THE NEOTIA UNIVERSITY

Object Oriented Programming using Python Laboratory

LAB MANUAL

Course Code : PC-CSE/P/301

Semester : III

Branch : Robotics Engineering

Prepared by

Mr. Kallol Bera

Assistant Professor

Program Outcomes (PO)	
PO-01	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO-02	**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.
PO-03	**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.
PO-04	**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.
PO-05	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations
PO-06	**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.
PO-07	**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.
PO-08	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO-09	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO-10	**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.
PO-11	**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.
PO-12	**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 (PSO)	
PSO-1	**Professional Skills:** Able to utilize the knowledge of aeronautical/aerospace engineering in innovative, dynamic and challenging environment for design and development of new products.
PSO-2	**Professional skills:** Imparted through simulation language skills and general-purpose CAE packages to solve practical, design and analysis problems of components to complete the challenge of airworthiness for flight vehicles.
PSO-3	**Practical implementation and testing skills:** Providing different types of in house and training and industry practice to fabricate and test and develop the products with more innovative technologies.
PSO-4	**Successful Career and Entrepreneurship:** To prepare the students with broad aerospace knowledge to design and develop systems and subsystems of aerospace and allied systems and become technocrats.

ATTAINMENT OF PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES			
S No	Experiment	Program Outcome Attained	Program Specific Outcomes Attained
1	BASICS OF PYTHON	PO-1, PO-2, PO-3	PSO-1, PSO-2
2	CONTROL FLOW	PO-1, PO-2, PO-3	PSO-1, PSO-2
3	STRINGS	PO-1, PO-2, PO-3	PSO-1, PSO-2
4	LIST	PO-2, PO-3	PSO-1, PSO-2
5	MULTI DIMENSIONAL LIST	PO-3, PO-4	PSO-1, PSO-2
6	CLASS	PO-2, PO-3	PSO-1, PSO-2
7	METHODS	PO-2, PO-3	PSO-1, PSO-2
8	CONSTRUCTORS	PO-2, PO-3	PSO-1, PSO-2
9	INHERITANCE	PO-3, PO-4	PSO-1, PSO-2
10	POLYMORPHISM	PO-3, PO-4	PSO-1, PSO-2
11	OVERRIDING MAGIC METHODS	PO-2, PO-3	PSO-1, PSO-2
12	EVENT-DRIVEN PROGRAMMING	PO-2, PO-3	PSO1, PSO2

COURSE OBJECTIVES:

The course should enable the students to:

I. Learn adequate knowledge to write, test, and debug simple Python programs

II. Understand programming skills using the fundamentals and basics of Python Language.

III. Improve problem solving skills using strings, and functions.

IV. Understand the compound data using Python lists, class, methods.

V. Understand the concepts of inheritance, polymorphism and overriding.

LIST OF EXPERIMENTS	
Week-1	BASICS OF PYTHON
Write Python programs for the following: a. To purposefully raise Indentation Error and Correct it. b. To compute distance between two points taking input from the user (Pythagorean Theorem). c. To take numbers as command line arguments and print its sum	
Week-2	CONTROL FLOW
Write Python programs for implementing the following: a. Checking whether the given number is even number or not. b. Finding the factorial of a number. c. Print the prime numbers below 100.	
Week-3	STRINGS
Write Python programs for implementing the following: a. Count the numbers of characters in the string and store them in a dictionary data structure b. Using split and joins methods in the string and trace a birthday with a dictionary data structure.	
Week-4	LIST
Write Python programs to for the following: a. Finding mean, median, mode for the given set of numbers in a list. b. Function dups to find all duplicates in the list.	
Week-5	MULTI DIMENSIONAL LIST
Write Python programs for the following: a. Addition of two square matrices. b. Multiplication of two matrices	
Week-6	CLASS
Write Python programs to implement the following: a. Find the validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These brackets must be close in the correct order, for example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid. b. Get all possible unique subsets from a set of distinct integers.	
Week-7	METHODS
Write Python programs to do the following a. Create a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle. b. Create a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.	
Week-8	CONSTRUCTORS
Write Python program to implement constructors.	
Week-9	INHERITANCE
Write Python program to implement inheritance.	
Week-10	POLYMORPHISM
Write Python program to implement Polymorphism.	
Week-11	OVERRIDING MAGIC METHODS
Write Python program to override Magic Methods	
Week-12	EVENT-DRIVEN PROGRAMMING
Write Python program to create a simple calculator, where the user will enter a number in a text field, and either add it to or subtract it from a running total, which we will display. We will also allow the user to reset the total	
Week-13	EXCEPTION HANDLING
Write a Python program to check Multiple Exception Handling	

