Experiment 08

Write python programs to understand different File Handling operations in Python.
1) Python implement the concept of pickling.
2) Python program to implement mail merge using file concept (Names are in the file names.txt, Body of the mail is in body.txt)
3) To implement create, modify, delete a record of files.
4) To implement functions of directories.

Roll No.	61
Name	V Krishnasubramaniam
Class	D10-A
Subject	Python Lab
LO Mapped	LO1: Understand the structure, syntax, and semantics of the Python language LO4: Gain proficiency in writing File Handling programs.

<u>**Aim**</u>:

Write python programs to understand different File Handling operations in Python.
1) Python implement the concept of pickling.
2) Python program to implement mail merge using file concept (Names are in the file names.txt and Body of the mail is in body.txt)
3) To implement create, modify, delete a record of files.
4) To implement functions of directories.

<u>**Introduction**</u>:

File Handling in Python

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun.

In Python, there is no need for importing external libraries to read and write files. Python provides an inbuilt function for creating, writing, and reading files.

open () function in Python to open a file in read or write mode. As explained above, open () will return a file object. To return a file object we use open() function along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: open(filename, mode). There are three kinds of mode, that Python provides and how files can be opened: "r" - Read - Default value. Opens a file for reading, error if the file does not exist

1. "a" - Append - Opens a file for appending, creates the file if it does not exist
2. "w" - Write - Opens a file for writing, creates the file if it does not exist
3. "x" - Create - Creates the specified file, returns an error if the file exists
4. "t" - Text - Default value. Text mode
5. "b" - Binary - Binary mode

One must keep in mind that the mode argument is not mandatory. If not passed, then Python will assume it to be “r” by default.

<u>Example:</u>
```
file = open('file.txt', 'r')
for each in file:
   print (each)            #printing every line of the file one by one
```

There is more than one way to read a file in Python. If you need to extract a string that contains all characters in the file then we can use file.read().

<u>Example:</u>
```
file = open("file.text", "r")
```

```
print (file.read())
```

For creating a file using write() function and write mode. The close() command terminates all the resources in use and frees the system of this particular program.

Example:
```
file = open('file.txt','w')
file.write("This is the write command")
file.write("It allows us to write in a particular file")
file.close()
```

Pickling

Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network. Later on, this character stream can then be retrieved and de-serialized back to a Python object.

Python pickle module is used for pickling python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can converts the byte stream (generated through pickling) back into python objects by a process called as unpickling.

In real world sceanario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database. It is advisable not to unpickle data received from an untrusted source as they may pose security threat. However, the pickle module has no way of knowing or raise alarm while pickling malicious data.

Only after importing pickle module we can do pickling and unpickling. Importing pickle can be done using the following command: import pickle

We can pickle objects with the following data types: Booleans, Integers, Floats, Complex numbers, (normal and Unicode) Strings, Tuples, Lists, Sets, and Dictionaries that contain picklable objects. All the above can be pickled, but you can also do the same for classes and functions, for example, if they are defined at the top level of a module.

The process of loading a pickled file back into a Python program is called unpickling.

Example:

Program in pickle1.py:
```
import pickle
mylist = ['a', 'b', 'c', 'd']
with open('datafile.txt', 'wb') as fh:
   pickle.dump(mylist, fh)
```

Program in unpickle1.py:

```
import pickle
pickle_off = open ("datafile.txt", "rb")
emp = pickle.load(pickle_off)
print(emp)
```

<u>Output on running unpickle1.py:</u>

```
['a', 'b', 'c', 'd']
```

To pickle a dictionary, we first define a dictionary `db` as follow and save it in a file named `pickle2.py`. To open the file for writing, simply use the open() function. The first argument should be the name of your file. The second argument is 'rb'. The w means that you'll be writing to the file, and b refers to binary mode. This means that the data will be written in the form of byte objects. If you forget the b, a TypeError: must be str, not bytes will be returned.

