PROGRAMMING FOR PROBLEM SOLVING LAB

LAB MANUAL

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.

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. 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: GE3171 Syllabus PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY

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, 1st Edition, 2021.

2. G Venkatesh and Madhavan Mukund, “Computational Thinking: A Primer for Programmers and Data Scientists”, 1st 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”, 2nd Edition, No Starch Press, 2019.

5. https://www.python.org/

6. Martin C. Brown, “Python: The Complete Reference”, 4th Edition, Mc-Graw Hill, 2018.

EXPT.NO.1 DATE:	**Identification and solving of simple real life or scientific or technical problems (Electricity Billing, Retail Shop billing ,Sin series etc)**

a) Electricity Billing

Aim:

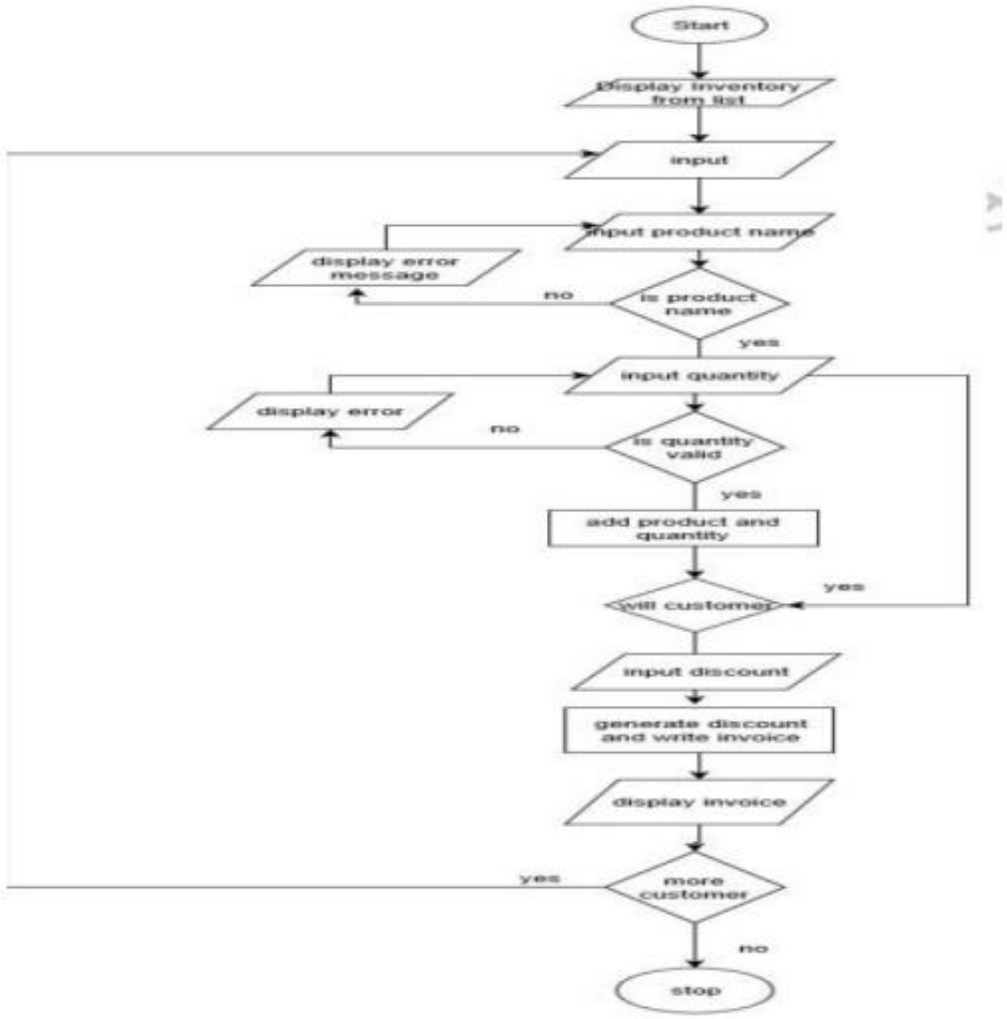
To Develop a flow chart and write the program for Electricity billing

