B.Sc. I-Semester (Data Science)

List of Practical Programs for Lab Practice on Python Programming

I. Programs to demonstrate the usage of operators and conditional statements

1. Write a program that takes two integers as command line arguments and prints the sum of two integers.
2. Program to display the information: Your name, Full Address, Mobile Number, College Name, Course Subjects
3. Program to find the largest number among 'n' given numbers.
4. Program that reads the URL of a website as input and displays contents of a webpage.

II. Programs to demonstrate usage of control structures

5. Program to find the sum of all prime numbers between 1 and 1000.
6. Program that reads set of integers and displays 1^{st} and 2^{nd} largest numbers.
7. Program to print the sum of first 'n' natural numbers.
8. Program to find the product of two matrices.
9. Program to find the roots of a quadratic equation

III. Programs to demonstrate the usage of Functions and Recursion

10. Write both recursive and non-recursive functions for the following:
    a. To find GCD of two integers
    b. To find the factorial of positive integer
    c. To print Fibonacci Sequence up to given number 'n'
    d. To convert decimal number to Binary equivalent Syllabus Approved by BOS in Statistics w. e. f. 2020-21 Page 7 of 7

11. Program with a function that accepts two arguments: a list and a number ‘n’. It should display all the numbers in the list that are greater than the given number ‘n’.
12. Program with a function to find how many numbers are divisible by 2, 3,4,5,6 and 7 between 1 to 1000

IV. Programs to demonstrate the usage of String functions

13. Program that accepts a string as an argument and return the number of vowels and consonants the string contains.
14. Program that accepts two strings S1, S2, and finds whether they are equal are not.
15. Program to count the number of occurrences of characters in a given string.
16. Program to find whether a given string is palindrome or not

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

17. Program with a function that takes two lists L1 and L2 containing integer numbers as parameters. The return value is a single list containing the pair wise sums of the numbers in L1 and L2.
18. Program to read the lists of numbers as L1, print the lists in reverse order without using reverse function. 22. Write a program that combine lists L1 and L2 into a dictionary.
19. Program to find mean, median, mode for the given set of numbers in a list.
20. Program to find all duplicates in the list.
21. Program to o find all the unique elements of a list.
22. Program to find max and min of a given tuple of integers.
23. Program to find union, intersection, difference, symmetric difference of given two sets.
24. Program to display a list of all unique words in a text file

25. Program to read the content of a text file and display it on the screen line wise with a line number followed by a colon
26. Program to analyze the two text files using set operations
27. Write a program to print each line of a file in reverse order.

VI. Programs to demonstrate the usage of Object-Oriented Programming

28. Program to implement the inheritance
29. Program to implement the polymorphism VII. Programs to search and sort the numbers
30. Programs to implement Linear search and Binary search
31. Programs to implement Selection sort, Insertion sort.

PROGRAM – 1

I. Program to demonstrate the usage of operators and conditional statements

1. Python Program that takes two integers as command line arguments and prints the sum of two integers.

```
a = int(input(‘ 1st number is  ‘))
b= int(input(‘ 2nd number is  ‘))
s=a+b
print (f’sum of 2 integers is {s}’)
```

Output:

1st number is 6

2nd number is 7

Sum of 2 integers is 13

PROGRAM – 2

I. Program to demonstrate the usage of operators and conditional statements

2. Python Program to display the information: Your name, address, mobile number, college name, course, subjects.

```
def personal_details():
    name='Seetha'
    age='20'
    address='Hyderabad'
    mobile_number='1234567809'
    college_name='Osmania University College for Women, Koti'
    course='B.Sc.'
    print(f'Name is: {name}\n Age is: {age}\n Address is: {address} \n Mobile number is: {mobile_number} \n College name is: {college_name}\n Course is: {course}')
personal_details()
```

Output:

Name is: Seetha

Age is: 20

Address is: Hyderabad

Mobile number is: 1234567809

College name is: Osmania University College for Women, Koti

Course is: B.Sc.

