J.B. INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

(Accredited by NBA & NAAC, Approved by AICTE & Permanently Affiliated to JNTUH)

Bhaskar Nagar, Yenkapally (V), Moinabad(M), P.O. Himayathnagar, R.R. District, Hyderabad-5000075.

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

OBSERVATION

PYTHON PROGRAMMING LAB

(R20)

Name of the Student:

Roll No:

Class & Section:

Index

6.2		Write python programs to Create a list of animal using dictionary variable “animal” and find out if the specific animal present in the list or not?	
7.1		Write a python program to create a class, its objects and accessing attributes	
7.2		Create a Customer class and check the balance and withdraw and deposit some amount	
8.1		Write a python script to implement exception handling. Check whether the input no is integer or not.	
8.2		Write a python script to implement exception handling. Handel the exceptions that are come at the time of division	
9		Write a python script to perform inheritance	
10		Write a python script to perform various FILE handling operations. Open, close, read, write, copy.	
11.1		Write a python script to connect to the database and perform DDL operations	
11.2		Create table, insert data into table and display the table data.	
12		Write a python script to connect to the database and perform various DML and DQL operations.	

Experiment 1:

1.1. Write a python program to obtain user input data (int, float, string) and display.

Code:

```
#program to obtain user input data (int, float, string) and display.
a =int(input("enter an integer :"))
b=input("enter an string :")
c=float(input("enter an float value :"))
print("you have entered integer as :", a)
print("you have entered string as :",b)
print("you have entered float value as :",c)
```

Result:

```
enter an integer :1635
enter an string : Raj
enter an float value :23.56
you have entered integer as : 1635
you have entered string as : Raj
you have entered float value as : 23.56
```

1.2. Write a python program to find the roots of a quadratic equation

Code:

#To find the roots of a quadratic equation

```
import math
a=float(input("Enter the value of 'a' :"))
b=float(input("Enter the value of 'b' :"))
c=float(input("Enter the value of 'c' :"))
d = b**2-4*a*c
if d < 0:
    print ("This equation has no real solution")
elif d == 0:
    e = (-b+math.sqrt(b**2-4*a*c))/2*a
    print ("This equation has one solutions: ", e)
else:
    f = (-b+math.sqrt(b**2-4*a*c))/2*a
    g = (-b-math.sqrt(b**2-4*a*c))/2*a
    print ("This equation has two solutions: ", f, " and", g)
```

Result:

```
Enter the value of 'a' :1

Enter the value of 'b' :3

Enter the value of 'c' :2

This equation has two solutions:  -1.0 and -2.0


Enter the value of 'a' :1

Enter the value of 'b' :3

Enter the value of 'c' :3

This equation has no real solution
```

1.3. Write a python program to perform arithmetic operations (+, -, *, /, %) for given input values and printout the result values.

Code:

#To perform arithmetic operations

```
a=float(input("enter the 1st no :"))
b=float(input("enter the 2nd no :"))
c=a+b;
d=a-b;
e=a*b;
f=a/b;
g=a%b
h=a//b
i=a**b
print("addition is :",c)
print("substraction is :",d)
print("multiplication is :",e)
print("division is :",f)
print("modulo is :",g)
print("floor division is :",h)
print("Exponent is :",i)
```

Result:

```
enter the 1st no :10
enter the 2nd no :3
addition is : 13.0
subtraction is : 7.0
multiplication is : 30.0
division is : 3.3333333333333335
modulo is : 1.0
floor division is : 3.0
Exponent is : 1000.0
```

2.1. Write a python programs that use both recursive and non-recursive functions to find the factorial of a given integer.

```
#2.1 To find the factorial of a given integer(recursive)
def factorial(n):
    if(n==1 or n==0):
        return 1
    else:
        return n * factorial(n-1)

print(factorial(int(input("enter the no : "))))
```

```
#To find the factorial of a given integer(Non-Recursive)

num = int(input("Enter a number: "))
factorial = 1
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i

print("The factorial of",num,"is",factorial)
```

OUTPUT:

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

2.2 Operators and Operands in Python: (Arithmetic, relational and logical Operators), operator precedence, Expressions and Statements.

```
#program for demonstrating Arithmetic Operations
#op1.py
a=float(input("Enter First Value:"))
b=float(input("Enter Second Value:"))
print("\t--------------------------------------------")
print("\tA r i t h m e t i c O p e r a t i o n s")
print("\t--------------------------------------------")
print("\t\tsum({},{})={}".format(a,b,a+b))
print("\t\tsub({},{})={}".format(a,b,a-b))
print("\t\tmul({},{})={}".format(a,b,a*b))
print("\t\tdiv({},{})={}".format(a,b,a/b))
print("\t\tFloorDiv({},{})={}".format(a,b,a//b))
print("\t\tmod({},{})={}".format(a,b,a%b))
print("\t\texpo({},{})={}".format(a,b,a**b))
print("\t--------------------------------------------")
```

OUTPUT:

```
Enter First Value:10
Enter Second Value:3
        --------------------------------------------
        A r i t h m e t i c O p e r a t i o n s
        --------------------------------------------
                sum(10.0,3.0)=13.0
                sub(10.0,3.0)=7.0
                mul(10.0,3.0)=30.0
                div(10.0,3.0)=3.3333333333333335
                FloorDiv(10.0,3.0)=3.0
                mod(10.0,3.0)=1.0
                expo(10.0,3.0)=1000.0
        --------------------------------------------
```

```
#Program for demonstrating Relational Operators
#op2.py
a=int(input("Enter Value of a:"))
b=int(input("Enter Value of b:"))
c=int(input("Enter Value of c:"))
print("--------------------------------------------------")
print("\tResult of Relational Operators")
print("--------------------------------------------------")
print("\t\t{} > {}={}".format(a,b,a>b))
print("\t\t{} > {}={}".format(b,a,b>a))
print("\t\t{} < {}={}".format(a,c,a<c))
print("\t\t{} < {}={}".format(c,a,c<a))
print("\t\t{} == {}={}".format(a,b,a==b))
print("\t\t{} != {}={}".format(a,b,a!=b))
print("\t\t{} >= {}={}".format(b,c,b>=c))
print("\t\t{} <= {}={}".format(b,c,b<=c))
print("--------------------------------------------------")
```

OUTPUT:

```
Enter Value of a:125
Enter Value of b:25
Enter Value of c:544
--------------------------------------------------
        Result of Relational Operators
--------------------------------------------------
                125 > 25=True
                25 > 125=False
                125 < 544=True
                544 < 125=False
                125 == 25=False
                125 != 25=True
                25 >= 544=False
                25 <= 544=True
--------------------------------------------------
```

```
#Program for demonstrating Logical Operators
#op3.py
a=int(input("Enter Value of a:"))
b=int(input("Enter Value of b:"))
print("--------------------------------------------------")
print("\tResult of Relational & Logical Operators")
print("--------------------------------------------------")
if(a!=0 and a<100):
    print("number {} is in between 0 to 100".format(a))
else:
    print("number {} is greater than 100".format(a))
if(b==0 or b<a):
    print("number {} is smaller num".format(b))
else:
    print("number {} is greater than num {}".format(b,a))
```

OUTPUT:

```
Enter Value of a:155
Enter Value of b:225
--------------------------------------------------
        Result of Relational & Logical Operators
--------------------------------------------------
number 155 is greater than 100
number 225 is greater than num 155
```

#Program for Operator precedence
#op4.py

```
a=int(input("Enter Value of a:"))
b=int(input("Enter Value of b:"))
c=int(input("Enter Value of c:"))
d=int(input("Enter Value of d:"))
e=int(input("Enter Value of e:"))
f=int(input("Enter Value of e:"))
g = a + (b * c) / d;
print("Value of a + (b * c) / d is ",g)
h = a+b*c%d/e
print("Value of a+b*c%d/e is ",h)
i=a**b//c%d/e*f
print("Value of a**b//c%d/e*f is ",i)
```