WEEK - 1

BASICS OF PYTHON

OBJECTIVE:

a. To purposefully raise Indentation Error and Correct it.
b. To compute distance between two points taking input from the user (Pythagorean Theorem).
c. To take numbers as command line arguments and print its sum

RESOURCE:

Python 3.8

PROGRAM LOGIC:

Compute distance between two points:

1. Read number of terms.
2. Send values to mathematical sqrt function
3. Print the distance

Sum of two numbers:

1. Read two integers n1 and n2.
2. Use arithmetic operator +
3. Print the sum.

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

Script to purposefully raise Indentation Error and Correct it:

```
print("OOP")
  print("Python")
```

output:

```
 print("Python ")
 ^
IndentationError: unexpected indent
```

```
print("OOP")
print("Python ")
```

Output:

```
OOP
Python
```

Compute distance between two points

```
import math
p11 =int(input())
p12 =int(input())
p21 =int(input())
p22 =int(input())
distance = math.sqrt( ((p11-p21)**2)+((p12-p22)**2) )
print(distance)
```

Input:

0 4 6 6

Output:

6.324555320336759

Sum of two numbers

```
import sys
a, b = sys.argv[1:2]
summ = int(a) + int(b)
print("sum is", summ)
```

Input:

2 3

Output:

sum is 5

PRE LAB VIVA QUESTIONS:

a. What is indent in python?
b. What are the three basic data types we have in python?
c. Explain the syntax for comment a statement?

POST LAB VIVA QUESTIONS:

a. What do you understand indentation in python?
b. Which file has to include to use sqrt method?
c. Explain about python language?

WEEK - 2
CONTROL FLOW

OBJECTIVES:

a. To checking whether the given number is even number or not.
b. To find the factorial of a number.
c. To print the prime numbers below 100

RESOURCE:

Python 3.8

PROGRAM LOGIC:

Checking whether the given number is even number or not

1. Read the number.
2. Divide the number by 2
3. Compare the reminder value
4. Display the result

Finding the factorial of a number

1. Read the number.
2. Decrement the value
3. Compare the reminder value
4. Display the result

Print the prime numbers below 100

1. Read the number.
2. Check the number prime or not
3. If yes print the number
4. If not decrement the value
5. Check the number reaches zero.
6. If not goto step2
7. End of the program.

PROCEDURE:

a. Create: Open a new file in Python shell, write a program and save the program with .py extension.
b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

Checking whether the given number is even number or not

```
a=int(input())
if(a%2==0):
        print("Even")
else: print("Odd")
```

Input: 4

Output: Even

Finding the factorial of a number

```
a=int(input())
k=1;
for i in range(1,a+1):
        k=k*i;
print("The factorial of given number is", k)
```

input: 5
Output:
The factorial of given number is 120

Print the prime numbers below 100

```
for i in range(2,100):
    c=0;
    for j in range(1,i+1):
        if(i%j==0):
            c=c+1
    if(c<=2):
        print(i,end =" "),
```

Output: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

PRE LAB VIVA QUESTIONS:

a. Define loop process?
b. How many loops are there?
c. Explain the syntax of for loop?