Procedure:

From Unit To Unit Rate (Rs.) Prices

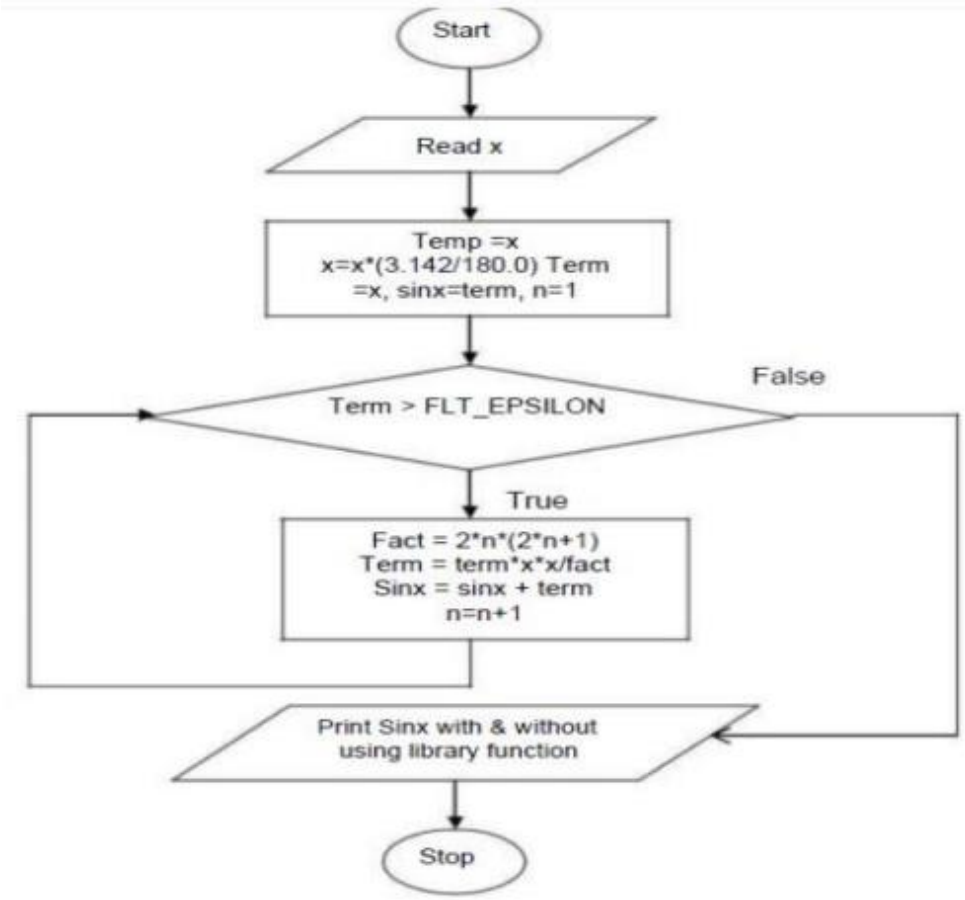
From Unit	To Unit	Rate (Rs.)	Max.Unit
1	100	0	100
101	200	2	200
201	500	3	500-
-	101 -200	3.5	>500
	201-500	4.6	>500
	>500	606	>500