Output:

```
Enter Value of a:15
Enter Value of b:25
Enter Value of c:1
Enter Value of d:4
Enter Value of e:3
Enter Value of e:7
Value of a + (b * c) / d is  21.25
Value of a+b*c%d/e is  15.333333333333334
Value of a**b//c%d/e*f is  7.0
```

3. (Assignment statement); Taking input (using raw input () and input ()) and displaying output (print statement); Putting Comments.

```
# hello statement.
"""welcome to JBIET
CSE Department"""

a=input("enter your gender : ")
b=input("enter your name : ")
if(a=="m" or a=="male" or a=="MALE" or a=="M"):
  print("Hello Mr.",b)
elif(a=="f" or a=="female" or a=="FEMALE" or a=="F"):
  print("Hello Miss. ",b);
else:
  print("Enter Correct gender");
```

Output:

```
enter your gender : male
enter your name : rAJ
Hello Mr. rAJ
```

Experiment 3:

i. **Write python programs to perform operation on Strings using following functions: len, capitalize, find, isalnum, isalpha, isdigit, lower, islower, isupper, upper, lstrip, rstrip, isspace, istitile, partition, replace, join, split, count, decode, encode, swapcase.**

Code:

```
x="welcome to $JBIET Engineering College \n Hyderabad"

print(x)

print(len(x))

print("capitalize word:",x.capitalize())

y = input("enter the finding String : ")

print("your given string starts from ",(x.find(y)+1)," position.")

d="sukanya101" #returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).

print(d.isalnum())

#isalpha:  returns “True” if all characters in the string are alphabets, Otherwise, It returns “False”.

# checking for alphabets

string = 'Ayush'

print(string.isalpha())

string = 'Ayush0212'

print(string.isalpha())

# Python program to illustrate counting number of alphabets using isalpha()

  # Given string

string5='Ayurvedic Medicine'

count=0

# Initialising new strings

newstring1 =""

newstring2 =""

# Iterating the string and checking for alphabets

# Incrementing the counter if an alphabet is found

# Finally printing the count

for a in string5:

    if (a.isalpha()) == True:
        count+=1
        newstring1+=a
print(count)
print(newstring1)
# Given string
string6='Ayush0212'
count=0
for a in string6:
    if (a.isalpha()) == True:
        count+=1
        newstring2+=a
print(count)
print(newstring2)
print(x.isdigit())
print("lower     :",x.lower())
print("islower   :",x.islower())
print("isupper   :",x.isupper())
print("upper     :",x.upper())
print("lstrip    :",x.lstrip())
print("rstrip    :",x.rstrip())
print("rstrip    :",x.rstrip('$'))
print("isspace   :",x.isspace())
print("istitle   :",x.istitle())
print("partition :",x.partition("to"))
print("replace   :",x.replace("welcome","wish",1))
print("join      :")
str = ""   # string
list = ['p','y','t','h','o','n',' ','p','r','o','g','r','a','m','m','i','n','g']    # iterable
# Calling function
str2 = str.join(list)
# Displaying result
print(str2)
```

```
# decode, encode, swapcase.
print("split    :")
#str.split(separator, maxsplit)
print(x.split())
print("Count    :",x.count("to"))
print("Encode and Decode")
# encoding string
str_original = input('Please enter string data:\n')
bytes_encoded = str_original.encode()
str_decoded = bytes_encoded.decode()
print('Encoded bytes =', bytes_encoded)
print('Decoded String =', str_decoded)
print('str_original equals str_decoded =', str_original == str_decoded)
print(x.swapcase())
```

