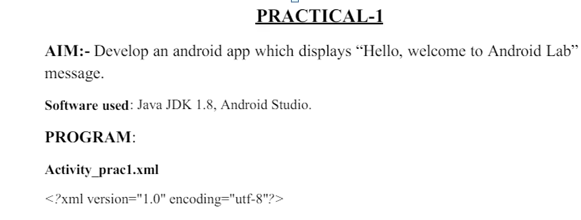
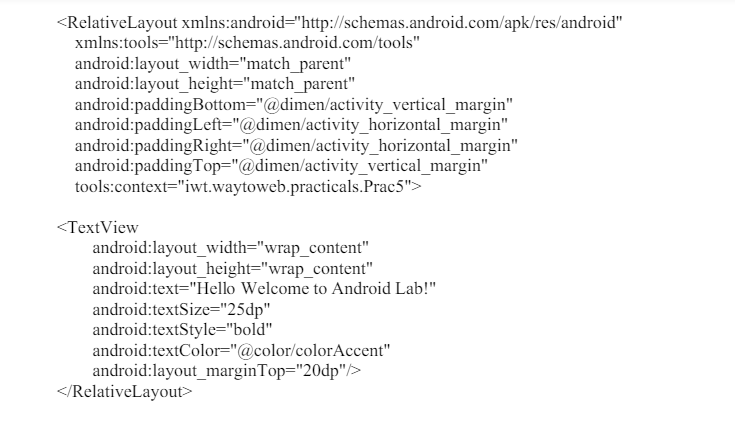
**MCA – 405**

**Elective Lab**

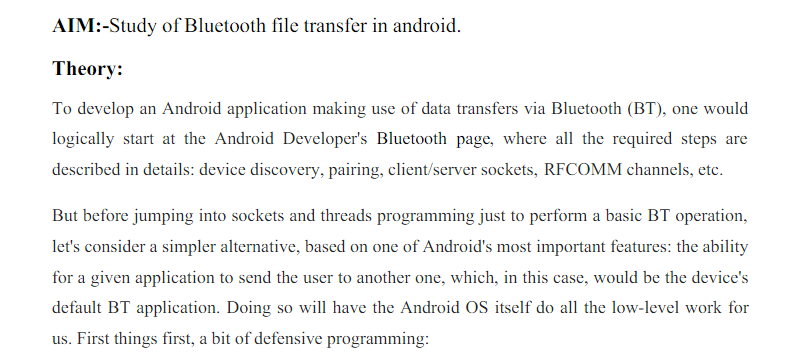
1. **Cloud Computing & Technologies**
2. Installation and configuration of Hadoop/Euceliptus etc.
3. Service deployment & Usage over cloud.
4. Management of cloud resources.
5. Using existing cloud characteristics & Service models.
6. Cloud Security Management.
7. **Mobile Computing**

