**Experiment 1. a,b,c**

(a) Write a program in Python that uses class to store the name and marks of students.

(b) Create two instances of this class and make use of instance method to display marks.

(c) Write code in Python to show the use of the following built-in functions:

 Getattr(), setattr(), hasattr() and delattr()

**Program:-**

class Sub:  
    name=""  
    marks=[]  
    def \_\_init\_\_(self,name,marks):  
        [self.name](http://self.name/)=name  
        self.marks=marks  
    def show(self):  
            print("Name",[self.name](http://self.name/))  
            print("Marks ",self.marks)  
n=input("Enter the name ")  
print("Enter the marks ")  
m=[]  
for x in range(3):

    a=int(input())  
    m.append(a)  
str=Sub(n,m)  
setattr(str,'name','Robin')  
print("Updated name: ", getattr(str,'name'))  
print(hasattr(str,'marks'))  
delattr(str,"name")  
str.show()

**INPUT:**

Enter the name Abhinav   
Enter the marks  
90  
80  
70

**OUTPUT:**  
Updated name:  abhi   
True  
Name  
Marks  [90, 80, 70]

**Result:-** This program has been executed successfully.

**Experiment 2.a**

a).Write a program to define the same function area to compute the area of circle and rectangle depending upon the number of arguments passed.

**Program**:-

# Python program to find Area of a circle

def findArea(a):

    PI = 3.142

    return PI \* (a\*a);

# Driver method

print("Area is %.6f" % findArea(5));

# Python3 code to find area

# and perimeter of rectangle

# Utility function

def areaRectangle(a, b):

    return (a \* b)

# Driver function

a = 5;

b = 6;

print ("Area = ", areaRectangle(a, b))

**output**:-

Area of circle is 78.550000

Area of Rectangle is 30

**Result**:- This program has been executed successfully.

**Experiment 2 b, c**

(b) Write the Python code for method overriding:

      Class Person: It has an instance method display (self)

      Class Student: It also has an instance method display (self)

Create objects of each class and invoke the instance method display.

(c) Write Python code to show the usage of the built-in class attributes like \_\_doc\_\_, etc.

**Program:-**

class Person:

    def \_\_init\_\_(self):

        pass

    def disp(self):

        print("Person")

class Student(Person):

    def \_\_init\_\_(self):

        pass

    def disp(self):

        print("Student")

p=Person();

s=Student();

p.disp();

s.disp();

def my\_function():

    """Demonstrate docstrings and does nothing really."""

return None

print ("Using \_\_doc\_\_:")

print(my\_function.\_\_doc\_\_)

**OUTPUT:-**

Person

Student

Using \_\_doc\_\_:

Demonstrate docstrings and does nothing really.

**Result**:- This program has been executed successfully.

**Experiment 3 a,b,c**

**(a) Write a program in Python for the class Person using \_\_init\_\_(self) , the class constructor.**

**(b) Add the method: \_\_del\_\_(self)  to the above definition. (Destructor)**

**(c) Add a class attribute- counter and object attributes-** [**self.name**](http://self.name/#sent/_blank) **and self.age. Increase counter by 1 in the constructor and initialize name and age in constructor.**

**Program:-**

class Person:

    name=""

    age=0

    def \_init\_(self,name,age):

        name=[self.name](http://self.name/#sent/_blank)

        age=self.age

    def s(self):

        counter=0

[self.name](http://self.name/#sent/_blank)=name

        self.age=age

        counter=counter+1

        print("constructor called")

    def \_del\_(self):

       print("destructor called")

name="Ram"

age=23

obj=Person()

obj.s()

**OUTPUT:**-

Constructor Called

Destructor Called

**Result**:- This program has been executed successfully.

**Experiment - 4 a**

(a) Write a code in Python to use single inheritance:

**Program:-**

class person:

    name= " "

    age=0

    def \_\_init\_\_(self,name,age):