Output:

```
welcome to $JBIET Engineering College
 Hyderabad
49
capitalize word: Welcome to $jbiet engineering college
 hyderabad
enter the finding String : college
your given string starts from  0  position.
True
True
False
17
AyurvedicMedicine
5
Ayush
False
lower     : welcome to $jbiet engineering college
 hyderabad
```

```
islower   : False
isupper   : False
upper     : WELCOME TO $JBIET ENGINEERING COLLEGE
 HYDERABAD
lstrip    : welcome to $JBIET Engineering College
 Hyderabad
rstrip    : welcome to $JBIET Engineering College
 Hyderabad
rstrip    : welcome to $JBIET Engineering College
 Hyderabad
isspace   : False
istitle   : False
partition : ('welcome ', 'to', ' $JBIET Engineering College \n Hyderabad')
replace   : wish to $JBIET Engineering College
 Hyderabad
join      :
python programming
split    :
['welcome', 'to', '$JBIET', 'Engineering', 'College', 'Hyderabad']
Count    : 1
Encode and Decode
Please enter string data:
Please enter string data:to
Encoded bytes = b'Please enter string data:to'
Decoded String = Please enter string data:to
str_original equals str_decoded = True
WELCOME TO $jbiet eNGINEERING cOLLEGE
 hYDERABAD
```

ii. Enter the details of 5 Student and display the details sequentially.

Code:

```
print("-----Program for Student Information-----")
D = dict()
n = int(input('How many student record you want to store?? '))
# Add student information to the dictionary
for i in range(0,n):
    x, y = input("Enter the complete name (First and last name) of student: ").split()
    z = input("Enter contact number: ")
    m = input('Enter Marks: ')
    D[x, y] = (z, m)
# define a function for shorting names based on first name
def sort():
    ls = list()
    # fetch key and value using
    # items() method
    for sname,details in D.items():
        # store key parts as an tuple
        tup = (sname[0],sname[1])
        # add tuple to the list
        ls.append(tup)
    # sort the final list of tuples
    ls = sorted(ls)
    for i in ls:
        # print first name and second name
        print(i[0],i[1])
    return
# define a function for finding the minimum marks  in stored data
def minmarks():
    ls = list()
    # fetch key and value using
    # items() methods
    for sname,details in D.items():
        # add details second element
        # (marks) to the list
        ls.append(details[1])
    # sort the list elemnts
    ls = sorted(ls)
    print("Minimum marks: ", min(ls))
    return
# define a function for searching student contact number
def searchdetail(fname):
    ls = list()
    for sname,details in D.items():
        tup=(sname,details)
        ls.append(tup)
    for i in ls:
        if i[0][0] == fname:
            print(i[1][0])
    return
# define a function for asking the options
def option():
    choice = int(input('Enter the operation detail: \n \
    1: Sorting using first name \n \
    2: Finding Minimum marks \n \
    3: Search contact number using first name: \n \
    4: Exit\n \
    Option: '))
    if choice == 1:
        # function call
        sort()
        print('Want to perform some other operation??? Y or N: ')
        inp = input()
        if inp == 'Y':
            option()
        # exit function call
        exit()
    elif choice == 2:
        minmarks()
        print('Want to perform some other operation??? Y or N: ')
        inp = input()
        if inp == 'Y':
            option()
        exit()
    elif choice == 3:
        first = input('Enter first name of student: ')
        searchdetail(first)
        print('Want to perform some other operation??? Y or N: ')
        inp = input()
        if inp == 'Y':
            option()
        exit()
    else:
        print('Thanks for executing me!!!!')
        exit()
option()
```

Output:

```
-----Program for Student Information-----
How many student record you want to store?? 5
Enter the complete name (First and last name) of student: raj kumar
Enter contact number: 6789134567
Enter Marks: 67
Enter the complete name (First and last name) of student: rama krishna
Enter contact number: 8712567894
Enter Marks: 78
Enter the complete name (First and last name) of student: niharika reddy
Enter contact number: 2789675678
Enter Marks: 56
Enter the complete name (First and last name) of student: sohel kumar
Enter contact number: 2897565876
Enter Marks: 89
Enter the complete name (First and last name) of student: rahul kumar
Enter contact number: 4788934567
Enter Marks: 57
Enter the operation detail:
    1: Sorting using first name
    2: Finding Minimum marks
    3: Search contact number using first name:
    4: Exit
    Option: 2
Minimum marks:  56
Want to perform some other operation??? Y or N:
N
```