PROGRAM – 3

I. Program to demonstrate the usage of operators and conditional statements

3. Python Program to find the largest number among 'n' given numbers.

```
max=0
while True:
    n= int(input('enter a number 0 to stop'))
    if n!=0:
        if n>max:
            max=n
            continue
    else:
        break
print(f'the maximum is{max}')
```

Output:

```
Enter a number 0 to stop 12
Enter a number 0 to stop 32
Enter a number 0 to stop 62
Enter a number 0 to stop 0
The maximum is 62
```

PROGRAM – 4

I. Program to demonstrate the usage of operators and conditional statements

4. Python Program that reads the URL of a website as input and displays contents of a webpage.

```
import requests
from bs4 import BeautifulSoup
def news():
    url=http://hundustantimes.com/top_news
    resp= requests.get(url)
    if resp.status_code==200:
        print('successfully opened the webpage')
        print('the news are as follows : \n')
        soup=BeautifulSoup(resp.text,'html.parser')
        l=soup.find('ul', {'class': 'searchNews'})
        for i in l.findAll('a'):
      print(i.text)
    else:
        print('error')
news()
```

Output:

Successfully opened the web page

The news are as follow :-

Govt extends toll tax suspension, use of old notes for utility bills extended till Nov 14

Modi, Abe seal historic civil nuclear pact: What it means for India

Rahul queues up at bank, says it is to show solidarity with common man

IS kills over 60 in Mosul, victims dressed in orange and marked 'traitors'

Rock On 2 review: Farhan Akhtar, Arjun Rampal's band hasn't lost its magic

Rumours of shortage in salt supply spark panic among consumers in UP

Worrying truth: India ranks first in pneumonia, diarrhoea deaths among kids

(Or)

```
import requests
from bs4 import BeautifulSoup
def news():
        url='http://www.hindustantimes.com/top-news'
        resp=requests.get(url)
        if resp.status_code==200:
                print("Successfully opened the web page")
                print("The news are as follow :-\n")
                soup=BeautifulSoup(resp.text,'html.parser')
                l=soup.find("ul",{"class":"searchNews"})
                for i in l.findAll("a"):
                        print(i.text)
        else:
                print("Error")
news()
```

Output:

Error

PROGRAM – 5

II. Programs to demonstrate usage of control structures

5. Program to find the sum of all prime numbers between 1 to 1000

```
max=int(input("Enter the maximum value for find sum of primes:"))
sum=0
for num in range(2,max+1):
   i=2
   for i in range(2,num):
      if(int(num%i==0)):
         i=num
         break;
      #when the number is prime calculate sum
   if i is not num:
      sum+=num
print("Sum of all prime numbers 1 to ",max,":",sum)
```

Output:

```
Enter the maximum value for find sum of primes:1000
Sum of all prime numbers 1 to  1000 : 76125
```

PROGRAM – 6

II. Programs to demonstrate usage of control structures

6. Program that reads set of integers and display 1st & 2nd largest numbers.

```
def calc_largest(arr):
    second_largest = arr[0]
    largest_val = arr[0]
    for i in range(len(arr)):
        if arr[i] > largest_val:
            largest_val = arr[i]
    print("the first largest value is:",largest_val)

    for i in range(len(arr)):
        if arr[i] > second_largest and arr[i] != largest_val:
            second_largest = arr[i]

    print("the second largest value is",second_largest)
print(calc_largest([20, 30, 40, 25, 10]))
```

Output:

```
the first largest value is: 40
the second largest value is 30
None
```

PROGRAM – 7

II. Programs to demonstrate usage of control structures

7. Program to print the sum of first n natural numbers.

```
s=0
c=1
n=int(input("Enter a number>0 :" ))
while c<=n:
    s+=c
    c+=1
print(f'the sum is {s}')
```

Output:

Enter any number>0 : 4

The sum is 10

PROGRAM – 8

II. Programs to demonstrate usage of control structures

8. Program to find the product of two matrices.

```
m=int(input("Enter number of rows for first matrix:"))
p=int(input("Enter number of columns for first matrix:"))
r=int(input("Enter number of columns for second matrix:"))
print('Enter elements of First matrix A :')
a=[[int(input()) for j in range(p)] for i in range(m)]
print('Enter elements of Second matrix B :')
b=[[int(input()) for j in range(r)] for i in range(p)]
c=[[0 for j in range(r)] for i in range(m)]
print("The First matrix A is :")
for i in range(m):
    for j in range(p):
        print(a[i][j])
for i in a:
    print(i)
print("The Second matrix B is :")
for i in range(p):
    for j in range(r):
        print(b[i][j])
for i in b:
    print(i)
print("The Product of two matrices A and B is :")
for i in range(m):
    for j in range(r):
        for k in range(len(c)):
          c[i][j]=c[i][j]+(a[i][k]*b[k][j])
          print(c[i][j])
for i in c:
    print(i)
```

Output:

```
Enter number of rows for first matrix:2
Enter number of columns for first matrix:2
Enter number of columns for second matrix:2
Enter elements of First matrix A :
1
2
3
4
Enter elements of Second matrix B :
5
6
7
8
```

The First matrix A is :
1
2
3
4
[1, 2]
[3, 4]
The Second matrix B is :
5
6
7
8
[5, 6]
[7, 8]
The Product of two matrices A and B is :
19
22
43
50
[19, 22]
[43, 50]

PROGRAM – 9

II. Programs to demonstrate usage of control structures

9. Program to find the roots of a quadratic equation.

```
import math
a=int(input("Enter the coefficient of X*X :"))
b=int(input("Enter the coefficient of X :" ))
c=int(input("Enter the constant of the equation :"))
d=b*b-4*a*c
if d<0:
    print("This equation has no real roots :")
elif d==0:
    x=(-b)/2*a
    print("The equation has unique real root :",x)
else:
    dd=math.sqrt(d)
    x1=(-b+dd)/2*a
    x2=(-b-dd)/2*a
    print("This equation has two real roots :" ,x1,x2)
```

Output:

```
Enter the coefficient of X*X : 1
Enter the coefficient of X : 4
Enter the constant of the equation :4
The equation has unique real root :  -2.0
```

PROGRAM – 10

II. Programs to demonstrate usage of control structures

10. a) Python Program for both recursive and non-recursive functions for finding GCD of two numbers.

```
def gcd (n1,n2):
    result=1
    k=2
    while k<=n1 and k<=n2:
        if n1%k==0 and n2 % k ==0:
            result=k
            k=k+1
        return result
def main():
    n1=int(input("enter first no:"))
    n2=int(input("enter second no:"))
    g=gcd(n1,n2)
    print(f'gcd is{g}')
if __name__=="__main__":
    main()
```

Output:

```
enter first no:34
enter second no:56
gcd is2
```

PROGRAM – 10

II. Programs to demonstrate usage of control structures

10. b) Python Program for both recursive and non-recursive functions to find the factorial of Positive integer.

```
number=int(input("Enter a number"))
factorial=1
if number<0:
   print("Factorial doesn't exist for negative number")
elif number==0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,number+1):
      factorial=factorial*i
print(f'The factorial of number {number} is {factorial}')
```

Output:

```
Enter a number 5
The factorial of number 5 is 120
```

PROGRAM – 10

II. Programs to demonstrate usage of control structures

10. c) Python Program for both recursive and non-recursive functions to Print Fibonacci Sequence up to given number 'n'

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < nterms:
      print(n1)
      nth = n1 + n2
      n1 = n2
      n2 = nth
      count += 1
```

Output:

How many terms? 5

Fibonacci sequence

0
1
1
2
3

PROGRAM – 10

II. Programs to demonstrate usage of control structures

10. d) Python Program for both recursive and non-recursive functions to Convert decimal number to its binary equivalent

```
def dec_to_bin(n):
    if n>=1:
        dec_to_bin(n//2)
        print(n%2, end='')
number=int(input("enter a decimal number:"))
dec_to_bin(number)
```

Output:

```
Enter a decimal number: 19
10011
```

PROGRAM – 11

II. Programs to demonstrate usage of control structures

1. Program with a function that accepts two arguments a list and a number 'n'. It should display all the numbers in the list that are greater than the given number 'n'.

