NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL

PYTHON

(CS – 506)

BACHELOR OF TECHNOLOGY

NIIST BHOPAL		**NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY** **DEPT NAME: Computer Science & Engineering**	**FORM NO**	**NIIST/A/10**
BRANCH	**CSE**	**LIST OF EXPERIMENT**	**REV. NO**	**0**
SEMESTER	**V**		**REV. DT**	**30/06/2011**
SUBJECT/CODE :- Python / CS 506				

S.NO.	LIST OF EXPERIMENT
1	To write a Python program to find GCD of two numbers
2	To write a Python Program to find the square root of a number by Newton's Method.
3	To write a Python program to find the exponentiation of a number
4	To write a Python Program to find the maximum from a list of numbers.
5	To write a Python Program to perform Linear Search
6	To write a Python Program to perform binary search.
7	To write a Python Program to perform selection sort.
8	To write a Python Program to perform insertion sort.
9	To write a Python Program to perform Merge sort.
10	To write a Python program to find first n prime numbers.
11	To write a Python program to multiply matrices.
12	To write a Python program for command line arguments.
12	To write a Python program to find the most frequent words in a text read from a file.
12	To write a Python program to simulate elliptical orbits in Pygame.
12	To write a Python program to bouncing ball in Pygame

EXPERIMENT NO. 1

AIM: - To write a Python program to find GCD of two numbers.

Algorithm: 1. Define a function named compute GCD()
2. Find the smallest among the two inputs x and y
3. Perform the following step till smaller+1
Check if ((x % i == 0) and (y % i == 0)), then assign GCD=i
4. Print the value of gcd

Program:
```
def computeGCD(x, y):
if x > y:
smaller = y else:
smaller = x for i in range(1, smaller+1):
if((x % i == 0) and (y % i == 0)):
gcd = i
 return gcd
num1 = 54
num2 = 24
 # take input from the user
# num1 = int(input("Enter first number: "))
 # num2 = int(input("Enter second number: "))
print("The GCD. of", num1,"and", num2,"is", computeGCD(num1, num2))
```

Sample Output: $python main.py ('The GCD. of', 54, 'and', 24, 'is', 6)

EXPERIMENT NO. 2

AIM: - To write a Python Program to find the square root of a number by Newton's Method.

Algorithm:
1. Define a function named newtonSqrt().
2. Initialize approx as 0.5*n and better as 0.5*(approx.+n/approx.)
3. Use a while loop with a condition better!=approx to perform the following,
 i. Set approx.=better
ii. Better=0.5*(approx.+n/approx.)
4. Print the value of approx.

Program:

```
def newtonSqrt(n):
approx = 0.5 * n
better = 0.5 * (approx + n/approx)
while better != approx:
approx = better
 better = 0.5 * (approx + n/approx)
return approx
 print('The square root is' ,newtonSqrt(100))
```

Sample Output: The square root is 10

EXPERIMENT NO. 3

AIM: - To write a Python program to find the exponentiation of a number.

Algorithm:
1. Define a function named power()
2. Read the values of base and exp
3. Use 'if' to check if exp is equal to 1 or not
i. if exp is equal to 1, then return base
 ii.if exp is not equal to 1, then return (base*power(base,exp-1))
4. Print the result.

Program:

```
def power(base,exp):
 if(exp==1):
return(base)
 if(exp!=1):
 return(base*power(base,exp-1))
base=int(input("Enter base: "))
exp=int(input("Enter exponential value: "))
print("Result:",power(base,exp))
```

Sample Output: Enter base: 7 Enter exponential value: 2 Result:49

EXPERIMENT NO. 4

AIM: - To write a Python Program to find the maximum from a list of numbers.