Experiment 4:

i. **Write python programs to perform List operators: (joining, list slices)**

Code:

```
# Initialize list
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]
# Show original list
print("\nOriginal List:\n", List)
print("\nSliced Lists: ")
# Display sliced list
print(List[3:9:2])
# Display sliced list
print(List[::2])
# Display sliced list
print(List[::])
# Initialize list
List1 = ['JBIET', 4, ' !']
# Show original list
print("\nOriginal List:\n", List1)
print("\nSliced Lists: ")
# Display sliced list
print(List1[::-1])
# Display sliced list
print(List1[::-3])
# Display sliced list
print(List1[:1:-2])
```

Output:

```
Original List:
 [1, 2, 3, 4, 5, 6, 7, 8, 9]

Sliced Lists:
[4, 6, 8]
[1, 3, 5, 7, 9]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]

Original List:
 ['JBIET', 4, ' !']

Sliced Lists:
[' !', 4, 'JBIET']
[' !']
[' !']
```

ii. Write python programs to perform List functions: len, insert, append, extend, sort, remove, and reverse, pop.

Code:

```
prices = [238.11, 237.81, 238.91]
prices.sort()
print("prices  :",prices)
fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]
print("fam     :",fam)
fam2 = [["liz", 1.73],
        ["emma", 1.68],
        ["mom", 1.71],
        ["dad", 1.89]]
print("fam2     :",fam2)
prices.append(456.98)
print("prices   :",prices)
prices.extend([34.8])
print(prices)
months = ['January', 'February', 'March']
print(months.index('February'))
print(months)
print(prices[1])
price_max = max(prices)
print("price maximun:",price_max)

min_price = min(prices)
# Identify min price index
min_index = prices.index(min_price)
print("Minimum Price          :",min_index)
print('stock_prices length is :', len(prices))
print("list of Prices         :",list(prices))
```

Output:

```
prices  : [237.81, 238.11, 238.91]
fam     : ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
fam2     : [['liz', 1.73], ['emma', 1.68], ['mom', 1.71], ['dad', 1.89]]
prices   : [237.81, 238.11, 238.91, 456.98]
[237.81, 238.11, 238.91, 456.98, 34.8]
1
['January', 'February', 'March']
238.11
price maximun: 456.98
Minimum Price          : 4
stock_prices length is : 5
list of Prices         : [237.81, 238.11, 238.91, 456.98, 34.8]
```

iii. Write python programs to check whether the string is palindrome or not?

Code:

```
x = "malayalam"
w = ""
for i in x:
    w = i + w
if (x == w):
    print("Yes")
else:
    print("No")
```

Output:

```
Yes
```

Experiment 5:

i. **Write python programs to perform Tuple functions: cmp(), len(), max(), min(), tuple()**

Code:

```
mytuple = ("apple", "banana", "cherry")
print(mytuple)
print("allow Duplicates")
mytuple = ("apple", "banana", "cherry", "apple", "cherry")
print(mytuple)
print("Length of a Tuple:",len(mytuple))
print("Create Tuple With One Item")
mytuple = ("apple",)
print(type(mytuple))

#NOT a tuple
mytuple = ("apple")
print(type(mytuple))
print("Tuple Items - Data Types")
tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)
print(tuple1)
print(tuple2)
print(tuple3)
tuple1 = ("abc", 34, True, 40, "male")
print(tuple1)
print("Tuple Constructor")
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
print(thistuple)
```

Output:

```
('apple', 'banana', 'cherry')
```

```
allow Duplicates
('apple', 'banana', 'cherry', 'apple', 'cherry')
Length of a Tuple: 5
Create Tuple With One Item
<class 'tuple'>
<class 'str'>
Tuple Items - Data Types
('apple', 'banana', 'cherry')
(1, 5, 7, 9, 3)
(True, False, False)
('abc', 34, True, 40, 'male')
Tuple Constructor
('apple', 'banana', 'cherry')
```

ii. **Write python programs to check whether the word is present in the tuple or not?**