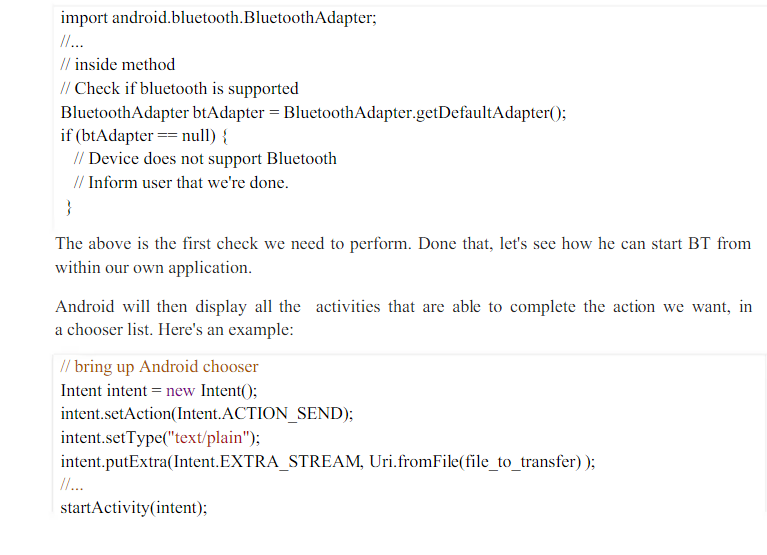


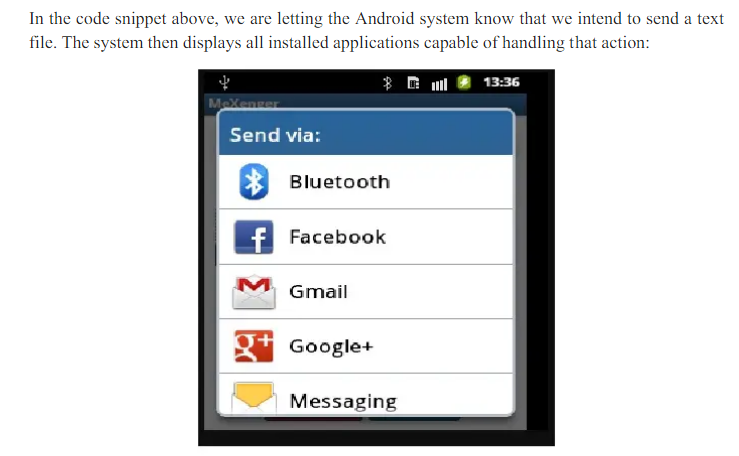


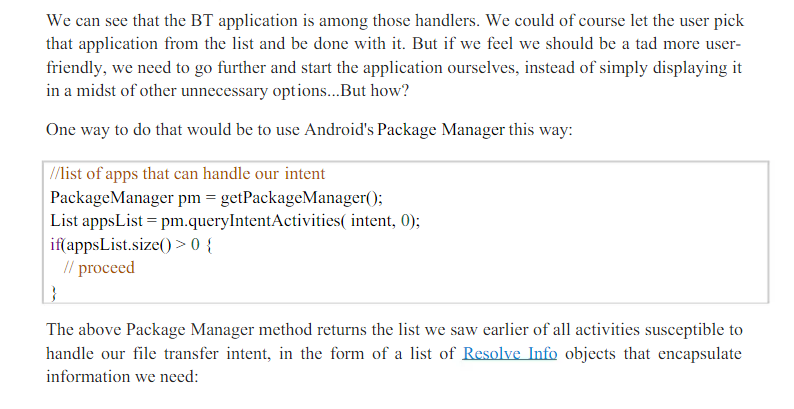


**PRACTICAL – 2**

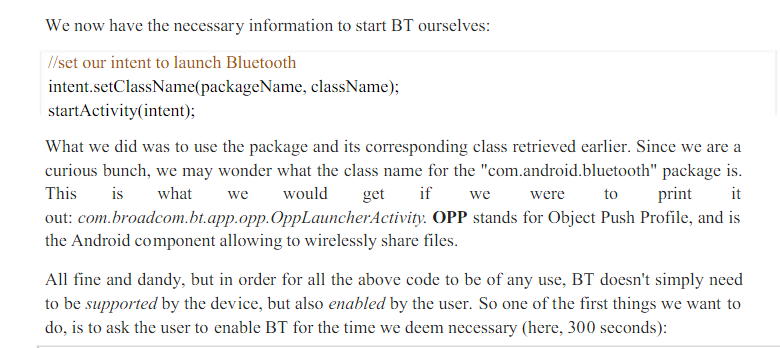
****

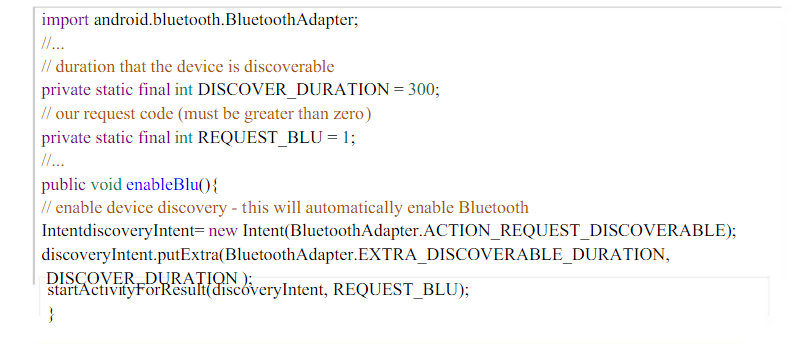
****

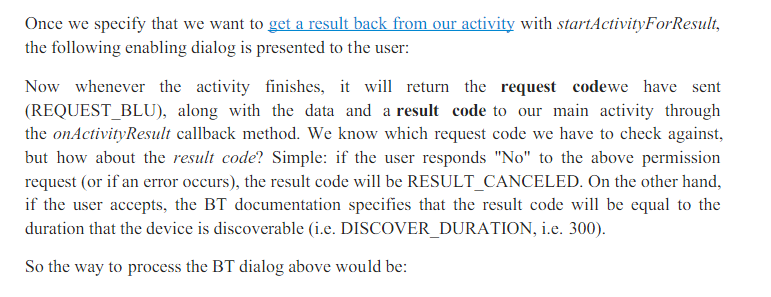
****

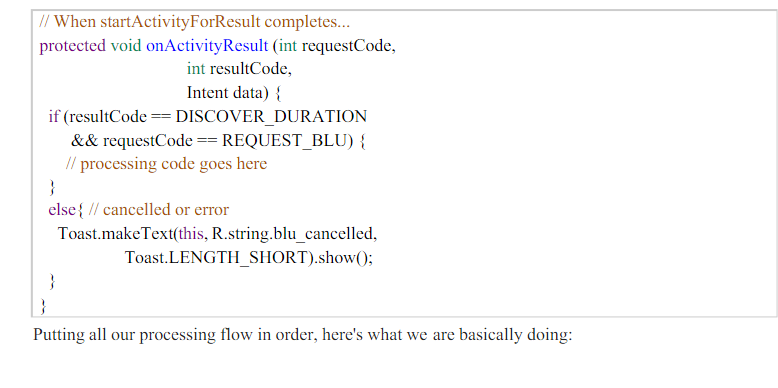
****

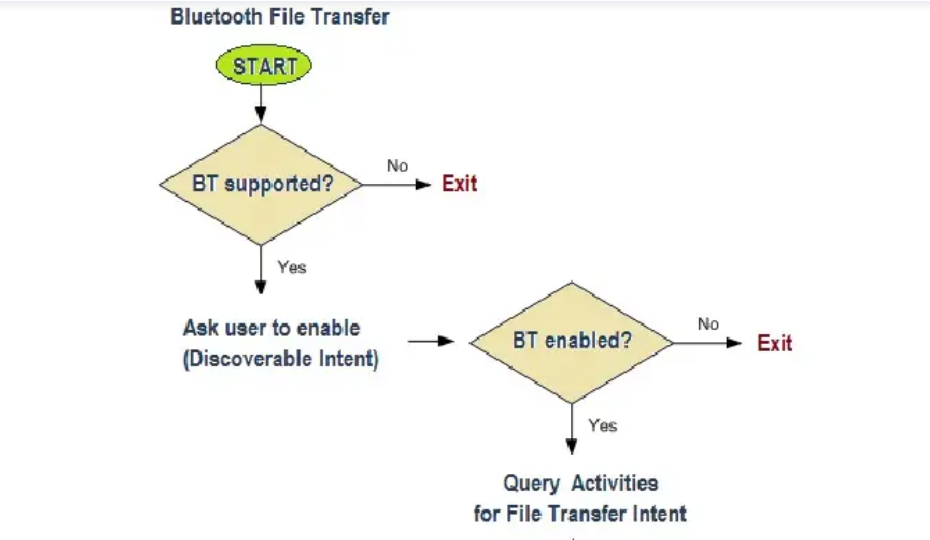
****

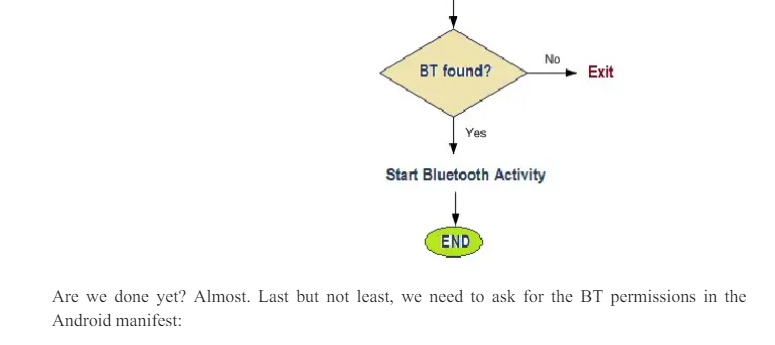
****

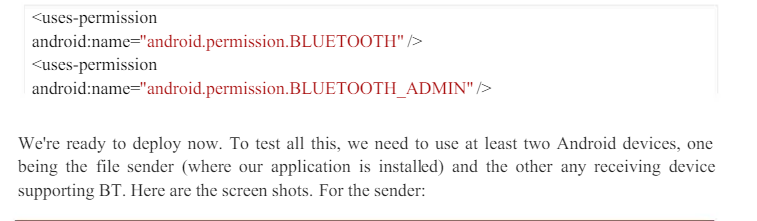
****

****

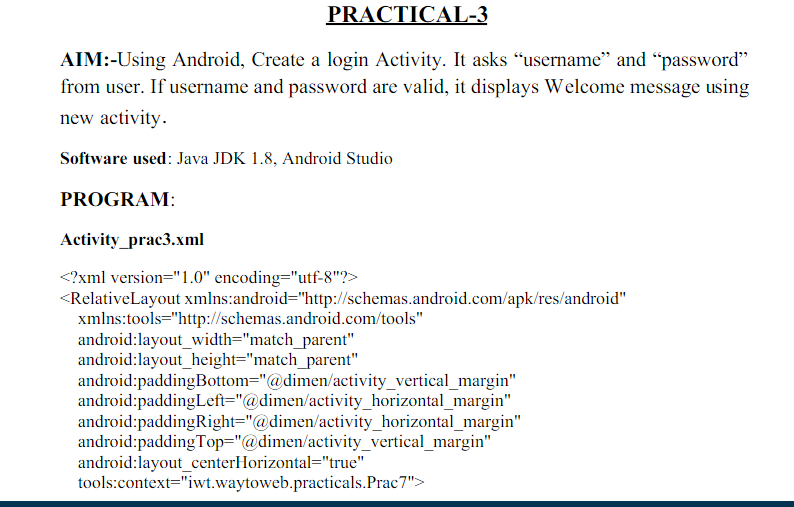
****

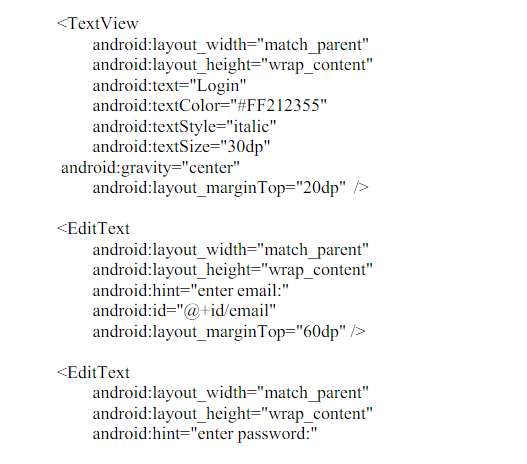
****

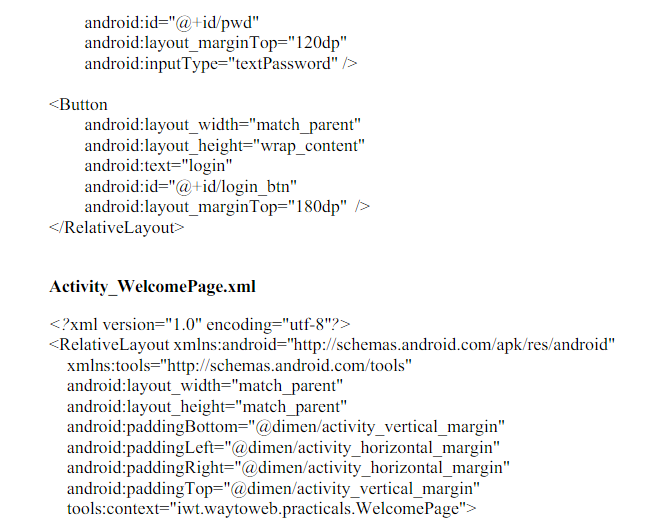