POST LAB VIVA QUESTIONS:

a. What do you understand by loop?
b. Explain the nested control statements?
c. Differentiate between for loop and while loop d. On which data types only for loop applicable

WEEK - 3

STRINGS

OBJECTIVES:

a. To implement to count the numbers of characters in the string and store them in a dictionary data structure

b. To use split and joins methods in the string and trace a birthday with a dictionary data structure

RESOURCE:

Python 3.8

PROGRAM LOGIC:

Count the numbers of characters in the string

1. Read the string.
2. Count the characters
3. Display the result

PROCEDURE:

a. Create: Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

Count the numbers of characters in the string

```
str1=input()
dict = {}
for n in str1:
        keys = dict.keys()
        if n in keys:
                dict[n] += 1
        else:
                dict[n] = 1
print(dict)
```

Input:

Success

Output:

{'s': 3, 'u': 1, 'c': 2, 'e': 1}

Split and joins methods in the string

```
birthdays = {'Alice': 'Apr 1 1998', 'Bob': 'Dec 12,2001', 'Carol': 'Mar 4,2002'}
name=input()
if name in birthdays:
        s=birthdays[name];
        L=s.split(" ")
        k="-"
        L=k.join(L)
```

```
        print(L + ' is the birthday of ' + name)
    else:
        print('I do not have birthday information for ' + name)
```

Input:

Alice

Output:

Apr-1-1998 is the birthday of Alice

PRE LAB VIVA QUESTIONS:

a. Define the syntax of dictionary?
b. What is the difference between List and Dictionary?
c. Explain the syntax of List slicing?

POST LAB VIVA QUESTIONS:

a. What are the various methods used on list objects?
b. Explain the purpose of dictionary?
c. Parameters used in get() of dictionary variable
d. Explain different ways to display dictionary elements

WEEK - 4

LIST

OBJECTIVES:

a. To finding mean, median, mode for the given set of numbers in a list.

b. To function dups to find all duplicates in the list.

RESOURCE:

Python 3.8

PROGRAM LOGIC:

To find mean, median, mode for the given set of numbers in a list.

1. Read the elements into a list.
2. Calculate the sum of list elements.
3. Calculate the mean, median.
4. Display the result.

Function dups to find all duplicates in the list.

1. Create a list to read elements.
2. Pass the list as parameter to dup function.
3. Define function dup to identify duplicate elements in the list.
4. Return the final list to called function.
5. Display the result.

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

To find mean, median, mode for the given set of numbers in a list

```
from collections import Counter
n_num = [1, 2, 3, 4, 5,5]
n = len(n_num)
get_sum = sum(n_num)
mean = get_sum / n
nnum=n_num;
print("Mean / Average is: " + str(mean))
nnum.sort()

if n % 2 == 0:
        median1 = nnum[n//2]
        median2 = nnum[n//2 - 1]
        median = (median1 + median2)/2
else:
        median = nnum[n//2]
print("Median is: " + str(median))
data = Counter(n_num)
```

```
get_mode = dict(data)
mode = [k for k, v in get_mode.items() if v == max(list(data.values()))]
if len(mode) == n:
        get_mode = "No mode found"
else:
        get_mode = "Mode is / are: " + ', '.join(map(str, mode))
        print(get_mode)
```

Output:

```
Mean / Average is: 3.3333333333333335
Median is: 3.5
Mode is / are: 5
```

Function dups to find all duplicates in the list.

```
def Remove(duplicate):
        final_list = []
        for num in duplicate:
                if num not in final_list:
                        final_list.append(num)
        return final_list

# Driver Code
duplicate = [2, 4, 10, 20, 5, 2, 20, 4]
print(Remove(duplicate))
```

Output:

```
[2, 4, 10, 20, 5]
```

PRE LAB VIVA QUESTIONS:

a. Define the syntax of function definition?
b. What is a parameter?
c. Explain the empty List()

POST LAB VIVA QUESTIONS:

a. What are the various methods used on list objects?
b. Explain the purpose of function?
c. How we pass list to a function d. Differentiate between list and dictionary

WEEK - 5
MULTI DIMENSIONAL LIST

OBJECTIVES:

a. Write a Python script for addition of two square matrices.
b. Write a Python script for multiplication of two matrices.