Code:

```
# initialize tuple
test_tup = (10, 4, 5, 6, 8)
# printing original tuple
print("The original tuple : " + str(test_tup))
# initialize N
N = 6
# Check if element is present in tuple
# using loop
res = False
for ele in test_tup :
    if N == ele :
        res = True
        break
# printing result
print("Does tuple contain required value ? : " + str(res))
```

Output:

```
The original tuple : (10, 4, 5, 6, 8)
Does tuple contain required value ? : True
```

iii. Write python programs to Take a string as ("1234567890") and create a pair {(1,2),(3,4),(5,6),(7,8),(9,0)} using tuple.

Code:

```
value = "1234567890"
pairs = []
for i in range(1, len(value), 2):
    one = value[i - 1]
    two = value[i]
    pairs.append((one, two))
# Display list of tuple pairs.
for pair in pairs:
    print(pair)
```

Output:

```
('1', '2')
('3', '4')
('5', '6')
('7', '8')
('9', '0')
```

Experiment 6:

i. **Write python programs to perform Dictionary functions & Methods: cmp, len, clear(), get(), has_key(), items(), keys(), update(), values() .**

Code:

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
car.clear()
print(car)
dic = {1:'JBIET',2:'CSE', 3:'Students'}
print('original: ', dic)
#print(dic.has_key('CSE'))
#print(dic.has_key('CSE'))
# Accessing value for key
print(dic.get(1))
# Accessing keys for the dictionary
print(dic.keys())
# Accessing keys for the dictionary
print(dic.values())
# Printing all the items of the Dictionary
print(dic.items())
#Update
dic2 = {1:'JBREC', 2:'CSIT', 3:'FACULTY'}
# Dictionary before Updation
print("Original Dictionary:")
print(dic)
# update the value of key 'B'
dic.update(dic2)
print("Dictionary after updation:")
print(dic)
```

Output:

```
{}
original:  {1: 'JBIET', 2: 'CSE', 3: 'Students'}
JBIET
dict_keys([1, 2, 3])
dict_values(['JBIET', 'CSE', 'Students'])
dict_items([(1, 'JBIET'), (2, 'CSE'), (3, 'Students')])
Original Dictionary:
{1: 'JBIET', 2: 'CSE', 3: 'Students'}
Dictionary after updation:
{1: 'JBREC', 2: 'CSIT', 3: 'FACULTY'}
```

ii. Write python programs to Create a list of animal using dictionary variable "animal" and find out if the specific animal present in the list or not?

Code:

```
animals = {}
animals["monkey"] = 1
animals["tuna"] = 2
animals["giraffe"] = 4
print(animals);
# Use in.
if "tuna" in animals:
    print("tuna is present.")
else:
    print("No tuna")
if "elephant" in animals:
    print("elephant is present.")
else:
    print("No elephant")
```

Output:

```
{'monkey': 1, 'tuna': 2, 'giraffe': 4}
tuna is present.
No elephant
```

Experiment 7:

i. **Write a python program to create a class, its objects and accessing attributes.**

Code:

```
class Student:
    def __init__(self, name, roll):
        self.name = name
        self.roll = roll
    def showStudent(self):
        print ("Name is : ", self.name)
        print("Roll No : ", self.roll)
s1 = Student(input("Enter Name :"),input("Enter Roll No :"))
s2 = Student(input("Enter Name :"),input("Enter Roll No :"))
s1.showStudent()
s2.showStudent()
```

Output:

```
Enter Name :raju
Enter Roll No :678965
Enter Name :ramu
Enter Roll No :65276238
Name is :  raju
Roll No :  678965
Name is :  ramu
Roll No :  65276238
```

ii. **Create a Customer class and check the balance and withdraw and deposit some amount**

Code:

```
# BankAccount class
class Bankaccount:
    def __init__(self,amount,balance):
        self.amount=amount
        self.balance=balance



    def deposit(self):
        amount = float(input("Enter amount to be deposited: "))
        self.balance += amount
        print("\n Amount Deposited:", amount)