****

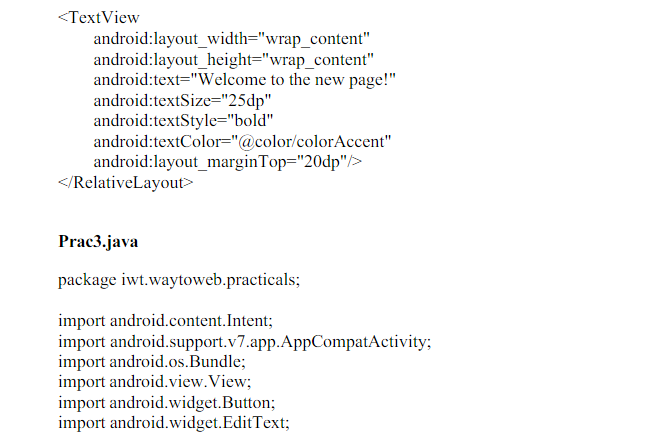
****

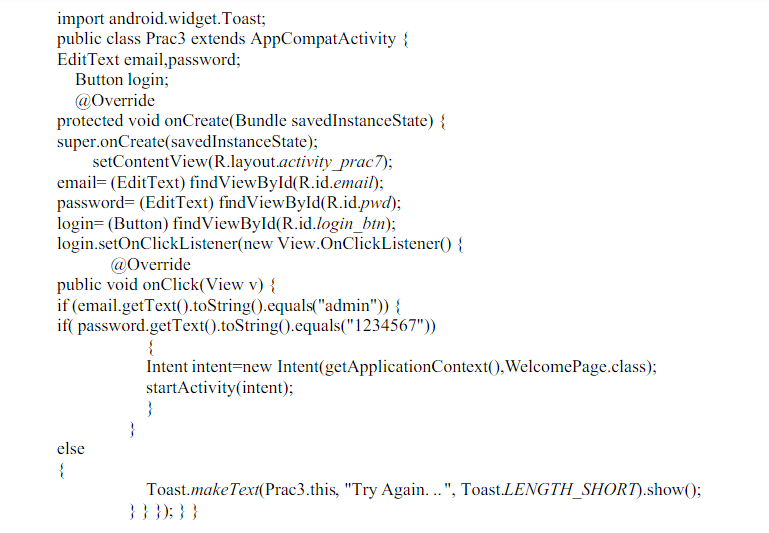
****

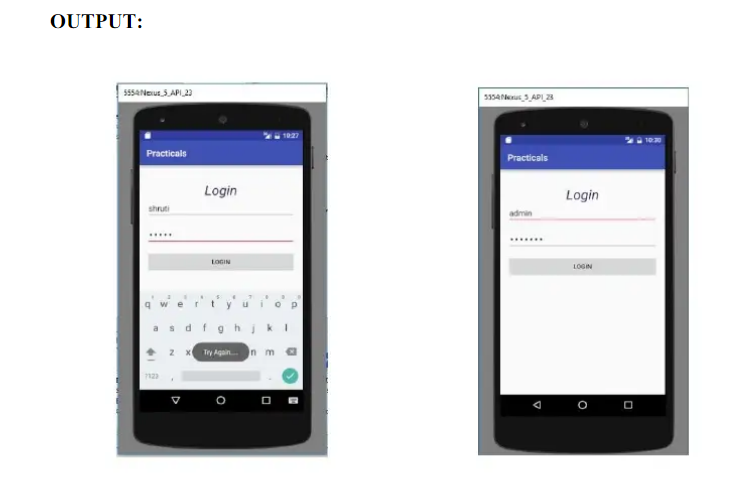


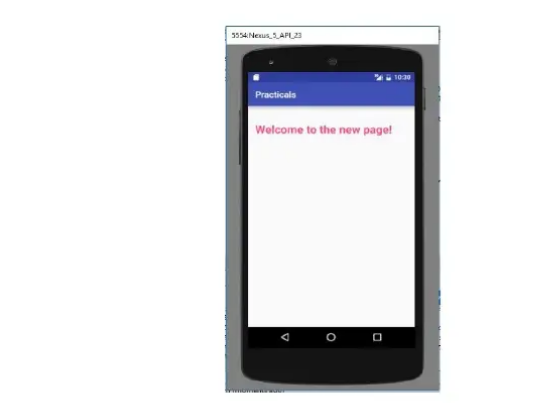




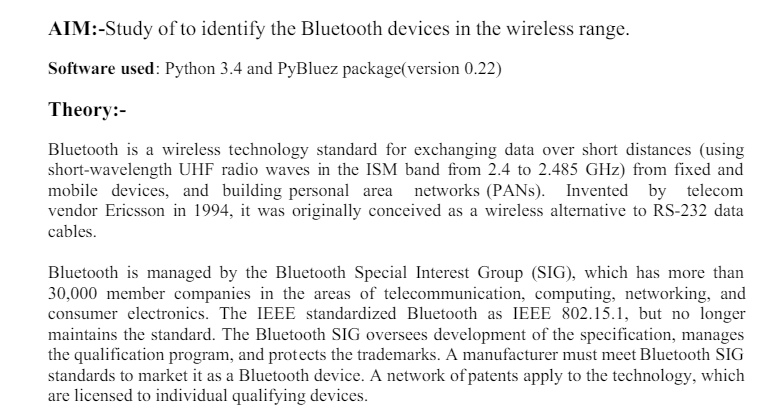


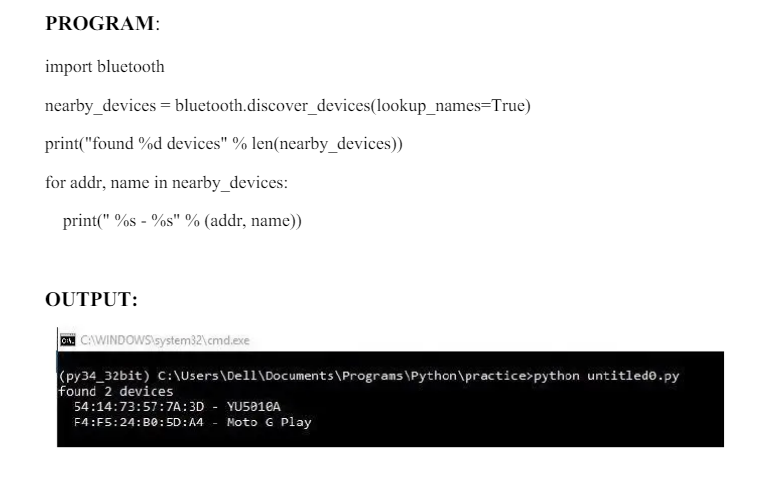
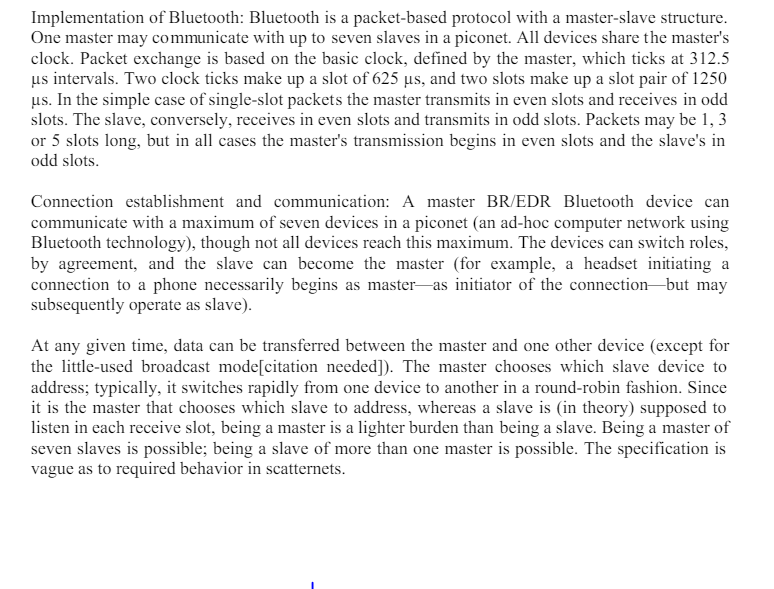






**PRACTICAL - 04**





**FINAL PRACTICAL QUESTIONS OF Advance PYTHON**

**Q. 1. PROGRAMS IN OOPS CONCEPTS**

1. **Create a Constructor in Python**

**class** Student:

# constructor

# initialize instance variable

**def** \_\_init\_\_(self, name):

**print**('Inside Constructor')

self.name = name

