Raj Kumar Goel Institute of Technology, Ghaziabad

LABORATORY MANUAL

Faculty Name	:	Ms. Harshita bhardwaj	**Department**	:	CSE
Course Name	:	Python Language Programming Lab	**Course Code**	:	KCS-453
Year/Sem	:	2nd/4th	**NBA Code**	:	C-218
Email ID	:	hrshfcs@rkgit.edu.in	**Academic Year**	:	2021-22

Department of Computer Science and Engineering

VISION OF THE INSTITUTE

To continually develop excellent professionals capable of providing sustainable solutions to challenging problems in their fields and prove responsible global citizens.

MISSION OF THE INSTITUTE

We wish to serve the nation by becoming a reputed deemed university for providing value based professional education.

VISION OF THE DEPARTMENT

To be recognized globally for delivering high quality education in the ever changing field of computer science & engineering, both of value & relevance to the communities we serve.

MISSION OF THE DEPARTMENT

1. To provide quality education in both the theoretical and applied foundations of Computer Science and train students to effectively apply this education to solve real world problems.
2. To amplify their potential for lifelong high quality careers and give them a competitive advantage in the challenging global work environment.

PROGRAM EDUCATIONAL OUTCOMES (PEOs)

PEO 1: Learning: Our graduates to be competent with sound knowledge in field of Computer Science & Engineering.

PEO 2: Employable: To develop the ability among students to synthesize data and technical concepts for application to software product design for successful careers that meet the needs of Indian and multinational companies.

PEO 3: Innovative: To develop research oriented analytical ability among students to prepare them for making technical contribution to the society.

PEO 4: Entrepreneur / Contribution: To develop excellent leadership quality among students which they can use at different levels according to their experience and contribute for progress and development in the society.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, a n d modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: The ability to use standard practices and suitable programming environment to develop software solutions.

PSO2: The ability to employ latest computer languages and platforms in creating innovative career opportunities.

COURSE OUTCOMES (COs)

C218.1	Design various functions in Python and demonstrate the basic concepts of Python.
C218.2	Examine searching and sorting in python using function.
C218.3	Examine and demonstrate the concepts of object oriented programming language in Python.
C218.4	Design the form using python program simulate in Pygame.

CO-PO MAPPING

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C218.1	3	1										2
C218.2	2	2	2	1	2							3
C218.3	2	1	2		2							3
C218.4	2	1	2		2							3
C218	2.25	1.25	2	1	2							2.75

CO-PSO MAPPING

	PSO1	PSO2
C218.1	3	3
C218.2	3	3
C218.3	3	3
C218.4	3	3
C218	3	3

LIST OF EXPERIMENTS

Expt. No.	Title of experiment	Corresponding CO
1.	To write a python program that takes in command line arguments as input and print the number of arguments.	C 218.1
2.	To write a python program to perform Matrix Multiplication.	C 218.1
3.	To write a python program to compute the GCD of two numbers.	C 218.1
4.	To write a python program to find the most frequent words in a text file.	C 218.1
5.	To write a python program find the square root of a number (Newton's method).	C 218.1
6.	To write a python program exponentiation (power of a number).	C 218.1
7.	To write a python program find the maximum of a list of numbers.	C 218.1
8.	To write a python program linear search.	C 218.2
9.	To write a python program Binary search.	C 218.2
10.	To write a python program selection sort.	C 218.2
11.	To write a python program Insertion sort.	C 218.2
12.	To write a python program merge sort.	C 218.2
13.	To write a python program first n prime numbers.	C 218.1
14.	. To write a python program simulate bouncing ball in Pygame.	C 218.4

Content Beyond Syllabus		
15.	Write a Python program to compute area and circumference of a triangle. Take input from user.	C 218.1
16.	Write a program to check if a number is Odd or even. Take input from user.	C 218.2
17.	Write a program to check that a given year is Leap Year or not.	C 218.3
18.	Write a Python program to print ‘n terms of Fibonacci series using iteration	C 218.3

INTRODUCTION

Python is a language with a simple syntax, and a powerful set of libraries. It is an interpreted language, with a rich programming environment, including a robust debugger and profiler. While it is easy for beginners to learn, it is widely used in many scientific areas for data exploration. This course is an introduction to the Python programming language for students without prior programming experience. We cover data types, control flow, object-oriented programming, and graphical user interface-driven applications. The examples and problems used in this course are drawn from diverse areas such as text processing, simple graphics creation and image manipulation, HTML and web programming, and genomics.

Scope & Objective:

1. Learn basic programming constructs –data types, decision structures, control structures in python
2. Know how to use libraries for string manipulation and user-defined functions.
3. Learn to use in-built data structures in python – Lists, Tuples, Dictionary and File handling.
4. Learn the fundamental principles of Object-Oriented Programming
5. Solve problems through application of OO concepts and using Files/database

Use Python Shell (using command line) and IDLE – Interactive development environment.

- To evaluate expression.
- To create a script.