Algorithm:

1. Create an empty list named l
2. Read the value of n
3. Read the elements of the list until n
4. Assign l[0] as maxno
5. If l[i]>maxno then set maxno=l[i]
6. Increment i by 1
7. Repeat steps 5-6 until I<n
8. Print the value of maximum number

Program:

```
l=[]
n=int(input("enter the upper limit"))
for i in range(0,n):
a=int(input("enter the numbers"))
l.append(a)
maxno=l[0]
for i in range(0,len(l)):
if l[i]>maxno:
maxno=l[i]
print("The maximum number is %d"%maxno)
```

Sample Output: Enter the upper limit 3
Enter the numbers 6
Enter the numbers 9
Enter the numbers 90
The maximum number is 90

EXPERIMENT NO. 5

AIM: -To write a Python Program to perform Linear Search

Algorithm:

1. Read n elements into the list
2. Read the element to be searched
3. If alist[pos]==item, then print the position of the item
4. else increment the position and repeat step 3 until pos reaches the length of the list.

Program:

```
items = [5, 7, 10, 12, 15]
print("list of items is", items)
x = int(input("enter item to search:")
i = flag = 0
while i < len(items):
if items[i] == x:
flag = 1
break i = i + 1
if flag == 1:
print("item found at position:", i + 1)
else:
print("item not found")
```

Sample Output:

```
$python main.py
(list of items is: [5, 7, 10, 12, 15] )
enter item to search: 7
(item found at position:, 2)
```

EXPERIMENT NO. 6

AIM: - To write a Python Program to perform binary search

```
# Modifications needed for the older Python 2 are found in comments.
# Returns index of x in arr if present, else -1
def binary_search(arr, low, high, x):
# Check base case
if high >= low:
mid = (high + low) // 2
# If element is present at the middle itself
if arr[mid] == x:
return mid
# If element is smaller than mid, then it can only
# be present in left subarray
elif arr[mid] > x:
return binary_search(arr, low, mid - 1, x)
# Else the element can only be present in right subarray
else:
return binary_search(arr, mid + 1, high, x)
else:
# Element is not present in the array
return -1
# Test array
arr = [ 2, 3, 4, 10, 40 ]
x = 10
# Function call
result = binary_search(arr, 0, len(arr)-1, x)
if result != -1:
print("Element is present at index", str(result))
else:
print("Element is not present in array")
```

Output:

Element is present at index 3

EXPERIMENT NO. 7

AIM: - To write a Python Program to perform selection sort.

Algorithm:

1. Create a function named selection sort
2. Initialise pos=0
3. If alist[location]>alist[pos] then perform the following till i+1,
4. Set pos=location
5. Swap alist[i] and alist[pos]
6. Print the sorted list

Program:

```
def selectionSort(alist):
 for i in range(len(alist)-1,0,-1):
 pos=0 for location in range(1,i+1):
 if alist[location]>alist[pos]:
 pos= location
temp = alist[i]
 alist[i] = alist[pos]
 alist[pos] = temp
alist = [54,26,93,17,77,31,44,55,20]
selectionSort(alist)
print(alist)
```

Sample Output: $python main.py [17, 20, 26, 31, 44, 54, 55, 77, 93]

EXPERIMENT NO. 8

AIM: - To write a Python Program to perform insertion sort.

Algorithm:

1. Create a function named insertionsort
2. Initialise currentvalue=alist[index] and position=index
3. while position>0 and alist[position-1]>currentvalue, perform the following till len(alist)
4. alist[position]=alist[position-1]
5. position = position-1
6. alist[position]=currentvalue
7. Print the sorted list

Program:

```
def insertionSort(alist):
for index in range(1,len(alist)):
 currentvalue = alist[index]
 position = index
 while position>0 and alist[position-1]>currentvalue:
alist[position]=alist[position-1] position = position-1
alist[position]=currentvalue
alist = [54,26,93,17,77,31,44,55,20]
 insertionSort(alist)
 print(alist)
```

Sample Output:

$python main.py [20, 54, 54, 54, 54, 54, 93, 93, 93]