```
def print_greater(l1,n):
    for num in l1 :
        if num>n:
            print(num)
print_greater([3,5,69,],3)
```

Output:

5

69

PROGRAM – 12

II. Programs to demonstrate usage of control structures

2. Program with a function to find how many numbers are divisible by 2, 3,4,5,6 and 7 between 1 to 1000

```
nl=[]
for x in range(1, 1000):
  if (x%2==0) and (x%3==0) and (x%4==0) and (x%6==0) and (x%7==0) and
(x%5==0):
    nl.append(str(x))
print ("numbers in between 1 to 1000 divisible by 2,3,4,5,6 and 7 are:",','.join(nl))
```

Output:

numbers in between 1 to 1000 divisible by 2,3,4,5,6 and 7 are: 420,840

PROGRAM – 13

IV. Program to demonstrate the usage of string functions

3. Program that accepts a string as an argument and return the number of vowels and consonants the string contains.

```
def main():
    user_string=input("Enter a string:")
    vowels=0
    consonants=0

    for each_character in user_string:
        if (each_character=='a' or each_character=='e' or each_character=='i' or
each_character=='o' or each_character =='u'):
            vowels+=1
        elif  'a' < each_character<'z':
            consonants+=1
    print(f'Total number of vowels user entered string is {vowels}')
    print(f'Total number of consonants in user entered string is {consonants}')
if __name__=='__main__':
    main()
```

Output:

Enter a string:welcome to faculty development program

Total number of vowels user entered string is 12

Total number of consonants in user entered string is 22

PROGRAM – 14

IV. Program to demonstrate the usage of string functions

4. Program that accepts two strings S1, S2 and finds whether they are equal are not

```
def check (S1, S2) :
    S1 = 'listen'
    S2 = 'silent'
        if (sorted (S1) == (sorted (S2)) :
            print ('The strings are equal')
        else:
            print ('The strings are not equal')
```

Output:

Case1:

S1=listen

S2=silent

The strings are equal

Case2:

S1=cat

S2=bat

The strings are not equal

PROGRAM – 15

IV. Program to demonstrate the usage of string functions

5. Program to count the number of occurrences of characters in a given string

```
string = input("please enter your own string:")
char = input("please enter your own character:")
count = 0
for I in range (len(string)) :
        if (string[I] == char):
            count = count+1
print(f'The total number of times {char} has occurred ={count}')
```

Output:

```
please enter your own string:datascience

please enter your own character:e
The total number of times e has occurred =2
```

PROGRAM – 16

IV. Program to demonstrate the usage of string functions

6. Program to find whether a given string is palindrome or not.

```
def main():
    user_string =input ("Enter a string:")
    if user_string ==user_string [::-1]:
        print (f'User entered string is palindrome')
    else:
        print (f'User entered is not a palindrome')
if __name__=='__main__':
    main()
```

Output:

Case 1:

Enter string: madam

User entered string is palindrome

Case 2 :

Enter string: hello

User entered string is not palindrome.

PROGRAM – 17

V. Programs to demonstrate the usage of lists, sets, dictionaries , tuples and files.

7. Program with a function that takes two lists L1, and L2 containing integer numbers as parameters. The return value is a single list containing the pair wise sum the numbers in L1 and L2

```
def sum(L1,L2):


    L3=[]
    k= len(L1)
    print(f'{k}')
    n=0
    while n<+k:
       L3.append(L1[n]+L2[n])
       n+=1
    return L3
L1=[1,2,3]
L2=[4,5,6]
L4=sum(L1,L2)
print(f'{L4}')
```

Output:

```
3
[5, 7, 9]
```

PROGRAM – 18

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

8. Program to read the lists of numbers as l1, print the lists in reverse order without using reverse function.