Using IDLE

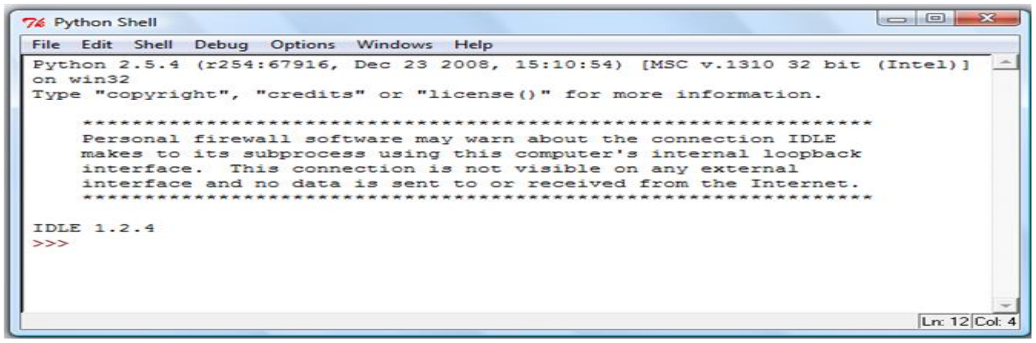
IDLE is the standard Python development environment. Its name is an acronym of "**I**ntegrated **D**eve**L**opment **E**nvironment". It works well on both Unix and Windows platforms.

It has a Python shell window, which gives you access to the Python interactive mode. It also has a file editor that lets you create and edit existing Python source files.

During the following discussion of IDLE's features, instead of passively reading along, you should start IDLE and try to replicate the screenshots.

Interactive Python shell

When you start up IDLE, a window with an interactive Python shell will pop up:



```
Python 2.5.4 (r254:67916, Dec 23 2008, 15:10:54) [MSC v.1310 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 1.2.4
>>>
```

You can type Python code directly into this shell, at the '>>>' prompt. Whenever you enter a complete code fragment, it will be executed. For instance, typing:

```
>>> print "hello world"
```

and pressing ENTER, will cause the following to be displayed:

```
hello world
```

Try typing an underscore (_). Can you see it? On some operating systems, the bottoms of hanging letters such as 'g' or 'y', as well as underscores, cannot be seen in IDLE. If this is the case for you, go to Options -> Configure IDLE, and change the size of the default font to 9 or 11. This will fix the problem!

IDLE can also be used as a calculator:

```
>>> 4+4
8
>>> 8**3
512
```

Addition (+), subtraction (-), multiplication (*), division (/), modulo (%) and power (**) operators are built into the Python language. This means you can use them right away. If you want to use a square root in your calculation, you can either raise something to the power of 0.5 or you can import the math module
Below are two examples of square root calculation:

```
>>> 16**0.5
4.0
>>> import math
>>> math.sqrt(16)
4.0
```

The math module allows you to do a number of useful operations:

```
>>> math.log(16, 2)
4.0
>>> math.cos( 0 )
1.0
```

Note that you only need to execute the import command once after you start IDLE; however you will need to execute it again if you restart the shell, as restarting resets everything back to how it was when you opened IDLE.

Creating scripts

1. save your hello.py program in the ~/pythonpractice folder.
2. Open up the terminal program. ...
3. Type cd ~/pythonpractice to change directory to your pythonpractice folder, and hit Enter.
4. Type chmod a+x hello.py to tell Linux that it is an executable program.
5. Type ./hello.py to run your program!

Program to add two integers.

Take input from user.

```
number1 = input(" Please Enter the First Number: ")
number2 = input(" Please Enter the second number:  ")

# Using arithmetic + Operator to add  two numbers
sum = float(number1) + float(number2)
print('The sum of {0} and {1} is {2}'.format(number1, number2,  sum))
```

Program to calculate area of a triangle:

(a) Given Base and height of the triangle. Take input from user.

(b) Given Three sides of a triangle (Make sure that it forms a triangle). Take input from user.

```
# Python Program to find Area of a Triangle

a = float(input('Please Enter the First side of a Triangle: '))
b = float(input('Please Enter the Second side of a Triangle: '))
c = float(input('Please Enter the Third side of a Triangle: '))

# calculate the Perimeter
Perimeter = a + b + c

# calculate the semi-perimeter
s = (a + b + c) / 2

# calculate the area
Area = (s*(s-a)*(s-b)*(s-c)) ** 0.5

print("\n The Perimeter of Traiangle = %.2f" %Perimeter);
print(" The Semi Perimeter of Traiangle = %.2f" %s);
print(" The Area of a Triangle is %0.2f" %Area)
```

PREFACE

Python is a powerful high-level, object-oriented programming language created by Guido van Rossum.

It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

Ms. Harshita Bhardwaj,
Assistant Professor, Dept. of CSE

DO'S AND DONT'S

DO's

1. Conform to the academic discipline of the department.
2. Enter your credentials in the laboratory attendance register.
3. Read and understand how to carry out an activity thoroughly before coming to the laboratory.
4. Ensure the uniqueness with respect to the methodology adopted for carrying out the experiments.
5. Shut down the machine once you are done using it.