[self.name](http://self.name/#sent/_blank)=name

        self.age=age

class student(person):

    mark=0

    def \_\_init\_\_(self,name,age,mark):

        person.\_\_init\_\_(self,name,age)

        self.mark=mark

    def s1(self):

        print("name", [self.name](http://self.name/#sent/_blank))

        print("age",self.age)

        print("mark",self.mark)

name=input("enter name")

age=int(input("enter age"))

mark=int(input("enter mark"))

obj=student(name,age,mark)

obj.s1()

**INPUT:-**

Abhinav

20

99

**OUTPUT:-**

Name Abhinav

Age 20

Mark 99

**Experiment - 4 b**

(b) Write Python code using multiple-inheritance.

**#Program:-**

class Person:

    def \_\_init\_\_(self):

        pass

    def s1(self):

        print("Parent")

class Child:

    def \_\_init\_\_(self):

        pass

    def s1(self):

        print("Child")

class GChild(Child,Person):

    def \_\_init\_\_(self):

        pass

    def s3(self):

        print("GChild")

obj=GChild()

obj.s1()

**OUTPUT:-**

Child

**Experiment 4 c**

(c) Write Python code using multi-level inheritance.

**Program:-**

class Person:

    def \_\_init\_\_(self):

        pass

    def s1(self):

        print("Parent")

class Child(Person):

    def \_\_init\_\_(self):

        pass

    def s2(self):

        print("Child")

class GChild(Child):

    def \_\_init\_\_(self):

        pass

    def s3(self):

        print("GChild")

obj=GChild()

obj.s1()

**OUTPUT:-**

Parent

**Result**:- This program has been executed successfully.

**Experiment 5.a,b,c**

a) Write a Python code for database connectivity – database = student.db

b) Create a database table – marks with columns name and marks

c) Insert two records in the table.

**Program**:-

import sqlite3

conn=sqlite3.connect('student1.db')

cur=conn.cursor()

cur.execute(‘’’create table marks(name varchar(20),m1 float,m2 float);’’’)

cur.execute(’’’insert into marks values(\'Seeta\',95.0,97.8);’’’)

cur.execute(‘’’insert into marks values(\'Ashish\',85.0,98.9);’’’)

data1=cur.execute(’’’select \* from marks where name=\'Seeta\';’’’)

for t1 in data1:

print('Name=‘+t1[0])

print(‘Marks1='t1[1])

print(‘Marks2=',t1[2])

cur.close()

**OUTPUT**:-

Name= Seeta

Marks1= 95.0

Marks2= 97.8

**Result**:- This program has been executed successfully.

**Experiment 6.a,b,c**

1. Write a Python code using ‘select’ statement and print the result of query.
2. b) Write Python code using update operation.
3. Write Python code using delete operation.

Program:-

# Database connectivity

import sqlite3

conn= sqlite3.connect('ABC1.db')

cur=conn.cursor()

cur.execute("CREATE TABLE TEST1 (NAME VARCHAR(6), MARKS INT);")

cur.execute("INSERT INTO TEST1 VALUES ('Himani', 98);")

cur.execute("INSERT INTO TEST1 VALUES ('Ramesh',91);")

cur.execute("UPDATE TEST1 SET MARKS= 95 WHERE NAME='Ramesh';")

cur.execute("DELETE FROM TEST1 WHERE NAME='Himani';")

conn.commit()

cur.execute("SELECT \* FROM TEST1 WHERE MARKS >= :VALUE;",{'VALUE':90})

for row in cur:

print(row)

conn.close()

**OUTPUT**:-

('Ramesh', 95)

**Result**:- This program has been executed successfully.

**EXPERIMENT NO 7 (a) (b) (c)**

7. Write Python code using database connectivity:

a) Performing transactions

b) commit operation

c) Roll back operation

**Program:**-

# Database connectivity – using commit and rollback

import sqlite3

conn= sqlite3.connect('ABC1.db')

cur=conn.cursor()