1b) Reatil Shop billing

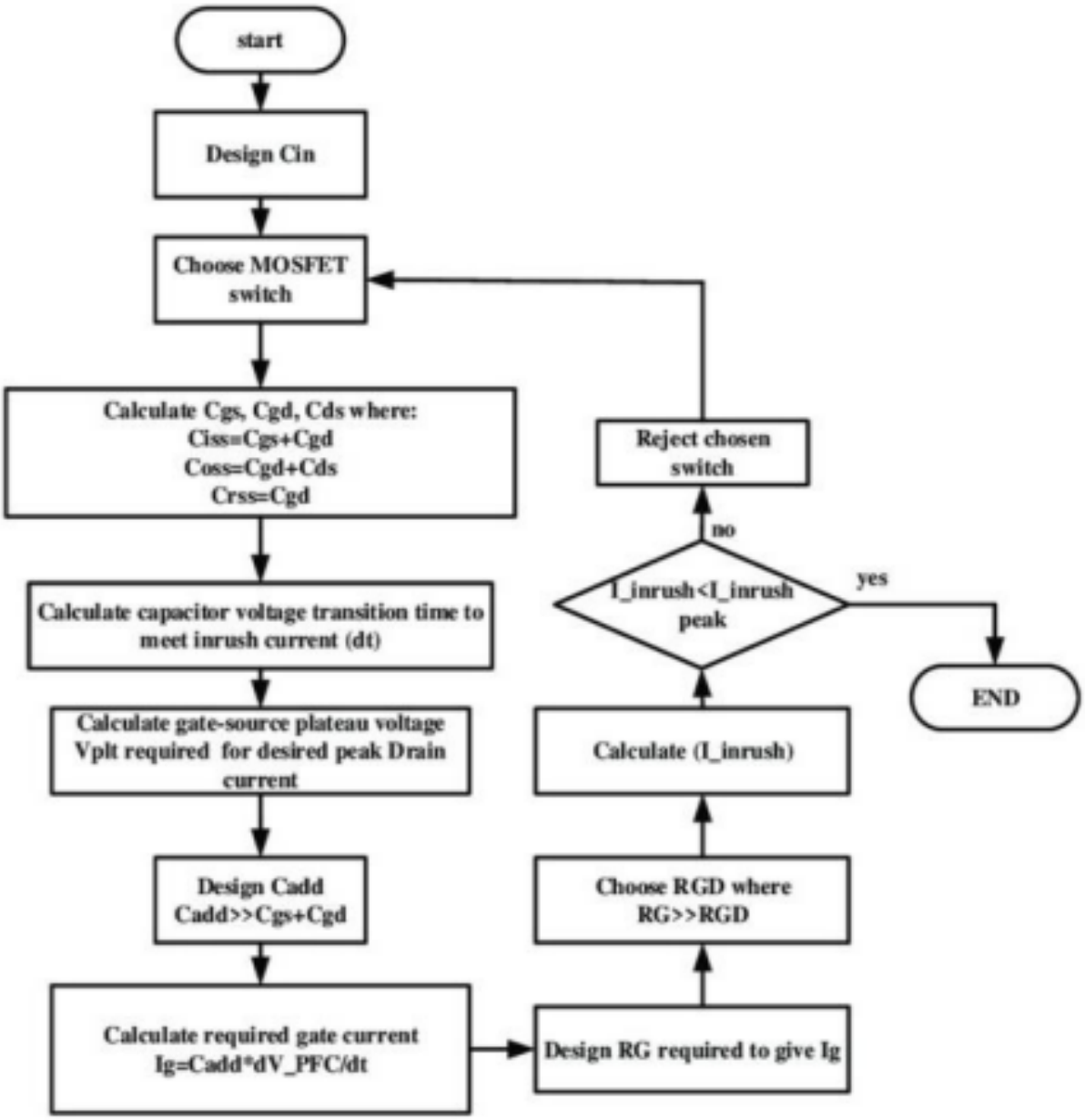


1c) Sin series

Flow Chart:



1d) To compute Electrical Current in Three Phase AC Circuit



Result :

Thus the flowchart of electric bill, sine series was successfully verified

EXPT.NO 2.a DATE:	Python Programming Using Simple Statements And Expressions - Exchange The Values Of Two Variables)

AIM:

Write a python program to exchange the values of two variables

PROCEDURE:

step 1:Declared a temporary variable a and b

step 2:Assign the value of a and b,

step 3:Assign the value of a to b, and b to a

step 4: we don't even need to perform arithmetic operations. We can use:

a,b = b,a

step 5:to print the result

PROGRAM:

```
a=10
b=20
a,b=b,a
print("The swapping of a value is=",a)
print("The swapping of b value is=",b)
```

OUTPUT:

```
The swapping of a value is= 20
The swapping of b value is= 10
```

RESULT:

Thus the swapping of two numbers python program was executed successfully and verified.