EXPERIMENT NO. – 9

To write a Python Program to perform Merge sort

```
#merge function
def merge(arr, l, m, r):
  n1 = m - l + 1
  n2 = r- m
  # create arrays
  L = [0] * (n1)
  R = [0] * (n2)
  # Copy data to arrays
  for i in range(0 , n1):
    L[i] = arr[l + i]
  for j in range(0 , n2):
    R[j] = arr[m + 1 + j]
  i = 0 # first half of array
  j = 0 # second half of array
  k = l # merges two halves
  while i < n1 and j < n2 :
    if L[i] <= R[j]:
      arr[k] = L[i]
      i += 1
    else:
      arr[k] = R[j]
      j += 1
    k += 1
  # copy the left out elements of left half
  while i < n1:
    arr[k] = L[i]
    i += 1
    k += 1
  # copy the left out elements of right half
  while j < n2:
    arr[k] = R[j]
    j += 1
    k += 1
# sort
def mergeSort(arr,l,r):
  if l < r:
    # getting the average
    m = (l+(r-1))/2
    # Sort
    mergeSort(arr, l, m)
    mergeSort(arr, m+1, r)
    merge(arr, l, m, r)
# main
arr = [2,5,3,8,6,5,4,7]
n = len(arr)
mergeSort(arr,0,n-1)
print ("Sorted array is")
for i in range(n):
  print (arr[i],end=" ")
```

EXPERIMENT NO. 10

AIM: - To write a Python program to find first n prime numbers.

Algorithm:

1. Read the value of n
2. for num in range(0,n + 1), perform the following
3. if num%i is 0 then break else print the value of num
4. Repeat step 3 for i in range(2,num).

Program:

```
n = int(input("Enter the upper limit: "))
 print("Prime numbers are")
 for num in range(0,n + 1):
 # prime numbers are greater than 1
if num > 1:
for i in range(2,num):
if (num % i) == 0:
break
else:
print(num)
```

Sample Output: $python main.py

```
Enter the upper limit: 20
 Prime numbers are 2 3 5 7 11 13 17 19
```

EXPERIMENT NO. 11

AIM: - To write a Python program to multiply matrices.

Algorithm: step1: input two matrix.
Step 2: nested for loops to iterate through each row and each column.
Step 3: take one resultant matrix which is initially contains all 0. Then we multiply each row elements of first matrix with each elements of second matrix, and then add all multiplied value. That is the value of resultant matrix.

```
Program # Program to multiply two matrices
A=[]
n=int(input("Enter N for N x N matrix: "))
print("Enter the element ::>")
for i in range(n):
  row=[]                          #temporary list to store the row
  for j in range(n):
    row.append(int(input()))      #add the input to row list
    A.append(row)                 #add the row to the list
print(A)
# [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#Display the 2D array
print("Display Array In Matrix Form")
for i in range(n):
  for j in range(n):
    print(A[i][j], end=" ")
  print()                         #new line
B=[]
n=int(input("Enter N for N x N matrix : "))       #3 here
#use list for storing 2D array
#get the user input and store it in list (here IN : 1 to 9)
print("Enter the element ::>")
for i in range (n):
  row=[]                          #temporary list to store the row
  for j in range(n):
    row.append(int(input()))      #add the input to row list
    B.append(row)                 #add the row to the list
print(B)
# [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#Display the 2D array
print("Display Array In Matrix Form")
for i in range(n):
  for j in range(n):
    print(B[i][j], end=" ")
  print()
result = [[0,0,0], [0,0,0], [0,0,0]]
for i in range(len(A)):
  for j in range(len(B[0])):
    for k in range(len(B)):
      result[i][j] += A[i][k] * B[k][j]
print("The Resultant Matrix Is ::>")
for r in result:
  print(r)
```

Output

```
Enter N for N x N matrix: 3
Enter the element ::>
2
1
4
2
1
2
3
4
3
[[2, 1, 4], [2, 1, 2], [3, 4, 3]]
Display Array In Matrix Form
2 1 4
2 1 2
3 4 3
Enter N for N x N matrix : 3
Enter the element ::>
1
2
3
4
5
6
7
8
9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Display Array In Matrix Form
1 2 3
4 5 6
7 8 9
The Resultant Matrix Is ::>
[34, 41, 48]
[20, 25, 30]
[40, 50, 60]
```