DONT'S

1. Eatables are not allowed in the laboratory.
2. Usage of mobile phones is strictly prohibited.
3. Do not open the system unit casing.
4. Do not remove anything from the computer laboratory without permission.
5. Do not touch, connect or disconnect any plug or cable without your faculty/laboratory technician's permission.

GENERAL SAFETY INSTRUCTIONS

1. Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.

2. Report fire or accidents to your faculty /laboratory technician immediately.

3. Report any broken plugs or exposed electrical wires to your faculty/laboratory technician immediately.

4. Do not plug in external devices without scanning them for computer viruses.

GUIDELINES FOR LABORTORY RECORD PREPARATION

While preparing the lab records, the student is required to adhere to the following guidelines:

Contents to be included in Lab Records:

1. Cover page
2. Vision
3. Mission
4. PEOs
5. POs
6. PSOs
7. COs
8. CO-PO-PSO mapping
9. Index
10. Experiments
    - Aim
    - Source code
    - Input-Output

A separate copy needs to be maintained for pre-lab written work.

The student is required to make the Lab File as per the format given on the next two pages.

Raj Kumar Goel Institute of Technology, Ghaziabad

PYTHON LANGUAGE PROGRAMMING LAB FILE (RCS 453)

Name	
Roll No.	
Section- Batch	

INDEX

Experiment No.	Experiment Name	Date of Conduction	Date of Submission	Faculty Signature

GUIDELINES FOR ASSESSMENT

Students are provided with the details of the experiment (Aim, pre-experimental questions, procedure etc.) to be conducted in next lab and are expected to come prepared for each lab class.

Faculty ensures that students have completed the required pre-experiment questions and they complete the in-lab programming assignment(s) before the end of class. Given that the lab programs are meant to be formative in nature, students can ask faculty for help before and during the lab class.

Students' performance will be assessed in each lab based on the following Lab Assessment Components:

Assessment Criteria-1: Performance (Max. marks = 5)

Assessment Criteria-2: VIVA (Max. marks = 5)

Assessment Criteria-3: Record (Max. marks = 5)

In each lab class, students will be awarded marks out of 5 under each component head, making it total out of 15 marks.

EXPERIMENT 1

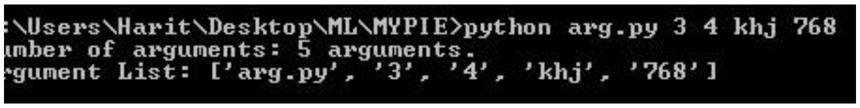
OBJECTIVE:- To write a python program that takes in command line arguments as input and print the number of arguments**.**

THEORY:- Python Command line arguments are input parameters passed to the script when executing them. Almost all programming language provides support for command line arguments.

PROGRAM:

```
import sys
print('Number of arguments:', len(sys.argv), 'arguments.')
print('Argument List:', sys.argv)
```

OUTPUT:



```
:\Users\Harit\Desktop\ML\MYPIE>python arg.py 3 4 khj 768
umber of arguments: 5 arguments.
rgument List: ['arg.py', '3', '4', 'khj', '768']
```

EXPERIMENT -2

OBJECTIVE:-To write a python program to perform Matrix Multiplication.

THEORY:- Matrix multiplication is an operation that takes two matrices as input and produces single matrix by multiplying rows of the first matrix to the column of the second matrix.In matrix multiplication make sure that the number of rows of the first matrix should be equal to the number of columns of the second matrix.

Example: Multiplication of two matrices by each other of size 3×3.

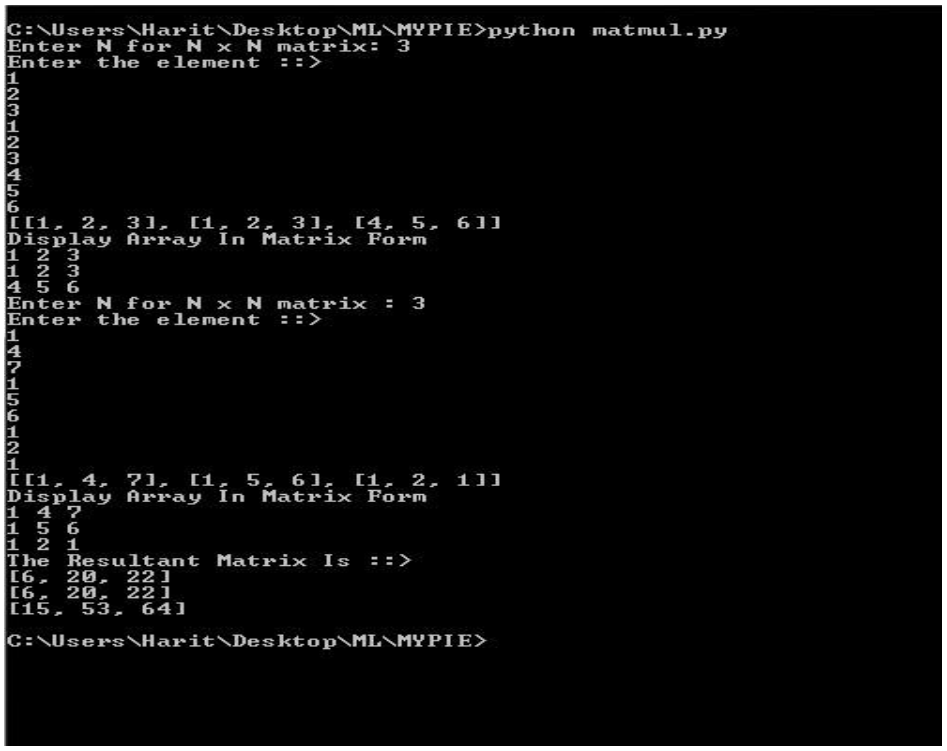
```
Input:matrix1 = ([1, 2, 3],
                 [3, 4, 5],
                 [7, 6, 4])
      matrix2 = ([5, 2, 6],
                 [5, 6, 7],
                 [7, 6, 4])


Output : [[36 32 32]
          [70 60 66]
          [93 74 100]]
```