#cur.execute("CREATE TABLE TEST2 (NAME VARCHAR(6), MARKS INT);")

cur.execute("INSERT INTO TEST2 VALUES ('Himani', 98);")

cur.execute("INSERT INTO TEST2 VALUES ('Ramesh',91);")

cur.execute("UPDATE TEST2 SET MARKS= 95 WHERE NAME='Ramesh';")

conn.commit()

cur.execute("UPDATE TEST2 SET MARKS= 99 WHERE NAME='Himani';")

conn.rollback()

cur.execute("SELECT \* FROM TEST2 WHERE MARKS >= :VALUE;",{'VALUE':70})

for row in cur:

print(row)

conn.close()

**OUTPUT**:-

('Himani', 98)

('Ramesh', 95)

**Result**:- This program has been executed successfully.

**EXPERIMENT NO 8**

1. Write Python code using sockets and sockets module.
2. Connect to google socket
3. Use server socket methods
4. Use client socket methods

**Program**:- **For (a)**

import socket

import sys

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

print("Socket successfully created")

except socket.error as err:

print("socket creation failed with error %s")%(err)

port = 80

try:

host\_ip = socket.gethostbyname('www.google.com')

except socket.gaierror:

print("there was an error resolving the host")

sys.exit()

s.connect((host\_ip, port))

print("the socket has successfully connected to google \

on port == %s" %(host\_ip))

**OUTPUT**:-

Socket successfully created

the socket has successfully connected to google on port == 74.125.196.99

**Result**:- This program has been executed successfully.

**Program:**- **For (b)**

#!/usr/bin/python

import socket

s = socket.socket()

host = socket.gethostname()

port = 12345

s.bind((host, port))

s.listen(5)

while True:

c, addr = s.accept()

print ('Got connection from', addr)

c.send(b'Thank you for connecting')

c.close()

**OUTPUT**:-

Server:

Got connection from ('10.10.38.187', 51667)

**Result**:- This program has been executed successfully.

**Program**:- **For (c)**

#!/usr/bin/python

import socket

s = socket.socket()

host = socket.gethostname()

port = 12345

s.connect((host, port))

print (s.recv(1024))

s.close()

**OUTPUT**:-

Client:

b'Thank you for connecting‘

**Result**:- This program has been executed successfully.

**Experiment-9**

AIM:-Write Python code using general socket methods.

b) Implement simple server

c) Implement simple client

**PROGRAM**:-

import socket

import sys

try:

    s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

    print "Socket successfully created"

except socket.error as err:

    print "socket creation failed with error %s" %(err)

# default port for socket

port = 80

try:

    host\_ip = socket.gethostbyname('www.google.com')

except socket.gaierror:

    # this means could not resolve the host

    print "there was an error resolving the host"

    sys.exit()

# connecting to the server

s.connect((host\_ip, port))

print "the socket has successfully connected to google \

on port == %s" %(host\_ip)

**output:**

**Socket successfully created**

Socket successfully created to google on port 80.

**RESULT** :- Thus the program has been successfully executed.

**Experiment-10**

AIM:- Write Python code using Python Internet modules

b) Use socket modules

c) Use socket libraries

**PROGRAM**:-

import socket

s = socket.socket()

host = socket.gethostname()

port = 12345

s.bind((host, port))

s.listen(5)

while True:

c, addr = s.accept()

print 'Got connection from', addr

c.send('Thank you for connecting')

c.close()

10B)

!/usr/bin/python

import socket

s = socket.socket()

host = socket.gethostname()

port = 12345

s.connect((host, port))

print s.recv(1024)

s.close()

Now run this server.py in background and then run above client.py to see the result.

# Following would start a server in background.

$ python server.py &

# Once server is started run client as follows:

$ python client.py

**OUTPUT:**

**Got connection from ('127.0.0.1', 48437)**

**Thank you for connecting**

**RESULT:-**Thus the program has been successfully Executed.