RESOURCE:

Python 3.8

PROGRAM LOGIC:

Addition of two square matrices.

1. Create a lists to read matrix elements
2. Read the elements of to matrices add the elements
3. Store the result in third matrix.
4. Repeat steps 2 and 3 till the addition of all elements
5. Display the result

Multiplication of two matrices

1. Create a lists to read matrix elements
2. Read the elements of to matrices, multiply the elements
3. Store the result in third matrix.
4. Repeat steps 2 and 3 till the multiplication of all elements
5. Display the result.

PROCEDURE:

a. Create: Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

Addition of two square matrices

```
X =      [[12,7,3],
          [4 ,5,6],
          [7 ,8,9]]
Y =      [[5,8,1],
          [6,7,3],
          [4,5,9]]
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

    # iterate through rows
    for i in range(len(X)):
            # iterate through columns
```

```
            for j in range(len(X[0])):
                    result[i][j] = X[i][j] + Y[i][j]
    for r in result:
            print(r)
```

Output:

```
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
```

Multiplication of two matrices

```
    X =     [[12,7,3],
            [4 ,5,6],
            [7 ,8,9]]
    # 3x3 matrix
    Y =     [[5,8,1,2],
            [6,7,3,0],
            [4,5,9,1]]
    # result is 3x4
    result = [[0,0,0,0],
            [0,0,0,0],
            [0,0,0,0]]
    # iterate through rows of X
    for i in range(len(X)):
            # iterate through columns of Y
            for j in range(len(Y[0])):
                    # iterate through rows of Y
                    for k in range(len(Y)):
                            result[i][j] += X[i][k] * Y[k][j]
    for r in result:
            print(r)
```

Output:

```
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 23]
```

PRE LAB VIVA QUESTIONS:

a. Define the multidimensional list?
b. What is a parameter?
c. Explain the various methods of List object

POST LAB VIVA QUESTIONS:

a. What is difference between List and tuple?
b. Explain the range function?
c. Explain various arguments used in range function d. Explain the slicing list elements

WEEK - 6

CLASS

OBJECTIVES:

a. Find the validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These brackets must be close in the correct order, for example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid.

b. Get all possible unique subsets from a set of distinct integers.

RESOURCE:

Python 3.8

PROGRAM LOGIC:

Validity of a string of parentheses

1. Create a list to read string of parentheses
2. Read the open, close parentheses into two lists
3. Compare the open and close list parentheses.
4. Display the result

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
open_list = ["[","{","("]
close_list = ["]","}",")"]
def check(myStr):
    stack = []
    for i in myStr:
        if i in open_list:
            stack.append(i)
        elif i in close_list:
            pos = close_list.index(i)
            if ((len(stack) > 0) and (open_list[pos] == stack[len(stack)-1])):
                stack.pop()
            else:
                return "Unbalanced"
    if len(stack) == 0:
        return "Balanced"
string = "{[]{()}}"
print(string,"-", check(string))
string = "[{}{})()]"
print(string,"-", check(string))
```

Output:

```
{[]{()}} - Balanced
[{}{})() - Unbalanced
```

Get all possible unique subsets from a set of distinct integers.