**print**('All variables initialized')

# instance Method

**def** show(self):

**print**('Hello, my name is', self.name)

# create object using constructor

s1 = Student('Emma')

s1.show()

**Output:**

Constructor called.

Hello, my name is Emma

1. **Implementing Encapsulation in Python**

**class** Employee:

*# constructor*

**def** \_\_init\_\_(self, name, id, salary, project):

*# data members*

self.name = name

self.id = id

self.salary = salary

self.project = project

*# method to print employee's details*

**def**show\_sal(self):

*# accessing public data member*

**print**("Name: ", self.name, 'Salary:', self.salary)

**def**proj(self):

**print**(self.name, 'is working on', self.project)

*# creating object of a class*

emp = Employee('James', 102, 100000, 'Python')

*# calling public method of the class*

emp.show\_sal()

emp.proj()

**Output:**

Name:James Salary:100000

James is Working on Python

1. **Python example to show the working of multiple inheritance**

class Base1(object):

    def \_\_init\_\_(self):

        self.str1 = "Geek1"

        print("Base1")

class Base2(object):

    def \_\_init\_\_(self):

        self.str2 = "Geek2"

        print("Base2")

 class Derived(Base1, Base2):

    def \_\_init\_\_(self):

 # Calling constructors of Base1

        # and Base2 classes

        Base1.\_\_init\_\_(self)

        Base2.\_\_init\_\_(self)

        print("Derived")

    defprintStrs(self):

        print(self.str1, self.str2)

ob = Derived()

ob.printStrs()

**Output:**

Base1

Base2

Derived

Geek1 Geek2

1. **polymorphism in Python using inheritance and method overriding**

class Animal:

    def speak(self):

        raise NotImplementedError("Subclass must implement this method")

class Dog(Animal):

    def speak(self):

        return "Woof!"

class Cat(Animal):

    def speak(self):

        return "Meow!"