    # Function to withdraw the amount
    def withdraw(self):
        amount = float(input("Enter amount to be withdrawn: "))
        if self.balance >= amount:
            self.balance -= amount
            print("\n You Withdrew:", amount)
        else:
            print("\n Insufficient balance  ")

    # Function to display the amount
    def display(self):
        print("\n Net Available Balance =", self.balance)
# Driver code

# creating an object of class
s = Bankaccount(400,500)
```

```
# Calling functions with that class object
s.deposit()
s.withdraw()
s.display()
```

Output:

```
Enter amount to be deposited: 1000

 Amount Deposited: 1000.0
Enter amount to be withdrawn: 500

 You Withdrew: 500.0

 Net Available Balance = 1000.0
```

Experiment 8: Write a python script to implement exception handling.

i. **Check whether the input no is integer or not.**

Code:

```
def check_user_input(input):
    try:
        # Convert it into integer
        val = int(input)
        print("Input is an integer number. Number = ", val)
    except ValueError:
        try:
            # Convert it into float
            val = float(input)
            print("Input is a float  number. Number = ", val)
        except ValueError:
            print("No.. input is not a number. It's a string")
input1 = input("Enter your Age ")
check_user_input(input1)
input2 = input("Enter any number ")
check_user_input(input2)
input2 = input("Enter the last number ")
check_user_input(input2)
```

Output:

```
Enter your Age 20
Input is an integer number. Number =  20
Enter any number 45
Input is an integer number. Number =  45
Enter the last number 5
Input is an integer number. Number =  5
```

ii. Handel the exceptions that are come at the time of division.

Code:

```
while True:
    try:
        num = int(input("Your Dividend: "))
        x = int(input("Your Divisor : "))
        div = num / x
    except ValueError:
        print("You should have given an int value ")
    except ZeroDivisionError:
        print("Infinity")
    else:
        print("Result: ",div);
        break
    finally:
        print("finally exception handeled..")
```

Output:

```
Your Dividend: 4
Your Divisor : 2
Result:  2.0
finally exception handeled..
>>>
```

Experiment 9: Write a python script to perform inheritance.

Code:

```
class Person(object):
    # Constructor
    def __init__(self, name):
        self.name = name
    # To get name
    def getName(self):
        return self.name
    # To check if this person is an employee
    def isEmployee(self):
        return False
# Inherited or Subclass (Note Person in bracket)
class Employee(Person):
    # Here we return true
    def isEmployee(self):
        return True
# Driver code
emp = Person("JBIET")  # An Object of Person
print(emp.getName(), emp.isEmployee())
emp = Employee("CSE") # An Object of Employee
print(emp.getName(), emp.isEmployee())
```

Output:

```
JBIET False
CSE True
```

Experiment 10: Write a python script to perform various FILE handling operations. Open, close, read, write, copy.

Code:

Python script to create a File welcome.txt

Welcome.txt

1. Implement Basic input /output operations with various Data Types supported by python.
2. Develop functions for code reusability and experiment string manipulation operations with the use of inbuilt functions.
3. Create a python program for experimenting list, tuple and dictionary
4. Demonstrate Class and objects to make use of object oriented programming concepts.
5. Implement File handling operations to access the contents of file

```
#Open a file
fo = open("D:\Rajkumar\Python lab programs\welcome.txt", "wb")
print("Name of the file: ", fo.name)
# Close opened file
fo.close()
```

Output:

```
Name of the file:  D:\Rajkumar\Python lab programs\welcome.txt
```

Code:

```
file1 = open("D:\Rajkumar\Python lab programs\welcome.txt","w")
L = ["This is Delhi \n","This is Paris \n","This is London \n"]
# \n is placed to indicate EOL (End of Line)
file1.write("Hello \n")
file1.writelines(L)
file1.close() #to change file access modes
file1 = open("D:\Rajkumar\Python lab programs\welcome.txt","r+")
print("Output of Read function is ")
print(file1.read())
print()
# seek(n) takes the file handle to the nth
# bite from the beginning.
file1.seek(0)
```