PROGRAM:

```
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:



```
C:\Users\Harit\Desktop\ML\MYPIE>python matmul.py
Enter N for N x N matrix: 3
Enter the element ::>
1
2
3
1
2
3
4
5
6
[[1, 2, 3], [1, 2, 3], [4, 5, 6]]
Display Array In Matrix Form
1 2 3
1 2 3
4 5 6
Enter N for N x N matrix : 3
Enter the element ::>
1
4
7
1
5
6
1
2
1
[[1, 4, 7], [1, 5, 6], [1, 2, 1]]
Display Array In Matrix Form
1 4 7
1 5 6
1 2 1
The Resultant Matrix Is ::>
[6, 20, 22]
[6, 20, 22]
[15, 53, 64]

C:\Users\Harit\Desktop\ML\MYPIE>
```

EXPERIMENT-3

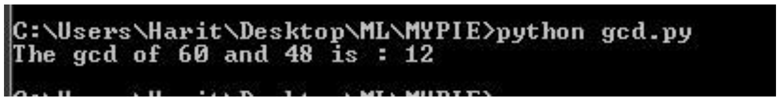
OBJECTIVE:- To write a python program to compute the GCD of two numbers.

THEORY: The highest common factor (H.C.F) or greatest common divisor (G.C.D) of two numbers is the largest positive integer that perfectly divides the two given numbers. For example, the H.C.F of 12 and 14 is 2.

PROGRAM:

```
def compGCD(x, y):
    if x > y:
        small = y
    else:
        small = x
    for i in range(1, small+1):
        if((x % i == 0) and (y % i == 0)):
            gcd = i
    return gcd
a = 60
b= 48
print ("The gcd of 60 and 48 is : ", end=" ")
print (compGCD(60,48))"""
import math
# prints 12
print ("The gcd of 60 and 48 is : ",end="")
print (math.gcd(60,48))
```

OUTPUT:



```
C:\Users\Harit\Desktop\ML\MYPIE>python gcd.py
The gcd of 60 and 48 is : 12
```

VIVA QUESTIONS

1. What is the difference between iteration and recursion?

2. How looping technique is different from Recursion?

EXPERIMENT-4

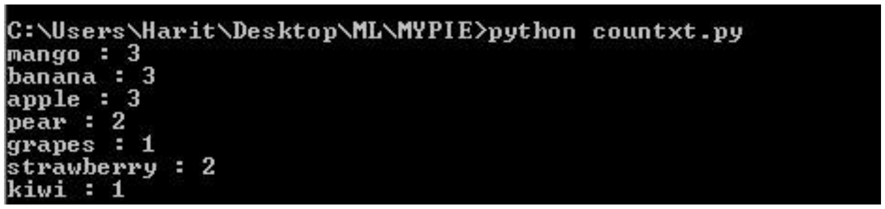
OBJECTIVE:**-** To write a python program to find the most frequent words in a text file.

PROGRAM:

```
import string
# Open the file in read mode
text = open("fruits.txt",  "r")
# Create an empty dictionary
d = dict()
# Loop through each line of the file
for line in text:
    # Remove the leading spaces and newline character
    line = line.strip()
    # Convert the characters in line to
    # lowercase to avoid case mismatch
    line = line.lower()
    # Remove the punctuation marks from the line
    line = line.translate(line.maketrans("", "", string.punctuation))
    # Split the line into words
    words = line.split(" ")
    # Iterate over each word in line
    for word in words:
        # Check if the word is already in dictionary
        if word in d:
            # Increment count of word by 1
            d[word] = d[word] + 1
        else:
```

```
# Add the word to dictionary with count 1
            d[word] = 1
# Print the contents of dictionary
for key in list(d.keys()):
    print(key, ":",  d[key])
```

OUTPUT:



```
C:\Users\Harit\Desktop\ML\MYPIE>python countxt.py
mango : 3
banana : 3
apple : 3
pear : 2
grapes : 1
strawberry : 2
kiwi : 1
```

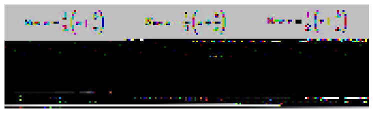
VIVA QUESTIONS

1. Are Strings immutable? Explain.
2. What are the ways to create a string?

EXPERIMENT-5

OBJECTIVE: To write a python program find the square root of a number (Newton's method).

THEORY: The square root of a positive number **b** can be computed with Newton's formula:



where **x** above starts with a "reasonable" guess. In fact, you can always start with **b** or some other value, say **1**.

With **b** and a guess value **x**, a new guess value is computed with the above formula. This process continues until the new guess value and the current guess value are very close. In this case, either one can be considered as an approximation of the square root of **b**.

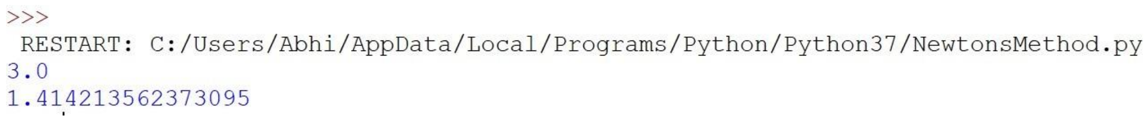
PROGRAM:

```
def newton_method(number, number_iters = 500):
    a = float(number) # number to get square root of
    for i in range(number_iters): # iteration number
        number = 0.5 * (number + a / number) # update
            # x_(n+1) = 0.5 * (x_n +a / x_n)
    return number
print (newton_method(9))
```

Output:

print (newton_method(2))

Output: 1.41421356237



```
>>>
 RESTART: C:/Users/Abhi/AppData/Local/Programs/Python/Python37/NewtonsMethod.py
3.0
1.414213562373095
```

VIVA QUESTIONS-

1. What is square root of a number (Newton's method)?
2. In Python what are iterations?

EXPERIMENT-6

OBJECTIVE:- To write a python program exponentiation (power of a number).

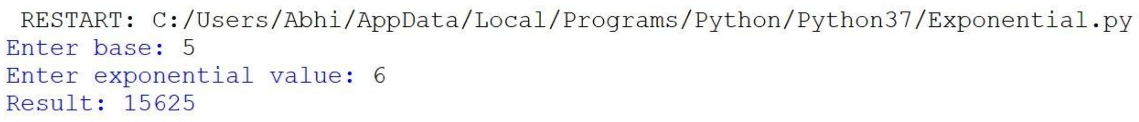
THEORY: Exponentiation (power) is an arithmetic operation on numbers. It is repeated multiplication, just as multiplication is repeated addition. People write exponentiation with upper index. This looks like this: x^y Other methods of mathematical notation have been used in the past. When writing with equipment that cannot use the upper index, people write powers using the ^ or ** signs, so 2^3 or 2**3 means 2^3.

The number x is called base, and the number y is called exponent. For example, in 2^3, 2 is the base and 3 is the exponent.

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))
```

Output:



```
 RESTART: C:/Users/Abhi/AppData/Local/Programs/Python/Python37/Exponential.py
Enter base: 5
Enter exponential value: 6
Result: 15625
```

Program Explanation

a. User must enter the base and exponential value.
b. The numbers are passed as arguments to a recursive function to find the power of the number.
c. The base condition is given that if the exponential power is equal to 1, the base number is returned.
d. If the exponential power isn't equal to 1, the base number multiplied with the power function is called recursively with the arguments as the base and power minus 1.
e. The final result is printed.

VIVA QUESTIONS:

1.Define string operations.
2.What is concatenation function?

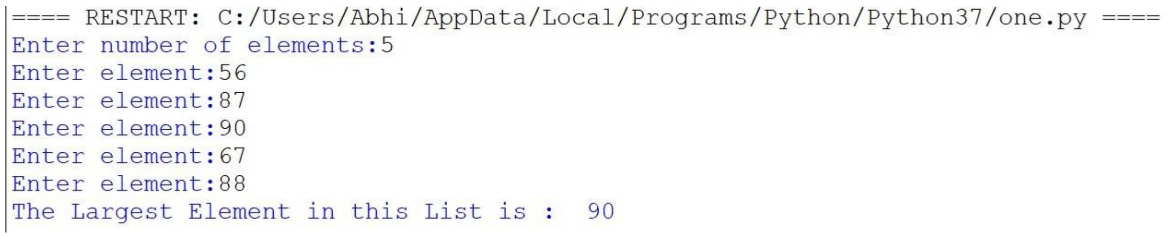
EXPERIEMNT-7

OBJECTIVE:- To write a python program find the maximum of a list of numbers.