```
class py_solution:
    def sub_sets(self, sset):
            return self.subsetsRecur([], sorted(sset))
    def subsetsRecur(self, current, sset):
      if sset:
            return self.subsetsRecur(current, sset[1:]) + self.subsetsRecur(current + [sset[0]], sset[1:])
      return [current]
print(py_solution().sub_sets([4,5,6]))
```

Output:

[[], [6], [5], [5, 6], [4], [4, 6], [4, 5], [4, 5, 6]]

PRE LAB VIVA QUESTIONS:

a. Define the class?
b. What is an object?

POST LAB VIVA QUESTIONS:

a. What is stack?
b. Explain the recursion function?
c. Which constructor used to create empty list?
d. Explain the stack operations?
e. What are the linear data structures?

WEEK - 7
METHODS

OBJECTIVE:

a. Create a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

b. Create a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.

RESOURCE:

Python 3.8

PROGRAM LOGIC:

Validity of a string of parentheses

1. Create a class circle
2. Define a method to calculate radius
3. Create object to call the methods of class
4. Display the result

PROCEDURE:

a. Create: Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

The area and the perimeter of a circle.

```
class Circle():
        def __init__(self, r):
                self.radius = r
        def area(self):
                return self.radius**2*3.14
        def perimeter(self):
                return 2*self.radius*3.14
NewCircle = Circle(8)
print(NewCircle.area())
print(NewCircle.perimeter())
```

Output:

```
200.96
50.24
```

The area of a rectangle.

```
class Rectangle():
        def __init__(self, l, w):
                self.length = l
                self.width = w
        def rectangle_area(self):
                return self.length*self.width
newRectangle = Rectangle(12, 10)
```

```
print(newRectangle.rectangle_area())
```

Output:

120

PRE LAB VIVA QUESTIONS:

a. How we create object to a class?
b. How we call the method of a class?

POST LAB VIVA QUESTIONS:

a. What is instance of a class?
b. Write the syntax to create a list object?
c. Write the syntax for creation of empty list?

WEEK - 8
CONSTRUCTORS

OBJECTIVE:

Write Python program to implement constructors.

RESOURCE:

Python 3.8

SOURCE CODE:

```
class NITMAS:
        geek = ""
        def __init__(self):
                self.geek = "NITMAS"
        def print_Geek(self):
                print(self.geek)
obj = NITMAS()
obj.print_Geek()
```

Output:

NITMAS

PRE LAB VIVA QUESTIONS:

a. What is a constructor?
b. When constructor is executed?

POST LAB VIVA QUESTIONS:

a. What is the use of constructor?
b. When constructor is executed?
c. Which constructor executes when creating a list object?

WEEK - 9
INHERITANCE

OBJECTIVE:

Write Python program to implement inheritance

RESOURCE:

Python 3.8

SOURCE CODE:

```
class Person(object):
        def __init__(self, name):
                self.name = name
        def getName(self):
                return self.name
        def isEmployee(self):
                return False
class Employee(Person):
        def isEmployee(self):
                return True
emp = Person("PYTHON")
print(emp.getName(), emp.isEmployee())
emp = Employee("OOPS")
print(emp.getName(), emp.isEmployee())
```

Output:

```
PYTHON False
OOPS True
```

PRE LAB VIVA QUESTIONS:

a. What is inheritance?
b. Multiple inheritance means what?

POST LAB VIVA QUESTIONS:

a. What is the use of inheritance?
b. Explain the multiple inheritances?
c. Explain polymorphism?

WEEK - 10
POLYMORPHISM

OBJECTIVE:

Write Python program to implement Polymorphism.

RESOURCE:

Python 3.8

PROGRAM LOGIC:

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
def add (x, y, z = 0):
        return x + y+z
print (add (7, 3))
print (add (7, 3, 4))
```

Output:

```
10
14
```

PRE-LAB VIVA QUESTIONS:

a. What is polymorphism?

b. Explain object-oriented features?

c. Explain the importance of polymorphism?

POST LAB VIVA QUESTIONS:

a. What is the use of polymorphism?

b. Differentiate between inheritance and polymorphism?

c. How we implement polymorphism in python?