EXPT.NO 2b DATE:	**Python Programming Using Simple Statements and Expressions - Circulate The DATE: Values Of N Variables**

AIM:

Write a python program to circulate the values of n variables

PROCEDURE:

Step1: Circulate the values of n variables.

Step2: Get the input from the user

Step 3: To create the empty list then declare the conditional statements using for loop

Step 4: Using append operation to add the element in the list and the values are rotate by using this append operation

Step 5: Stop the program

PROGRAM:

```
n = int(input("Enter number of values : "))
list1 = []
for val in range(0,n,1):
    ele = int(input("Enter integer : "))
    list1.append(ele)
print("Circulating the elements of list ", list1)
for val in range(0,n,1):
     ele = list1.pop(0)
     list1.append(ele)
     print(list1)
```

OUTPUT:

```
Enter number of values : 4
Enter integer : 87
Enter integer : 58
Enter integer : 98
Enter integer : 52
Circulating the elements of list [87, 58, 98, 52]
[58, 98, 52, 87]
[98, 52, 87, 58]
[52, 87, 58, 98]
[87, 58, 98, 52]
```

RESULT:

Thus the python program to circulate the values of n variables was executed successfully and verified.

EXPT.NO 2c DATE:	Python Programming Using Simple Statements And Expressions (Calculate The Distance Between Two Points)

AIM:

Write a python program to calculate the distance between two numbers

PROCEDURE:

Step 1: Start the program.

Step 2: Read all the values of x1,x2,y1,y2.

Step 3: Calculate the result.

Step 4: Print the result.

Step 5: Stop the program

PROGRAM

```
import math
x1=int(input("enter the value of x1="))
x2=int(input("enter the value of x2="))
y1=int(input("enter the value of y1="))
y2=int(input("enter the value of y2="))
dx=x2-x1
dy=y2-y1
d=dx**2+dy**2
result=math.sqrt(d)
print(result)
```

OUTPUT:

```
enter the value of x1=34
enter the value of x2=56
enter the value of y1=34
enter the value of y2=54
29.732137494637012
```

RESULT:

Thus the distance between of two points was successfully executed and verified.

EXPT.NO 3(a) DATE:	**Scientific problems using Conditionals and Iterative loops.- Number series**

AIM:

Write a Python program with conditional and iterative statements for Number Series.

PROCEDURE:

Step 1: Start the program.

Step 2: Read the value of n.

Step 3: Initialize i = 1,x=0.

Step 4: Repeat the following until i is less than or equal to n.

Step 4.1: x=x*2+1.

Step 4.2: Print x.

Step 4.3: Increment the value of i .

Step 5: Stop the program.

PROGRAM:

```
n=int(input(" enter the number of terms for the series "))
i=1
x=0
while(i<=n):
    x=x*2+1
    print(x, sep= "")
    i+=1
```

OUTPUT:

```
enter the number of terms for the series 5
1
3
7
15
31
```

RESULT:

Thus the python program to print numbers patterns is executed and verified

EXPT.NO 3b DATE:	Scientific Problems Using Conditionals And Iterative Loops. –Number Patterns

AIM:

Write a Python program with conditional and iterative statements for Number Pattern.

PROCEDURE:

Step 1: Start the program

Step 2: Declare the value for rows.

Step 3: Let i and j be an integer number

Step 4: Repeat step 5 to 8 until all value parsed.

Step 5: Set i in outer loop using range function, i = rows+1 and rows will be initialized to i

Step 6: Set j in inner loop using range function and i integer will be initialized to j;

Step 7: Print i until the condition becomes false in inner loop.

Step 8: Print new line until the condition becomes false in outer loop.

Step 9: Stop the program.

PROGRAM

```
rows = int(input('Enter the number of rows'))
for i in range(rows+1):
        for j in range(i):
                print(i, end=' ')
        print(' ')
```

OUTPUT:

```
Enter the number of rows=7
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
```

RESULT:

Thus the python program to print numbers patterns is executed and verified.

EXPT.NO 3c DATE:	Scientific Problems Using Conditionals and Iterative Loops. -Pyramid

AIM:

Write a Python program with conditional and iterative statements for Pyramid Pattern.