PROGRAM:

```
a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:"))
    a.append(b)
'''a.sort()
print("Largest element is:",a[n-1])'''

print("The Largest Element in this List is : ", max(a))

'''a.sort()
a.reverse()

print("The Largest Element in this List is : ", a[0])'''
```

Output:



```
==== RESTART: C:/Users/Abhi/AppData/Local/Programs/Python/Python37/one.py ====
Enter number of elements:5
Enter element:56
Enter element:87
Enter element:90
Enter element:67
Enter element:88
The Largest Element in this List is :  90
```

or

#First way to find max number in list using function

```
def max_num_in_list( list ):

    max = list[ 0 ]

    for a in list:

   if a > max:

            max = a

    return max
```

```
print(max_num_in_list([41, 52, -8, 0]))
```

# Second way to find max number in list using function

```
def max_num_in_list( list ):
    max = list[ 0 ]
    for a  in list:
        if a > max:
            max = a
    return max
ls=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:"))
    ls.append(b)
print("Max number in the list is:",max_num_in_list(ls))'''
```

VIVA QUESTIONS:

1. What is Dynamic programming?
2. Mention what are the rules for local and global variables in Python?

EXPERIMENT-8

OBJECTIVE:- To write a python program linear search.

THEORY: Linear search or sequential search is a method for finding a particular value in a list that checks each element in sequence until the desired element is found or the list is exhausted. The list need not be ordered. Linear search is the simplest search algorithm; it is a special case of brute-force search.

PROGRAM:

```
def Sequential_Search(dlist, item):
pos =  0
found =  False
              while pos < len(dlist) and not found:
 if dlist[pos] == item:
 found = True
else:
pos = pos + 1
 return found, pos
print(Sequential_Search([11,23,58,31,56,77,43,12,65,19],31))
```

Output:

```
True, 3
```

VIVA QUESTIONS:

1. How do the linear search and binary search work??

2. How To implement function in python?

EXPERIMENT-9

OBJECTIVE:- To write a python program Binary search.

THEORY:- Binary search or half-interval search algorithm finds the position of a target value within a sorted array. The binary search can be classified as divide-and conquer search algorithm and executes in logarithmic time. The binary search algorithm begins by comparing the target value to the value of the middle element of the sorted array. If the target value is equal to the middle element's value, then the position is returned and the search is finished. If the target value is less than the middle element's value, then the search continues on the lower half of the array; or if the target value is greater than the middle element's value, then the search continues on the upper half of the array. This process continues, eliminating half of the elements, and comparing the target value to the value of the middle element of the remaining elements - until the target value is either found (and its associated element position is returned), or until the entire array has been searched (and "not found" is returned).

PROGRAM:

```
def binary_search(item_list,item):
        first = 0
        last = len(item_list)-1
        found = False
        while( first<=last and not found):
                mid = (first + last)//2
                 if item_list[mid] == item :
                        found = True
                else:
                if item < item_list[mid]:
                        last = mid - 1
               else:
                        first = mid + 1
        return found
```

```
print(binary_search([1,2,3,5,8],   6))

print(binary_search([1,2,3,5,8],   5))
```

Output:

```
False

True
```

VIVA QUESTIONS:-

1. What is binary search?

2. When does a dictionary is used instead of a list?

EXPERIMENT-10

OBJECTIVE:- To write a python program selection sort.

THEORY:- Selection sort is a sorting algorithm, specifically an in-place comparison sort. The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

PROGRAM:

```
def selectionSort(nlist):
  for fillslot in range(len(nlist)-1,0,-1):
        maxpos=0
        for location  in range(1,fillslot+1):
                if nlist[location]>nlist[maxpos]:
                        maxpos = location
                            temp = nlist[fillslot]
                              nlist[fillslot] = nlist[maxpos]
                              nlist[maxpos] = temp
                              nlist = [14,46,43,27,57,41,45,21,70]
      selectionSort(nlist)
      print(nlist)
```

Output:

```
14      21      27      41      43      45      45      57      70
```

VIVA QUESTIONS

1. What is a tuple?

2. Differentiate between a tuple and a list?

3. What is Sorting?

EXPERIMENT NO 11

OBJECTIVE:- To write a python program Insertion sort.

THEORY:- Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. Each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain. Sorting is typically done in-place, by iterating up the array, growing the sorted list behind it. At each array-position, it checks the value there against the largest value in the sorted list (which happens to be next to it, in the previous array-position checked). If larger, it leaves the element in place and moves to the next. If smaller, it finds the correct position within the sorted list, shifts all the larger values up to make a space, and inserts into that correct position.

PROGRAM:

```
def insertionSort(nlist):
    for index in range(1,len(nlist)):
        currentvalue = nlist[index]
        position = index
        while position>0 and nlist[position-1]>currentvalue:
            nlist[position]=nlist[position-1]
            position = position-1
            nlist[position]=currentvalue
 nlist = [14,46,43,27,57,41,45,21,70]
 insertionSort(nlist)
print(nlist)
```

Output:

```
14    21    27    41    43    45    45    57    70
```

VIVA QUESTIONS

a) What is Stack and how implementation of Stack in c is different from python?

b) Difference between queue and stack?

c) What is selection sort?

EXPERIMENT NO 12

OBJECTIVE :- To write a python program merge sort.

THEORY:- Merge sort is an O(n log n) comparison-based sorting algorithm. Conceptually, a merge sort works as follows:

Divide the unsorted list into n sublists, each containing 1 element (a list of 1 element is considered sorted).