```
l1= []
n = int (input("Enter no of items"))
i=1
while i<=n:
    n1=int (input ('enter any no'))
    l1. append (n1)
    i+=1
len=len(l1)-1
j=0
while len>=j:
    print (f'{l1[len]}')
    len-=1
```

Output:

```
Enter no of items4

enter any no5

enter any no6

enter any no7

enter any no8
8
7
6
5
```

PROGRAM – 18

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

Program that combines lists L1 and L2 into a dictionary.

```
d1= dict()
l1 = [1,2,3,4,5]
l2=['c', 'a', 'n', 'd', 'y']
for i  in range (len(l1)):
    d1.update({l1[i]:l2[i]})
print(f'{d1}')
```

Output :

```
{1: 'c', 2: 'a', 3: 'n', 4: 'd', 5: 'y'}
```

PROGRAM – 19

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

9. Program to find Mean, Median and Mode for the given set of numbers in a list.

```
#mean calculation
pythonic_machine_ages = [19, 22, 34, 26, 32, 30, 24, 24]

def mean(dataset):
    return sum(dataset) / len(dataset)

print("mean value:",mean(pythonic_machine_ages))

#median calculation
pythonic_machines_heights = [181, 187, 196, 196, 198, 203, 207, 211, 215]
after_retirement = [181, 187, 196, 198, 203, 207, 211, 215]

def median(dataset):
    data = sorted(dataset)
    index = len(data) // 2

    # If the dataset is odd
    if len(dataset) % 2 != 0:
        return data[index]

    # If the dataset is even
    return (data[index - 1] + data[index]) / 2
print("median value:",median(pythonic_machines_heights))
print("median value of after_retirement list:",median(after_retirement))
points_per_game = [3, 15, 23, 42, 30, 10, 10, 12]
sponsorship = ['nike', 'adidas', 'nike', 'jordan',
               'jordan', 'rebook', 'under-armour', 'adidas']

#mode calculation
def mode(dataset):
    frequency = {}

    for value in dataset:
        frequency[value] = frequency.get(value, 0) + 1

    most_frequent = max(frequency.values())
```

```
    modes = [key for key, value in frequency.items()
                    if value == most_frequent]

    return modes
print("mode value of points_per_game:",mode(points_per_game))
print("mode value of sponsorship:",mode(sponsorship))
```

Output:

```
mean value: 26.375
median value: 198
median value of after_retirement list: 200.5
mode value of points_per_game: [10]
mode value of sponsorship: ['nike', 'adidas', 'jordan']
```

PROGRAM – 20

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

10. Program to find all duplicates in the list.

```
L1 = [1, 2, 3, 1, 2, 4, 5, 6, 7, 8, 7, 9, 10, 10]
i = len (L1)
j = 0
k = 1
while j<i:
    k=j+1
    while k<i:
        if L1 [j] == L1 [k]:
            print (f'{L1 [j]}, {L1, [k]}')
        k=k+1
    j=j+1
```

Output:

```
1, ([1, 2, 3, 1, 2, 4, 5, 6, 7, 8, 7, 9, 10, 10], [3])
2, ([1, 2, 3, 1, 2, 4, 5, 6, 7, 8, 7, 9, 10, 10], [4])
7, ([1, 2, 3, 1, 2, 4, 5, 6, 7, 8, 7, 9, 10, 10], [10])
10, ([1, 2, 3, 1, 2, 4, 5, 6, 7, 8, 7, 9, 10, 10], [13])
```

PROGRAM – 21

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

11. Program to find all unique elements of lists

```
l1= [1,2,3,1,4,2,5,6,7,8,7]
i=len(l1)
dup=True
j=0
k=1
ld= []
while j<i:
  k=j+1
  while k<i:
     if l1[j]==l1[k]:
        t=l1[j]
        ld. append(l1[j])
        ld. append(l1[k])
     k=k+1
  j=j+1
for item in l1:
   if item not in ld:
      print(f'{item}')
```

Output:

```
3
4
5
6
8
```

PROGRAM – 22

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

22. Program to find maximum and minimum K elements of a given tuple of integers.