PROCEDURE:

Step 1: Start the program

Step 2: Read the value for rows.

Step 3: Let i and j be an integer number.

Step 4: Repeat step 5 to 8 until all value parsed.

Step 5: Set i in outer loop using range function, i = 0 to rows ;

Step 6: Set j in inner loop using range function, j=0 to i+1;

Step 7: Print * until the condition becomes false in inner loop.

Step 8: Print new line until the condition becomes false in outer loop.

Step 9: Stop the program.

PROGRAM:

```
def pypart(n):
    for i in range(0, n):
        for j in range(0, i+1):
            print("*",end=" ")
        print("\r")
```

OUTPUT:

```
enter the no. of rows 5
*
* *
* * *
* * * *
* * * * *
```

RESULT:

Thus the python program to print numbers pyramid patterns is executed and verified.

EXPT.NO 4(a) DATE:	Implementing Real-Time/Technical Applications Using Lists, Tuples -Items Present In a Library)

AIM :

To Write a python program to implement items present in a library

PROCEDURE:

STEP 1: Start the program

STEP 2: Create the variable inside that variable assigned the list of elements based on the library using List and tuple

STEP 3:Using array index to print the items using list and tupel

STEP 4:To print the result using output statement

STEP 5: Stop the program

PROGRAM:

```
library=["books", "author", "bar code number" , "price"]
library[0]="ramayanam"
print(library[0])
library[1]="valmiki"
library[2]=123987
library[3]=234
print(library)
#Tuple:
tup1 = (12134,250000 )
tup2 = ('books', 'totalprice')
# tup1[0] = 100 -------□ Not assigned in tuple
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print(tup3)
#TUPLE:
tup1 = (12134,250000 )
tup2 = ('books', 'totalprice')
# tup1[0] = 100 ------- Not assigned in tuple # So let's create a new tuple as follows
tup3 = tup1 + tup2;
print(tup3)
```

OUTPUT:

ramayanam

['ramayanam', 'valmiki', 123987, 234]

TUPLE:

(12134, 250000, 'books', 'totalprice')

RESULT:

Thus the Python Program is executed successfully and the output is Verified.

EXPT.NO 4(b) DATE:	Implementing Real-Time/Technical Applications Using Lists, Tuples - Components DATE: Of a Car.

AIM:

To Write a python program to implement components of a car

PROCEDURE:

STEP 1: Start the program

STEP 2: Create the variable inside that variable assigned the list of elements based on the car using List and tuple

STEP 3:Using array index to print the items using list and tuple

STEP 4:To print the result using output statement

STEP 5: Stop the program

PROGRAM:

```
cars = ["Nissan", "Mercedes Benz", "Ferrari", "Maserati", "Jeep", "Maruti Suzuki"]

new_list = []

for i in cars:

  if "M" in i:

      new_list.append(i)

print(new_list)
```

#TUPLE:

```
cars=("Ferrari", "BMW", "Audi", "Jaguar")

print(cars)

print(cars[0])

print(cars[1])

print(cars[3])

print(cars[3])

print(cars[4])
```

OUTPUT:

LIST:

['Mercedes Benz', 'Maserati', 'Maruti Suzuki']

TUPLE:

('Ferrari', 'BMW', 'Audi', 'Jaguar')

Ferrari

BMW

Jaguar

Jaguar

RESULT:

Thus the Python Program is executed successfully and the output is verified.

EXPT.NO 4C DATE:	**Implementing Real-Time/Technical Applications Using Lists, Tuples - Materials Required For Construction Of A Building.**

AIM:

To Write a python program to implement materials required for construction of building

PROCEDURE:

STEP 1: Start the program

STEP 2: Create the variable and stored the unordered list of elements based on materials

Required for construction of building List and tuple

STEP 3: Using array index to print the items using list and tuple

STEP 4: To print the result using output statement

STEP 5: Stop the program.

