```

5. Get Frequency counts of unique items of a series

```
import pandas as pd

technologies = {

    'Courses':["Spark","PySpark","Hadoop","Python","pandas","PySpark","Python","pandas"],

    'Fee' :[24000,25000,25000,24000,24000,25000,25000,24000]


   }

df = pd.DataFrame(technologies)


df1 = df['Courses'].value_counts()

print(df1)
```

OUTPUT

```
PySpark    2

Python     2

pandas     2

Spark      1

Hadoop     1

Name: Courses, dtype: int64
```

6. Union of Two Numpy Arrays

```
import numpy as np

  # 2-d array

arr1 = np.array([[1, 2, 3], [4, 5, 6]])

print("array1 ")

print(arr1)


arr2 = np.array([0, 5, 10])

print("array2 ", arr2)
```

```
# print union of 2-d array and 1-d array
print("Union of two arrays", np.union1d(arr1, arr2))
array1
[[1 2 3]
 [4 5 6]]
array2  [ 0  5 10]
Union of two arrays [ 0  1  2  3  4  5  6 10]
```

7. Creation of Dataframe from a numpy array

```
import numpy as np
import pandas as pd

# creating a numpy array
numpyArray = np.array([[15, 22, 43],
                       [33, 24, 56]])

# generating the Pandas dataframe
# from the Numpy array and specifying
# name of index and columns
panda_df = pd.DataFrame(data = numpyArray,
                        index = ["Row_1", "Row_2"],
                        columns = ["Column_1",
                                   "Column_2", "Column_3"])

# printing the dataframe
print(panda_df)
```

OUTPUT

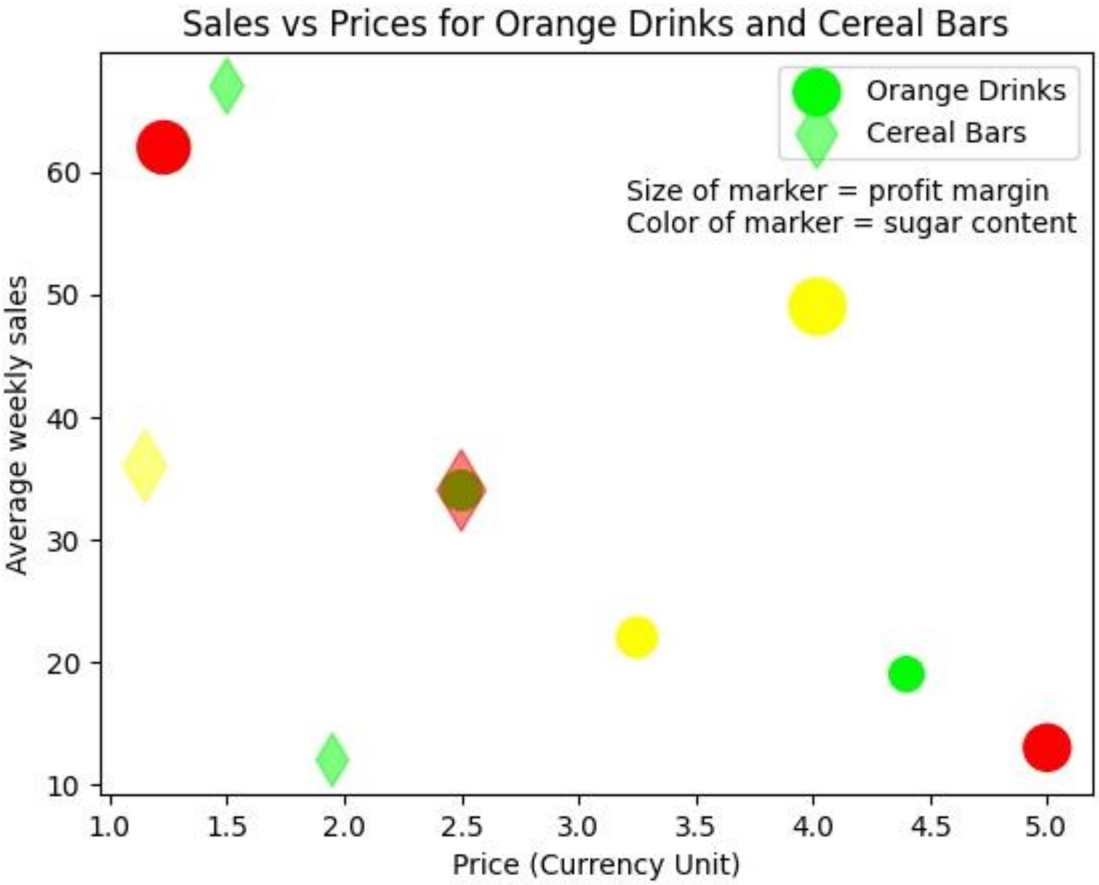
```
       Column_1  Column_2  Column_3
Row_1        15        22        43
Row_2        33        24        56
```

8. Draw a scatterplot from a dataset

```
import matplotlib.pyplot as plt
import numpy as np
```

```
 low = (0, 1, 0)
medium = (1, 1, 0)
high = (1, 0, 0)
price_orange = np.asarray([2.50, 1.23, 4.02, 3.25, 5.00, 4.40])
sales_per_day_orange = np.asarray([34, 62, 49, 22, 13, 19])

profit_margin_orange = np.asarray([20, 35, 40, 20, 27.5, 15])
sugar_content_orange = [low, high, medium, medium, high, low]

price_cereal = np.asarray([1.50, 2.50, 1.15, 1.95])
sales_per_day_cereal = np.asarray([67, 34, 36, 12])
profit_margin_cereal = np.asarray([20, 42.5, 33.3, 18])
sugar_content_cereal = [low, high, medium, low]

plt.scatter(
    x=price_orange,
    y=sales_per_day_orange,
    s=profit_margin_orange * 10,
    c=sugar_content_orange,
)
plt.scatter(
    x=price_cereal,
    y=sales_per_day_cereal,
    s=profit_margin_cereal * 10,
    c=sugar_content_cereal,
    marker="d",
    alpha=0.5,
 )
 plt.title("Sales vs Prices for Orange Drinks and Cereal Bars")
plt.legend(["Orange Drinks", "Cereal Bars"])
plt.xlabel("Price (Currency Unit)")
plt.ylabel("Average weekly sales")
plt.text(
    3.2,
    55,
    "Size of marker = profit margin\n" "Color of marker = sugar content",
)
 plt.show()
```

OUTPUT:



RESULT:

Thus, different applications are implemented by using Numpt, Pandas and MatplotLib Packages.

CONTENT BEYOND SYLLABUS

Ex. No: 12

Date:

SCIKIT-LEARN

AIM:

To perform program that performs classification using the Naive Bayes classifier from scikit-learn. This program uses the famous Iris dataset for training and evaluating the classifier:

PROGRAM:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data  # Features
y = iris.target  # Target variable

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Naive Bayes classifier
classifier = GaussianNB()

# Train the classifier
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)


# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

In this program, we start by loading the Iris dataset using the datasets.load_iris() function. Then, we split the dataset into training and testing sets using train_test_split(). We create a Naive Bayes classifier using GaussianNB() and train it on the training set with the fit() method. Next, we make predictions on the test set using predict(), and finally, we calculate the accuracy of the classifier by comparing the predicted labels (y_pred) with the true labels (y_test).

OUTPUT :

```
Accuracy: 0.966666666666666
```

RESULT:

Thus, the program that performs classification using the Naive Bayes classifier from scikit-learn. is executed and successfully verified.