```
print( "Output of Readline function is ")
print(file1.readline())
print()
file1.seek(0)
# To show difference between read and readline
print("Output of Read(9) function is ")
print(file1.read(9))
print()
file1.seek(0)
print("Output of Readline(9) function is ")
print(file1.readline(9))
file1.seek(0)
# readlines function
print("Output of Readlines function is ")
print(file1.readlines())
print()
file1.close()
```

Output:

```
Output of Read function is
Hello
This is Delhi
This is Paris
This is London

Output of Readline function is
Hello
Output of Read(9) function is
Hello
Th
Output of Readline(9) function is
Hello
Output of Readlines function is
```

```
['Hello \n', 'This is Delhi \n', 'This is Paris \n', 'This is London \n']
```

Code:

```
# Open a file
fo = open("D:\Rajkumar\Python lab programs\welcome.txt", "r+")
str = fo.read(10);
print("Read String is : ", str)
# Close opend file
fo.close()
```

Output:

```
Read String is :  Hello
Thi
```

Code:

```
fo = open("D:\Rajkumar\Python lab programs\welcome.txt", "r+")
str = fo.read(10);
print("Read String is : ", str)
# Check current position
position = fo.tell();
print("Current file position : ", position)
# Reposition pointer at the beginning once again
position = fo.seek(0, 0);
str = fo.read(10);
print("Again read String is : ", str)
# Close opend file
fo.close()
```

Output:

```
Read String is :  Hello
Thi
Current file position :  11
Again read String is :  Hello
Thi
```

Experiment 11:

i. Write a python script to connect to the database and perform DDL operations.

The following Python script is used for establishing the connection to Mysql Database.

```
import MySQLdb
# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")
# Fetch a single row using fetchone() method.
data = cursor.fetchone()
print ("Database version : %s " ,% data)
# disconnect from server
db.close()
```

ii. Create table, insert data into table and display the table data.

Create Database table EMPLOYEE:

```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Drop table if it already exist using execute() method.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement
sql = """CREATE TABLE EMPLOYEE (
        FIRST_NAME  CHAR(20) NOT NULL,
        LAST_NAME  CHAR(20),
        AGE INT,
        SEX CHAR(1),
        INCOME FLOAT )"""

cursor.execute(sql)

# disconnect from server
db.close()
```

SQL *INSERT* statement to create a record into EMPLOYEE table

```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
        LAST_NAME, AGE, SEX, INCOME)
        VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()

# disconnect from server
db.close()
```

Above example can be written as follows to create SQL queries dynamically –

```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
sql = "INSERT INTO EMPLOYEE(FIRST_NAME, \
       LAST_NAME, AGE, SEX, INCOME) \
       VALUES ('%s', '%s', '%d', '%c', '%d' )" % \
       ('Mac', 'Mohan', 20, 'M', 2000)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()

# disconnect from server
db.close()
```

Experiment 12: Write a python script to connect to the database and perform various DML and DQL operations.

Code:

The following procedure queries all the records from EMPLOYEE table having salary more than 1000

```
import MySQLdb
# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to INSERT a record into the database.
sql = "SELECT * FROM EMPLOYEE \
       WHERE INCOME > '%d'" % (1000)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Fetch all the rows in a list of lists.
   results = cursor.fetchall()
   for row in results:
      fname = row[0]
      lname = row[1]
      age = row[2]
      sex = row[3]
      income = row[4]
      # Now print fetched result
      print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \
             (fname, lname, age, sex, income )
except:
   print "Error: unable to fecth data"
# disconnect from server
db.close()
```

Update Operation

The following procedure updates all the records having SEX as 'M'. Here, we increase AGE of all the males by one year.

```
import MySQLdb
# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to UPDATE required records
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1
                          WHERE SEX = '%c'" % ('M')
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
# disconnect from server
db.close()
```

Following is the procedure to delete all the records from EMPLOYEE where AGE is more than 20

```
import MySQLdb
# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# Prepare SQL query to DELETE required records
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
# disconnect from server
db.close()
```