WEEK - 11

OVERRIDING MAGIC METHODS

OBJECTIVE:

Write Python program to override Magic Methods.

RESOURCE:

Python 3.8

PROGRAM LOGIC:

PROCEDURE:

a. Create: Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
class Bird:
        def intro(self):
                print("There are many types of birds.")
        def flight(self):
                print("Most of the birds can fly but some cannot.")
class sparrow(Bird):
        def flight(self):
                print("Sparrows can fly.")
class ostrich(Bird):
        def flight(self):
                print("Ostriches cannot fly.")
obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()
obj_bird.intro()
obj_bird.flight()
obj_spr.intro()
obj_spr.flight()
obj_ost.intro()
obj_ost.flight()
```

Output:

```
There are many types of birds.
Most of the birds can fly but some cannot.
There are many types of birds.
Sparrows can fly.
There are many types of birds.
Ostriches cannot fly.
```

PRE LAB VIVA QUESTIONS:

a. What is method over loading?

b. Explain method overriding?

c. Explain the importance method overloading?

POST LAB VIVA QUESTIONS:

a. How compiler knows which function has to execute at the time of method over loading?
b. Explain the various parameters used in function calling?
c. Differentiate between function calling by reference and call by value

WEEK - 12
EVENT-DRIVEN PROGRAMMING

OBJECTIVE:

Write Python program to create a simple calculator, where the user will enter a number in a text field, and either add it to or subtract it from a running total, which we will display. We will also allow the user to reset the total.

RESOURCE:

Python 3.8

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
def add(x, y):
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
    return x / y
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
choice = input("Enter choice(1/2/3/4):")
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if choice == '1':
    print(num1,"+",num2,"=", add(num1,num2))
elif choice == '2':
    print(num1,"-",num2,"=", subtract(num1,num2))
elif choice == '3':
    print(num1,"*",num2,"=", multiply(num1,num2))
elif choice == '4':
    print(num1,"/",num2,"=", divide(num1,num2))
else:
    print("Invalid input")
```

Input and Output:

```
Select operation.
1.Add
2.Subtract
3.Multiply
```

4.Divide
Enter choice (1/2/3/4):1
Enter first number: 3
Enter second number: 4
3 + 4 = 7

PRE-LAB VIVA QUESTIONS:

a. What is the predefined function required?
b. Which files we are importing to execute?
c. Explain the importance function declaration?

POST LAB VIVA QUESTIONS:

a. Explain the object creation in python?
b. Explain the exception handling?
c. Explain try catch in python

WEEK - 13
EXCEPTION HANDLING

OBJECTIVE:

Write a Python program to check Multiple Exception Handling

RESOURCE:

Python 3.8

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .py extension.

b. Execute : Go to Run -> Run module (F5)

SOURCE CODE:

```
# import module sys to get the type of exception
import sys

randomList = ['a', 0, 2]

for entry in randomList:
        try:
                print("The entry is", entry)
                r = 1/int(entry)
                break
        except:
print("Oops!",sys.exc_info()[0],"occured.")
print("Next entry.")
print()
print("The reciprocal of",entry,"is",r)
```

OR,

```
import math

def square(x):
        if int(x) is 0:
                raise ValueError('Input argument cannot be zero')
        if int(x) < 0:
                raise TypeError('Input argument must be positive integer')
                return math.pow(int(x), 2)
        while True:

                try:
                        y = square(input('Please enter a number\n'))
                        print(y)
                except ValueError as ve:
```

```
                print(type(ve), '::', ve)
        except TypeError as te:
                print(type(te), '::', te)
while True:

        try:
                y = square(input('Please enter a number\n'))
                print(y)
        except (ValueError, TypeError) as e:
                print(type(e), '::', e)
```

PRE-LAB VIVA QUESTIONS:

a. What is the predefined function required?
b. Explain the exception handling?
c. Explain try catch in python

POST LAB VIVA QUESTIONS:

a. Explain the object creation in python?