Repeatedly merge sublists to produce new sorted sublists until there is only 1 sublist remaining. This will be the sorted list.

PROGRAM:

```
def mergeSort(nlist):

print("Splitting ",nlist)

    if len(nlist)>1:

        mid = len(nlist)//2

        lefthalf = nlist[:mid]

        righthalf = nlist[mid:]

        mergeSort(lefthalf)

        mergeSort(righthalf)

        i=j=k=0

        while i < len(lefthalf) and j < len(righthalf):

            if lefthalf[i] < righthalf[j]:

            nlist[k]=lefthalf[i]

            i=i+1

            else:

            nlist[k]=righthalf[j]
```

```
                j=j+1
                k=k+1
        while i < len(lefthalf):
            nlist[k]=lefthalf[i]
            i=i+1
            k=k+1
        while j < len(righthalf):
            nlist[k]=righthalf[j]
            j=j+1
            k=k+1
nlist = [14,46,43,27,57,41,45,21,70]
mergeSort(nlist)
print(nlist)
```

Output:

```
14    21    27    41    43    45    45    57    70
```

VIVA QUESTIONS

a) What is queue and how implementation of queue in c is different from python?
b) Difference between queue and stack?
c) What is Merge sort?

Department of Computer Science & Engineering

EXPERIMENT NO 13

OBJECTIVE:- To write a python program first n prime numbers.

THEORY:-

Prime numbers: A prime number is a natural number greater than 1 and having no positive divisor other than 1 and itself.

For example: 3, 7, 11 etc are prime numbers.

Composite number: Other natural numbers that are not prime numbers are called composite numbers.

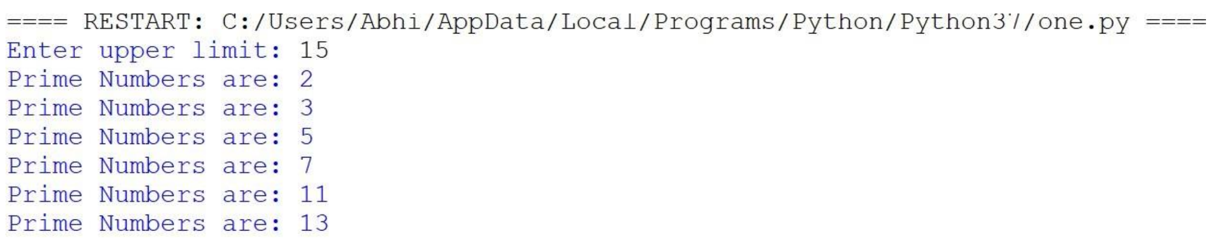
For example: 4, 6, 9 etc. are composite numbers.

Here is source code of the Python Program to check if a number is a prime number.

PROGRAM:

```
r=int(input("Enter upper limit: "))

for a in range(2,r+1):

    k=0

    for i in range(2,a//2+1):

        if(a%i==0):

            k=k+1

    if(k<=0):

        print("Prime Numbers are:",a)
```

Output:



```
==== RESTART: C:/Users/Abhi/AppData/Local/Programs/Python/Python37/one.py ====
Enter upper limit: 15
Prime Numbers are: 2
Prime Numbers are: 3
Prime Numbers are: 5
Prime Numbers are: 7
Prime Numbers are: 11
Prime Numbers are: 13
```

Program Explanation

a. User must enter the upper limit of the range and store it in a different variable.
b. The first for loop ranges till the upper limit entered by the user.
c. The count variable is first initialized to 0.
d. The for loop ranges from 2 to the half of the number so 1 and the number itself aren't counted as divisors.
e. The if statement then checks for the divisors of the number if the remainder is equal to 0.
f. The count variable counts the number of divisors and if the count is lesser or equal to 0, the number is a prime number.
g. If the count is greater than 0, the number isn't prime.
h. The final result is printed.

VIVA QUESTIONS

1. What is write command?
2. What is the difference between write and append?

EXPERIMENT NO 14

OBJECTIVE: To write a python program simulate bouncing ball in Pygame.

PROGRAM:

Main.py

```
from turtle import *

from shapes import *

screen = Screen()

board = Turtle()

board.speed(0)

rectangle(board, 0, 0, 200, 5,  "black")

rectangle(board, 0, 0, 5, 200,  "black")

rectangle(board, 0, 195, 200, 5, "black")

rectangle(board, 195, 0, 5, 200, "black")

board.ht()

ball = Turtle()

ball.speed(0)

screen.register_shape("ball", ((0, 0), (0, 5), (5, 5), (5, 0)))

ball.shape("ball")

ball.penup()

ball.setx(100)

ball.sety(100)
```

```
def collisionCheck(r1x, r1y, r1w, r1h, r2x, r2y, r2w, r2h):
 return (r1x < r2x + r2w) and (r1x + r2w > r2x) and (r1y < r2y + r2h) and (r1h + r1y > r2y)
dx = 1
dy = 1.5
while True:
  ball.setx(ball.xcor() + dx)
  ball.sety(ball.ycor() + dy)
  if collisionCheck(ball.xcor(), ball.ycor(), 5, 5, 0, 0, 200, 5):
    dy = dy * -1
  if collisionCheck(ball.xcor(), ball.ycor(), 5, 5, 0, 0, 5, 200):
    dx = dx * -1
  if collisionCheck(ball.xcor(), ball.ycor(), 5, 5, 0, 195, 200, 5):
    dy = dy * -1
if collisionCheck(ball.xcor(), ball.ycor(), 5, 5, 195, 0, 5, 200):
    dx = dx * -1
```