```
def Findel(tup,K):
    result = []
    test_tup = list(tup)
    temp = sorted(tup)
    for i, val in enumerate(temp):
        if i < K or i >= len(temp) - K:
            result.append(val)
    result = tuple(result)
    # printing result
    print("Max and Min K elements : ",result)

tup = (13, 10, 23, 2, 5, 6, 12)
K = 2
print("The original tuple: ", tup)
Findel(tup,K)
```

Output:

```
The original tuple:  (13, 10, 23, 2, 5, 6, 12)
Max and Min K elements :  (2, 5, 13, 23)
```

PROGRAM – 23

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

23. Program to find the Union, Intersection, Difference, Symmetric difference of any two sets.

```
A = {0,2,4,6,8}
B = {1,2,3,4,5}
print ("Union:", A|B)
print ("Intersection:",A&B)
print ("Difference:", A-B)
print ("Symmetric Difference:", A^B)
```

Output:

```
Union: {0, 1, 2, 3, 4, 5, 6, 8}
Intersection: {2, 4}
Difference: {0, 8, 6}
Symmetric Difference: {0, 1, 3, 5, 6, 8}
```

PROGRAM – 24

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

24. Program to display a list of all unique words in a text file

Create data.txt file:
apple is a very big company. An apple a day keeps doctor away. A big fat cat came across the road beside doctor's office.
The doctor owns apple device.

Program:

```
text_file = open('data.txt', 'r')
text = text_file.read()
#cleaning
text = text.lower()
words = text.split()
words = [word.strip('.,!;()[]') for word in words]
words = [word.replace("'s", '') for word in words]
#finding unique
unique = []
for word in words:
    if word not in unique:
        unique.append(word)
#sort
unique.sort()
#print
print(unique)
```

Output:

['a', 'across', 'an', 'apple', 'away', 'beside', 'big', 'came', 'cat', 'company', 'day', 'device', 'doctor', 'fat', 'is', 'keeps', 'office', 'owns', 'road', 'the', 'very']

PROGRAM – 25

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

25. Program to read the content of a text file and display it on the screen line wise with a line number followed by a colon.

```
L=["welcome\n", "to\n", "Datascience\n","FDP\n"]
file1 = open('myfile.txt', 'w')
file1.writelines(L)
file1.close()
file1 = open('myfile.txt', 'r')
Lines = file1.readlines()
count = 0
for line in Lines:
    count += 1
    print("Line{}: {}".format(count, line.strip()))
```

Output:

```
Line1: welcome
Line2: to
Line3: Datascience
Line4: FDP
```

26. Program to analyze the two text files using set operations (comparing files line by line)

```
f1 = open("myfile.txt", 'r')
f2 = open("data.txt", 'r')
i = 0
for line1 in f1:
    i += 1
    for line2 in f2:
        if line1 == line2:
            print("Line ", i, ": IDENTICAL")
        else:
            print("Line ", i, ":")
            print("\tFile 1:", line1, end='')
            print("\tFile 2:", line2, end='')
        break
f1.close()
f2.close()
```

OUTPUT:

Line 1 :

File 1: welcome

File 2: Apple is a very big company. An apple a day keeps doctor away. A big fat cat came across the road beside doctor's office.

Line 2 :

File 1: to

File 2: The doctor owns apple device.

PROGRAM – 27

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

27. Program to print each line of a file in reverse order.

```
textfile = open("story.txt")
lines = textfile.readlines()
for line in reversed(lines):
    print(line)
```

create story.txt file:

spiderman into the spider verse is a good movie.

It is animated.

I watched at with my siblings.

Output:

I watched it with my siblings

It is animated.

Spiderman into the Spider Verse is a good movie.

PROGRAM – 28

VI. Program to demonstrate the usage of Object Oriented Programming.

28. Program to implement the inheritance.