# Create a list of Animal objects

animals = [Dog(), Cat()]

# Call the speak method on each object

for animal in animals:

    print(animal.speak())

**Output:**

Woof!

Meow!

**Q. 2. Write a program in file handling read and write data in text file.**

# Program to show various ways to read and

# write data in a text file.

file = open("myfile.txt","w")

L = ["This is Lagos \n","This is Python \n","This is Fcc \n"]

#i assigned ["This is Lagos \n","This is Python \n","This is Fcc \n"]

#to variable L

#The \n is placed to indicate End of Line

file.write("Hello There \n")

file.writelines(L)

file.close()

# use the close() to change file access modes

file = open("myfile.txt","r+")

print("Output of the Read function is ")

print(file.read())

print()

# The seek(n) takes the file handle to the nth

# byte from the start.

file.seek(0)

print( "The output of the Readline function is ")

print(file.readline())

print()

file.seek(0)

# To show difference between read and readline

print("Output of Read(12) function is ")

print(file.read(12))

print()

file.seek(0)

print("Output of Readline(8) function is ")

print(file.readline(8))

file.seek(0)

# readlines function

print("Output of Readlines function is ")

print(file.readlines())

print()

file.close()

**Output:**

bash

Output of the read function:

Hello There

This is Lagos

This is Python

This is FCC

Output of the readline function:

Hello There

Output of read(12) function:

Hello There

Output of readline(8) function:

Hello Th

Output of readlines function:

['Hello There\n', 'This is Lagos\n', 'This is Python\n', 'This is FCC\n']

**Q. 3. EXCEPTION HANDLING**

**A) Program to handle multiple errors with one**

# except statement

# Python 3

def fun(a):

    if a < 4:

        # throws ZeroDivisionError for a = 3

        b = a/(a-3)

    # throws NameError if a >= 4

    print("Value of b = ", b)

try:

    fun(3)

    fun(5)

# note that braces () are necessary here for

# multiple exceptions

except ZeroDivisionError:

    print("ZeroDivisionError Occurred and Handled")

except NameError:

    print("NameError Occurred and Handled")

**Output:**

Value of b = -3.0

ZeroDivisionError Occurred and Handled

**B) Program to demonstrate finally**

# Python program to demonstrate finally

 # No exception Exception raised in try block

try:

    k = 5//0  # raises divide by zero exception.

    print(k)

 # handles zerodivision exception

except ZeroDivisionError:

    print("Can't divide by zero")

 finally:

    # this block is always executed

    # regardless of exception generation.

    print('This is always executed')

**Output:**

Can't divide by zero

This is always executed

**Q. 4. Program to demonstrate SQLite3**

1. **Connecting to SQLite3 database using Python**

|  |
| --- |
| import sqlite3  try:  # Connect to DB and create a cursor  sqliteConnection = sqlite3.connect('sql.db')  cursor = sqliteConnection.cursor()  print('DB Init')  # Write a query and execute it with cursor  query = 'select sqlite\_version();'  cursor.execute(query)  # Fetch and output result  result = cursor.fetchall()  print('SQLite Version is {}'.format(result))  # Close the cursor  cursor.close()  # Handle errors  except sqlite3.Error as error:  print('Error occurred - ', error)  # Close DB Connection irrespective of success  # or failure  finally:  if sqliteConnection:  sqliteConnection.close()  print('SQLite Connection closed')  **output:**  DB Init  SQLite Version is [('3.37.2',)]  SQLite Connection closed |