EXPERIMENT NO. : 12

AIM : To write a Python program for command line arguments.

```
#!/usr/bin/python

import sys, getopt

def main(argv):
   inputfile = ''
   outputfile = ''
   try:
      opts, args = getopt.getopt(argv,"hi:o:",["ifile=","ofile="])
   except getopt.GetoptError:
      print 'test.py -i <inputfile> -o <outputfile>'
      sys.exit(2)
   for opt, arg in opts:
      if opt == '-h':
         print 'test.py -i <inputfile> -o <outputfile>'
         sys.exit()
      elif opt in ("-i", "--ifile"):
         inputfile = arg
      elif opt in ("-o", "--ofile"):
         outputfile = arg
   print 'Input file is "', inputfile
   print 'Output file is "', outputfile

if __name__ == "__main__":
   main(sys.argv[1:])
```

EXPERIMENT NO. 13

AIM: - To write a Python program to find the most frequent words in a text read from a file.

Algorithm:

1. Variable maxCount will store the count of most repeated word.
2. Open a file in read mode using file pointer.
3. Read a line from file. Convert each line into lowercase and remove the punctuation marks.
4. Split the line into words and store it in an array.
5. Use two loops to iterate through the array. Outer loop will select a word which needs to be count. Inner loop will match the selected word with rest of the array. If match found, increment count by 1.
6. If count is greater than maxCount then, store value of count in maxCount and corresponding word in variable word.
7. At the end, maxCount will hold the maximum count and variable word will hold most repeated word.

Program

```
count = 0;
word = "";
maxCount = 0;
words = [];

#Opens a file in read mode
file = open("data.txt", "r")

#Gets each line till end of file is reached
for line in file:
   #Splits each line into words
   string = line.lower().replace(',',").replace('.',").split(" ");
   #Adding all words generated in previous step into words
   for s in string:
      words.append(s);

#Determine the most repeated word in a file
for i in range(0, len(words)):
   count = 1;
   #Count each word in the file and store it in variable count
   for j in range(i+1, len(words)):
      if(words[i] == words[j]):
         count = count + 1;

   #If maxCount is less than count then store value of count in maxCount
   #and corresponding word to variable word
   if(count > maxCount):
      maxCount = count;
      word = words[i];

print("Most repeated word: " + word);
file.close();
```

Output:

```
Most repeated word: computer
```

EXPERIMENT NO. 14

Aim : To write a Python program to simulate elliptical orbits in Pygame.

```
import pygame
import math
import sys
pygame.init()
#setting screen size
screen = pygame.display.set_mode((600, 300))
#setting caption
pygame.display.set_caption("Elliptical orbit")
#creating clock variable
clock=pygame.time.Clock()
while(True):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
# setting x and y radius of ellipse
    xRadius = 250
    yRadius = 100
    for degree in range(0,360,10):
        x1 = int(math.cos(degree * 2 * math.pi/360) * xRadius)+300
        y1 = int(math.sin(degree * 2 * math.pi/360) * yRadius)+150
        screen.fill((0, 0, 0))
        pygame.draw.circle(screen, (255, 69, 0), [300, 150], 40)
        pygame.draw.ellipse(screen,(255,255,255),[50,50,500,200],1)
        pygame.draw.circle(screen, (0, 255, 0), [x1, y1], 20)
        pygame.display.flip()
        clock.tick(5)# screen refresh rate
```

EXPERIMENT NO.: 15

AIM : To write a Python program to bouncing ball in Pygame.

```
import os
os.environ['PYGAME_HIDE_SUPPORT_PROMPT'] = "hide"
import sys, pygame
from pygame.locals import *
pygame.init()

speed = [1, 1]
color = (255, 250, 250)
width = 550
height = 300

screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Pygame bouncing ball")

ball = pygame.image.load("ball.png")
rect_boundry = ball.get_rect()

while 1:
    for event in pygame.event.get():
        rect_boundry = rect_boundry.move(speed)
        if rect_boundry.left < 0 or rect_boundry.right > width:
            speed[0] = -speed[0]
        if rect_boundry.top < 0 or rect_boundry.bottom > height:
            speed[1] = -speed[1]

        screen.fill(color)
        screen.blit(ball, rect_boundry)
        pygame.display.flip()

        if event.type == QUIT:
            pygame.quit()
            sys.exit()
```