shapes.py

```
def rectangle(turtle, x, y, width, height, color):
  turtle.penup()
  turtle.setx(x)
  turtle.sety(y)
  turtle.pendown()
  turtle.color(color, color)
turtle.begin_fill()

for i in range(2):

    turtle.forward(width)

 turtle.left(90)

    turtle.forward(height)

  turtle.left(90)

  turtle.end_fill()
```

VIVA QUESTIONS

1. What is the difference write and read mode?
2. What are the symbols used for the mode?

EXPERIMENT NO 15

OBJECTIVE:- Write a program to compute area and circumference of a triangle. Take input from user.

THEORY : Area of Triangle is math.sqrt((s*(s-a)*(s-b)*(s-c)))

Python Code :

```
# Python Program to find Area of a Triangle using Functions

import math

def Area_of_Triangle(a, b, c):

    # calculate the Perimeter
    Perimeter = a + b + c
    # calculate the semi-perimeter
    s = (a + b + c) / 2

    # calculate the area
    Area = math.sqrt((s*(s-a)*(s-b)*(s-c)))

    print("\n The Perimeter of Traiangle = %.2f" %Perimeter);
    print(" The Semi Perimeter of Traiangle = %.2f" %s);
    print(" The Area of a Triangle is %0.2f" %Area)

Area_of_Triangle(6, 7, 8)
```

VIVA QUESTIONS

1. What is formula of area of traingle?
2. What is value of S?

Department of Computer Science & Engineering

EXPERIMENT NO 16

OBJECTIVE:- Write a program to check if a number is Odd or even. Take input from user.

THEORY : If the number is divisible by 2 , that number is called even number, else the number is odd.

PROGRAM :

```
# Python program to check if the input number is odd or even.
# A number is even if division by 2 give a remainder of 0.
# If remainder is 1, it is odd number.
num = int(input("Enter a number: "))
if (num % 2) == 0:
  print("{0} is Even".format(num))
else:
  print("{0} is Odd".format(num))
```

VIVA QUESTIONS

1. Which number is called odd number?
2. What is if… else condition ?

EXPERIMENT NO 17

OBJECTIVE:-Write a program to check that a given year is Leap Year or not

THEORY : If the year is divisible by 4, 100 and 400 then that year is a leap year.

PROGRAM :

```
year = 2000
# To get year (integer input) from the user
# year = int(input("Enter a year: "))
if (year % 4) == 0:
   if (year % 100) == 0:
       if (year % 400) == 0:
           print("{0} is a leap year".format(year))
       else:
           print("{0} is not a leap year".format(year))
   else:
       print("{0} is a leap year".format(year))
else:
   print("{0} is not a leap year".format(year))
```

VIVA QUESTIONS :

1. What do you mean by Leap Year?
2. What is nested if.. else condition?

EXPERIMENT-18

OBJECTIVE:- To print 'n terms of Fibonacci series using iteration.

THEORY: The Fibonacci numbers are the numbers in the following integer sequence.
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8....

The first two terms are 0 and 1. All other terms are obtained by adding the preceding two terms. This means to say the nth term is the sum of (n-1)th and (n-2)th term.

In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation

Fn = Fn-1 + Fn-2 with initial valu F0 = 0 and F1 = 1

```
# Program to display the Fibonacci sequence up to n-th term where n is provided by the user
# change this value for a different result nterms = 10
# uncomment to take input from the user
nterms = int(input("How many terms? "))

# first two terms
n1 = 0
n2 = 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence upto",nterms,":")
   while count < nterms:
       print(n1)
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

VIVA QUESTIONS

1.What is the difference between iteration and recursion?

2.How looping technique is different from Recursion?

APPENDIX

<u>AKTU SYLLABUS</u>

Python Language Programming Lab (KCS453)

1. To write a python program that takes in command line arguments as input and print the number of arguments.

2. To write a python program to perform Matrix Multiplication.

3. To write a python program to compute the GCD of two numbers.

4. To write a python program to find the most frequent words in a text file.

5. To write a python program find the square root of a number (Newton's method).

6. To write a python program exponentiation (power of a number).

7. To write a python program find the maximum of a list of numbers.

8. To write a python program linear search.

9. To write a python program Binary search.

10. To write a python program selection sort.

11. To write a python program Insertion sort.

12. To write a python program merge sort.

13. To write a python program first n prime numbers.

14. To write a python program simulate bouncing ball in Pygame.