PROGRAM:

```
materials= ["cementbags", "bricks", "sand", "Steelbars", "Paint"]
materials.append(“ Tiles”)
materials.insert(3,"Aggregates")
materials.remove("sand")
materials[5]="electrical"
print(materials)
#TUPLE:
materials = ("cementbags", "bricks", "sand", "Steelbars", "Paint")
print(materials)
print ("list of element is=",materials)
print ("materials[0]:", materials [0])
print ("materials[1:3]:", materials [1:3])
```

OUTPUT:

```
LIST:
['cementbags', 'bricks', 'Aggregates', 'Steelbars', 'Paint', 'electrical']
TUPLE:
('cementbags', 'bricks', 'sand', 'Steelbars', 'Paint')
list of element is= ('cementbags', 'bricks', 'sand', 'Steelbars', 'Paint')
materials[0]: cementbags
materials[1:3]: ('bricks', 'sand')
```

RESULT:

Thus the Python Program is executed successfully and the output is verified.

EXPT.NO : 5 DATE:	**Implementing Real-Time/Technical Applications Using Sets, Dictionaries. –Components Of An Automobile**

AIM:

To write a python program to implement Components of an automobile using Sets and Dictionaries

PROCEDURE:

STEP 1: Start the program

STEP 2: Create the variable and stored the unordered list of elements based on

materials required for construction of building set and dictionary

STEP 3: Using for loop to list the number of elements and using array index to

print the items using set and dictionary

STEP 4: To print the result using output statement

STEP 5: Stop the program

PROGRAM:

```
cars = {'BMW', 'Honda', 'Audi', 'Mercedes', 'Honda', 'Toyota', 'Ferrari', 'Tesla'}
print('Approach #1= ', cars)
print('==========')
print('Approach #2')
for car in cars:
        print('Car name = {}'.format(car))
print('==========')
cars.add('Tata')
print('New cars set = {}'.format(cars))
cars.discard('Mercedes')
print('discard() method = {}'.format(cars))
```

OUTPUT:

```
Approach #1= {'BMW', 'Mercedes', 'Toyota', 'Audi', 'Ferrari', 'Tesla', 'Honda'}
==========
Approach #2
Car name = BMW
Car name = Mercedes
Car name = Toyota
Car name = Audi
Car name = Ferrari
Car name = Tesla
Car name = Honda
```

==========

New cars set = {'BMW', 'Mercedes', 'Toyota', 'Audi', 'Ferrari', 'Tesla', 'Honda',

'Tata'}

discard() method = {'BMW', 'Toyota', 'Audi', 'Ferrari', 'Tesla', 'Honda', 'Tata'}

DICTIONARY

PROGRAM:

```
Dict = {}
print("Empty Dictionary: ")
print(Dict)
# Adding elements one at a time
Dict[0] = 'BRICKS'
Dict[2] = 'CEMENT'
Dict[3] = BLUE PRINT
print("\nDictionary after adding 3 elements: ")
print(Dict)
# Adding set of values
# to a single Key
Dict['Value_set'] = 2, 3, 4
print("\nDictionary after adding 3 elements: ")
print(Dict)
# Updating existing Key's Value
Dict[2] = 'STEEL'
print("\nUpdated key value: ")
print(Dict)
```

```
# Adding Nested Key value to Dictionary
Dict[5] = {'Nested': {'1': 'LIME', '2': 'SAND'}}
print("\nAdding a Nested Key: ")
print(Dict)
```

OUTPUT

```
Empty Dictionary:
{}
Dictionary after adding 3 elements:
{0: 'BRICKS', 2: 'CEMENT', 3: 'BLUE_PRINT'}
Dictionary after adding 3 elements: {0: 'BRICKS', 2: 'CEMENT', 3: 'BLUE_PRINT', 'Value_set': (2, 3, 4)}
Updated key value:
{0: 'BRICKS', 2: 'STEEL', 3: 'BLUE_PRINT', 'Value_set': (2, 3, 4)}
Adding a Nested Key:
{0: 'BRICKS', 2: 'STEEL', 3: 'BLUE_PRINT', 'Value_set': (2, 3, 4), 5: {'Nested': {'1': 'LIME', '2':
'SAND'}}}
```

RESULT:

Thus the program was executed successfully and verified.