```
class A:
   def __init__(self,x):
        self.x=x
   def show(self):
        print (f'{self.x}')
class C:
   def __init__(self,z):
       self.z=z
   def show(self):
      print(f'{self.z}')
class B(A,C):
    def __init__(self,x,z,y):
        A.x=x
        C.z=z
        self.y=y
    def output(self):
      print(f' invoked from a class A')
      A.show(self)
      print(f'invoked from a class C')
      C.show(self)
      print(f'x={self.x},y={self.y},z={self.z}')
def main():
    obj=B(4,6,10)
    obj.output()
if __name__=='__main__':
    main()
```

Output:

```
invoked from a class A
4
invoked from a class C
6
x=4,y=10,z=6.
```

PROGRAM – 28

VI. Program to demonstrate the usage of Object Oriented Programming.

29. Program to implement the Polymorphism.

```
class shape:
  def area(self):
    pass
class rect(shape):
  def __init__(self,w,h):
      self.w=w
      self.h=h
  def area(self):
       print(f'area of rectangle is {self.w*self.h}')
class square(shape):
     def __init__(self,s):
        self.s=s
     def area(self):
        print(f'area of a square is {self.s**2}')
obj1=rect(2,3)
obj1.area()
obj2=square(3)
obj2.area()
```

Output:

area of rectangle is 6

area of square is 9

PROGRAM – 30

VI. Program to demonstrate the usage of Object Oriented Programming.

30. Program to implement Linear Search

```
def def read_key_item():
    key_item = int(input("Enter the key item to search: "))
    return key_item
def linear_search(search_key):
    list_of_items = [10, 20, 30, 40, 50]
    found = False
    for item_index in range(len(list_of_items)):
        if list_of_items[item_index] == search_key:
            found = True
            break
    if found:
            print(f'{search_key} found at position {item_index+1}')
    else:
        print("Item not found in the list")
def main():
    key_in_list = read_key_item()
    linear_search(key_in_list)
if __name__ == "__main__":
    main()
```

Output

enter the key item to search:50

50 found at position 5

PROGRAM – 30

VI. Program to demonstrate the usage of Object Oriented Programming.

30 Program to implement Binary search program

```
L=[3,6,10,15,56,61,82,99,101]
low=0
high =len(L)-1
found = False
k=int(input('enter element to search'))
while low<=high:
    mid=((low+high)//2)
    print(f'{mid}')
    if k==L[mid]:
        print(f'element{k} exists at {mid+1} position')
        found=True
        break
    elif k<L[mid]:
        high=mid-1
    else:
        low=mid+1
        if found== False:
            print(f'key{k} does not exists in the list')
```

Output:

```
enter element to search 56
4
element56 exists at 5 position.
```

PROGRAM – 31

VI. Program to demonstrate the usage of Object Oriented Programming.

31. Program to implement selection sort, insertion sort

```
def selectionSort(array, size):
    for step in range(size):
        min_idx = step
        for i in range(step + 1, size):
            # to sort in descending order, change > to < in this line
            # select the minimum element in each loop
            if array[i] < array[min_idx]:
                min_idx = i
        # put min at the correct position
        (array[step], array[min_idx]) = (array[min_idx], array[step])
data = [-2, 45, 0, 11, -9]
size = len(data)
selectionSort(data, size)
print('Sorted Array in Ascending Order:')
print(data)
```

Output;

```
Sorted Array in Ascending Order:
[-9, -2, 0, 11, 45]
```

PROGRAM – 31

VI. Program to demonstrate the usage of Object Oriented Programming.

31 Program to implement Insertion sort in Python

```
def insertionSort(array):
    for step in range(1, len(array)):
        key = array[step]
        j = step - 1
        # Compare key with each element on the left of it until an element smaller than it is found
        # For descending order, change key<array[j] to key>array[j].
        while j >= 0 and key < array[j]:
            array[j + 1] = array[j]
            j = j - 1
            # Place key at after the element just smaller than it.
        array[j + 1] = key
data = [9, 5, 1, 4, 3]
insertionSort(data)
print('Sorted Array in Ascending Order:')
print(data)
```

Output:

```
Sorted Array in Ascending Order:
[1, 3, 4, 5, 9]
```