1. **Inserting Data into SQLite3 table using Python**

|  |
| --- |
| # Import module  import sqlite3  # Connecting to sqlite  conn = sqlite3.connect('geeks2.db')  # Creating a cursor object using the  # cursor() method  cursor = conn.cursor()  # Creating table  table ="""CREATE TABLE STUDENT(NAME VARCHAR(255), CLASS VARCHAR(255),  SECTION VARCHAR(255));"""  cursor.execute(table)  # Queries to INSERT records.  cursor.execute('''INSERT INTO STUDENT VALUES ('Raju', '7th', 'A')''')  cursor.execute('''INSERT INTO STUDENT VALUES ('Shyam', '8th', 'B')''')  cursor.execute('''INSERT INTO STUDENT VALUES ('Baburao', '9th', 'C')''')  # Display data inserted  print("Data Inserted in the table: ")  data=cursor.execute('''SELECT \* FROM STUDENT''')  for row in data:  print(row)  # Commit your changes in the database  conn.commit()  # Closing the connection  conn.close()  **Output:**  **('Raju', '7th', 'A')**  **('Shyam', '8th', 'B')**  **('Baburao', '9th', 'C')** |

### C) Deleting from SQLite3 table using Python

|  |
| --- |
| import sqlite3# Connecting to sqlite# connection objectconnection\_obj = sqlite3.connect('geek.db')# cursor objectcursor\_obj = connection\_obj.cursor()cursor\_obj.execute("SELECT \* FROM GEEK")print(cursor\_obj.fetchall())#delete data'''It will delete all rows fromthe table'''cursor\_obj.execute("DELETE FROM GEEK")print()print("After deleting all rows")cursor\_obj.execute("SELECT \* FROM GEEK")print(cursor\_obj.fetchall())connection\_obj.commit()# Close the connectionconnection\_obj.close() |

### Output:

### []

### After deleting all rows

### []

### Q. 5. Program in demonstrate in Panda

### Program to create Data frame of three series

### import pandas as pd

### # Define series 1

### s1 = pd.Series([1, 3, 4, 5, 6, 2, 9])

### # Define series 2

### s2 = pd.Series([1.1, 3.5, 4.7, 5.8, 2.9, 9.3])

### # Define series 3

### s3 = pd.Series(['a', 'b', 'c', 'd', 'e'])

### # Define Data

### Data ={'first':s1, 'second':s2, 'third':s3}

### # Create DataFrame

### dfseries = pd.DataFrame(Data)

### Output:

### Copy code

### DataFrame of three series:

### first second third

### 0 1 1.1 a

### 1 3 3.5 b

### 2 4 4.7 c

### 3 5 5.8 d

### 4 6 2.9 e

### 5 2 9.3 NaN

### 6 9 NaNNaN

### When Data contains scalar values

|  |
| --- |
| # Program to Create series with scalar values  # Numeric dataData =[1, 3, 4, 5, 6, 2, 9]  # Creating series with default index valuess = pd.Series(Data) # predefined index valuesIndex =['a', 'b', 'c', 'd', 'e', 'f', 'g']  # Creating series with predefined index valuessi = pd.Series(Data, Index) |

### Output:

### sql

### Series with default index values:

### 0 1

### 1 3

### 2 4

### 3 5

### 4 6

### 5 2

### 6 9

### dtype: int64

### Series with predefined index values:

### a 1

### b 3

### c 4

### d 5

### e 6

### f 2

### g 9

### dtype: int64

### Q. 6. How to search the maximum and minimum element in the given array using NumPy

### import numpy as np

### # Creating 5x4 array

### array = np.arange(20).reshape(5, 4)

### print(array)

### print()

### # If no axis mentioned, then it works on the entire array print(np.argmax(array))

### # If axis=1, then it works on each row

### print(np.argmax(array, axis=1))

### # If axis=0, then it works on each column print(np.argmax(array, axis=0))

### Output:

### sql

### Copy code

### Array:

### [[ 0 1 2 3]

### [ 4 5 6 7]

### [ 8 9 10 11]

### [12 13 14 15]

### [16 17 18 19]]

### Index of maximum element in the array (whole):

### 19

### Index of maximum element along each row:

### [3 3 3 3 3]

### Index of maximum element along each column:

### [4 4 4 4]

### Q. 07. How to sort the elements in the given array using Numpy?

### import numpy as np

### array = np.array([ [3, 7, 1], [10, 3, 2], [5, 6, 7] ])

### print(array) print()

### # Sort the whole array print(np.sort(array, axis=None))

### # Sort along each row print(np.sort(array, axis=1))

### # Sort along each column

### print(np.sort(array, axis=0))

### Output:

### lua

### Array:

### [[ 3 7 1]

### [10 3 2]

### [ 5 6 7]]

### Sorted array (whole):

### [ 1 2 3 3 5 6 7 10]

### Sorted array along each row:

### [[ 1 3 7]

### [ 2 3 10]

### [ 5 6 7]]

### Sorted array along each column:

### [[ 3 3 1]

### [ 5 6 2]

### [10 7 7]]