Once the file is opened for writing, we can use pickle.dump(), which takes two arguments: the object you want to pickle and the file to which the object has to be saved. In this case, the former will be buggers, while the latter will be outfile. Now, a new file named `examplePickle` should have appeared in the same directory

<u>Example:</u>

```
import pickle
def storeData():
    # initializing data to be stored in db
    Omkar = {'key' : 'Omkar', 'name' : 'Omkar Pathak',
    'age' : 21, 'pay' : 40000}
    Jagdish = {'key' : 'Jagdish', 'name' : 'Jagdish Pathak',
    'age' : 50, 'pay' : 50000}
    # database
    db = {}
    db['Omkar'] = Omkar
    db['Jagdish'] = Jagdish
    # Its important to use binary mode
    dbfile = open('examplePickle', 'ab')
    # source, destination
    pickle.dump(db, dbfile)
    dbfile.close()
def loadData():
    # for reading also binary mode is important
    dbfile = open('examplePickle', 'rb')
    db = pickle.load(dbfile)
    for keys in db:
        print(keys, '=>', db[keys])
    dbfile.close()
if __name__ == '__main__':
    storeData()
    loadData()
```

Output:

```
Omkar => {'key': 'Omkar', 'name': 'Omkar Pathak', 'age': 21, 'pay': 40000}
Jagdish => {'key': 'Jagdish', 'name': 'Jagdish Pathak', 'age': 50, 'pay': 50000}
```

Mail Merge

When we want to send the same invitations to many people, the body of the mail does not change. Only the name (and maybe address) needs to be changed. Mail merge is a process of doing this. Instead of writing each mail separately, we have a template for body of the mail and a list of names that we merge together to form all the mails.

Mail merge is a process in which instead of writing separate emails we list all the names we have and merge them to the main mail body.

Steps:

1. Open the file which has all the names.
2. Inside it opens the file which has the content of the email.
3. Copy the mail content in a variable.
4. Now create a for loop which iterates through the names, and inside the for loop write a statement that creates a new mail file for each person.

Python Program to merge mail:

```
with open("names_file.txt",'r') as names:
    with open("content.txt",'r') as mail: # open the mail content file
        #mail content
        content = mail.read()
        for name in names: # loop over the names
            new_mail = "Hi"+name+content
            # this will create a mail file for each individual name
            with open(name.strip()+".txt",'w') as individual_mail:
                individual_mail.write(new_mail)  #write a new mail for individual
```

Here first we open the file name_file.txt which contains all the names, then we open the file content.txt which contains the mail body or message. Then using the statement content = mail.read()we copy the mail message into a variable content. Then we loop through the names and with each iteration of names we create a new_mail which contain the mail body and person name, and inside the loop itself we create a file with open(name.strip()+".txt",'w') by user name.

Operations on Record of Files:

Create a new file

To create a new file in Python, use the open() method, with one of the following parameters:
"x" - Create: will create a file, returns an error if the file exist
"w" - Write: will create a file if the specified file does not exist

```
f = open("myfile.txt", "x")
        f.close()
```

Creating an empty file and write into it

```
f = open("myfile.txt", "w")
        f.write("Now the file has more content!")
        f.close()
```

Delete a file

To delete a file, you must import the OS module, and run its os.remove() function.
For example, we can remove the file "demofile.txt":

```
import os
os.remove("demofile.txt")
```

Modify a file

Open an existing file and write to it: write() method. To write to an existing file, you must add a parameter to the open() function:

1. "a" - Append: will append to the end of the file
2. "w" - Write: will overwrite any existing content

Example to append at the end of the file:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
```

Example to overwrite the content:

```
f = open("demofile3.txt", "w")
f.write("Whoops! I have deleted the content!")
f.close()
```

Delete a record

To delete a record from a file, we must first read the file from line to line. Look for condition to delete then set that line equal to an empty string, and store the rest of the lines in a variable in given order. Open the file with `w` and overwrite the content of the file with the new lines variable

Example to remove a line from a txt file that starts with 55:

```
with open("in.txt") as f:
    lines = f.readlines()
    for ind, line in enumerate(lines):
        if line.startswith("55"):
            lines[ind] = ""
    with open("in.txt","w") as f:
        f.writelines(lines)
```

Content of in.txt:

```
foo
bar
55 foobar
44 foo
```

Output:

```
foo
bar
44 foo
```

Functions of Directory

The os Python module provides a big range of useful methods to manipulate files and directories. Most of the useful methods are:

1. **os.chdir(path):** Change the current working directory to path
2. **os.chroot(path):** Change the root directory of the current process to path.
3. **os.fchdir(fd):** Change current working directory to the directory represented by the file descriptor fd.
4. **os.getcwd():** Return a string representing the current working directory.
5. **os.getcwdu():** Return a Unicode object representing the current working directory.
6. **os.listdir(path):** Return a list containing the names of the entries in the directory given by path.
7. **os.makedirs(path[, mode]):** Recursive directory creation function.
8. **os.mkdir(path[, mode]):** Create a directory named path with numeric mode mode.
9. **os.removedirs(path):** Remove directories recursively.
10. **os.rename(src, dst):** Rename the file or directory src to dst.
11. **os.renames(old, new):** Recursive directory or file renaming function.
12. **os.rmdir(path):** Remove the directory path
13. **os.walk(top[, topdown=True[, onerror=None[, followlinks=False]]]):** Generate the file names in a directory tree by walking the tree either top-down or bottom-up.

Results:

Pickling:

Program in pickle1.py:

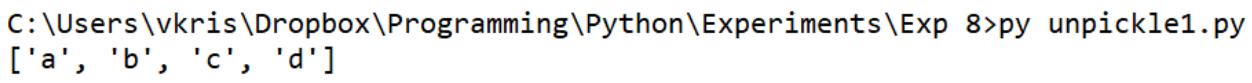
```
import pickle
```

```
mylist = ['a', 'b', 'c', 'd']
with open('datafile.txt', 'wb') as fh:
  pickle.dump(mylist, fh)
```

<u>Program in unpickle1.py:</u>

```
import pickle
pickle_off = open ("datafile.txt", "rb")
emp = pickle.load(pickle_off)
print(emp)
```

<u>Output:</u>

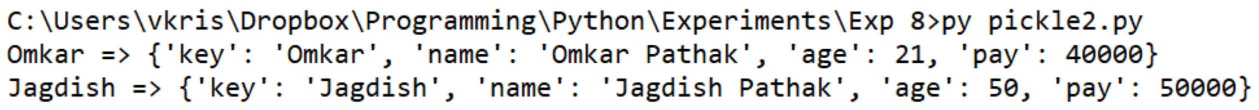


```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py unpickle1.py
['a', 'b', 'c', 'd']
```

<u>Program in pickle2.py:</u>

```
import pickle
def storeData():
   # initializing data to be stored in db
   Omkar = {'key' : 'Omkar', 'name' : 'Omkar Pathak',
   'age' : 21, 'pay' : 40000}
   Jagdish = {'key' : 'Jagdish', 'name' : 'Jagdish Pathak',
   'age' : 50, 'pay' : 50000}
   # database
   db = {}
   db['Omkar'] = Omkar
   db['Jagdish'] = Jagdish
   # Its important to use binary mode
   dbfile = open('examplePickle', 'ab')
   # source, destination
   pickle.dump(db, dbfile)
   dbfile.close()
def loadData():
   # for reading also binary mode is important
   dbfile = open('examplePickle', 'rb')
   db = pickle.load(dbfile)
   for keys in db:
      print(keys, '=>', db[keys])
   dbfile.close()
if __name__ == '__main__':
   storeData()
   loadData()
```

<u>Output:</u>



```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py pickle2.py
Omkar => {'key': 'Omkar', 'name': 'Omkar Pathak', 'age': 21, 'pay': 40000}
Jagdish => {'key': 'Jagdish', 'name': 'Jagdish Pathak', 'age': 50, 'pay': 50000}
```

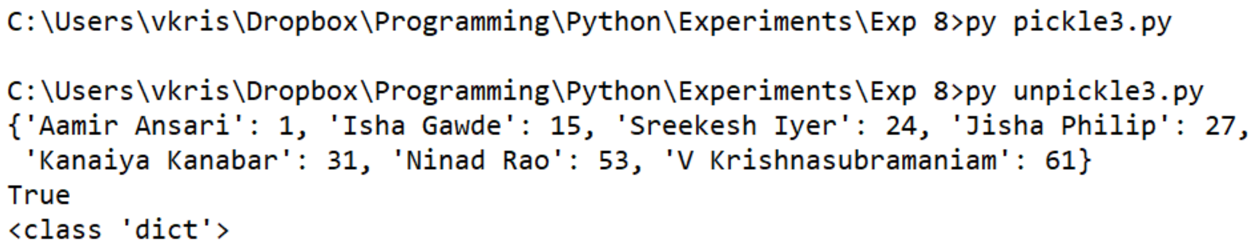
Program in pickle3.py:

```
import pickle
buggers = {"Aamir Ansari":1, "Isha Gawde":15, "Sreekesh Iyer":24, "Jisha Philip":27, "Kanaiya Kanabar":31, "Ninad Rao":53, "V Krishnasubramaniam":61}
filename = "debuggers"
outfile = open(filename, "wb")
pickle.dump(buggers, outfile)
outfile.close()
```

Program in unpickle3.py:

```
import pickle
buggers = {"Aamir Ansari":1, "Isha Gawde":15, "Sreekesh Iyer":24, "Jisha Philip":27, "Kanaiya Kanabar":31, "Ninad Rao":53, "V Krishnasubramaniam":61}
filename = "debuggers"
infile = open(filename,'rb')
new_dict = pickle.load(infile)
infile.close()
print(new_dict)
print(new_dict==buggers)
print(type(new_dict))
```

Output:



```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py pickle3.py

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py unpickle3.py
{'Aamir Ansari': 1, 'Isha Gawde': 15, 'Sreekesh Iyer': 24, 'Jisha Philip': 27,
 'Kanaiya Kanabar': 31, 'Ninad Rao': 53, 'V Krishnasubramaniam': 61}
True
<class 'dict'>
```

Mail Merge:

Program in mailMerge.py:

```
# open names.txt for reading
with open("names.txt", 'r', encoding='utf-8') as names_file:
    # open body.txt for reading
    with open("body.txt", 'r', encoding='utf-8') as body_file:
        # read entire content of the body
        body = body_file.read()
        # iterate over names
```

```
for name in names_file:
   mail = "Hello " + name.strip() + "\n" + body
   # write the mails to individual files
   with open(name.strip()+".txt", 'w', encoding='utf-8') as mail_file:
      mail_file.write(mail)
```

Contents of names.txt:

```
Jovany Barker
Eddie Travis
Ismael Stout
Payten Cantrell
Zaid Shea
Kenny Hebert
Dulce Holder
Mark Koch
Reina Strickland
Yosef Aguirre
Jaidyn Booker
Gideon Pena
```

Contents of body.txt:

When I take you to the Valley, you'll see the blue hills on the left and the blue hills on the right, the rainbow and the vineyards under the rainbow late in the rainy season, and maybe you'll say, "There it is, that's it!" But I'll say. "A little farther." We'll go on, I hope, and you'll see the roofs of the little towns and the hillsides yellow with wild oats, a buzzard soaring and a woman singing by the shadows of a creek in the dry season, and maybe you'll say, "Let's stop here, this is it!" But I'll say, "A little farther yet." We'll go on, and you'll hear the quail calling on the mountain by the springs of the river, and looking back you'll see the river running downward through the wild hills behind, below, and you'll say, "Isn't that the Valley?" And all I will be able to say is "Drink this water of the spring, rest here awhile, we have a long way yet to go and I can't go without you.

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py mailMerge.py
```

Output as contents of Jovany Barker.txt:

Hello Jovany Barker

When I take you to the Valley, you'll see the blue hills on the left and the blue hills on the right, the rainbow and the vineyards under the rainbow late in the rainy season, and maybe you'll say, "There it is, that's it!" But I'll say. "A little farther." We'll go on, I hope, and you'll see the roofs of the little towns and the hillsides yellow with wild oats, a buzzard soaring and a woman singing by the shadows of a creek in the dry season, and maybe you'll say, "Let's stop here, this is it!" But I'll say, "A little farther yet." We'll go on, and you'll hear the quail calling on the mountain by the springs of the river, and looking back you'll see the river running downward through the wild hills behind, below, and you'll say, "Isn't that the Valley?" And all I will be able to say is "Drink this water of the spring, rest here awhile, we have a long way yet to go and I can't go without you.

Output as contents of Eddie Travis.txt:

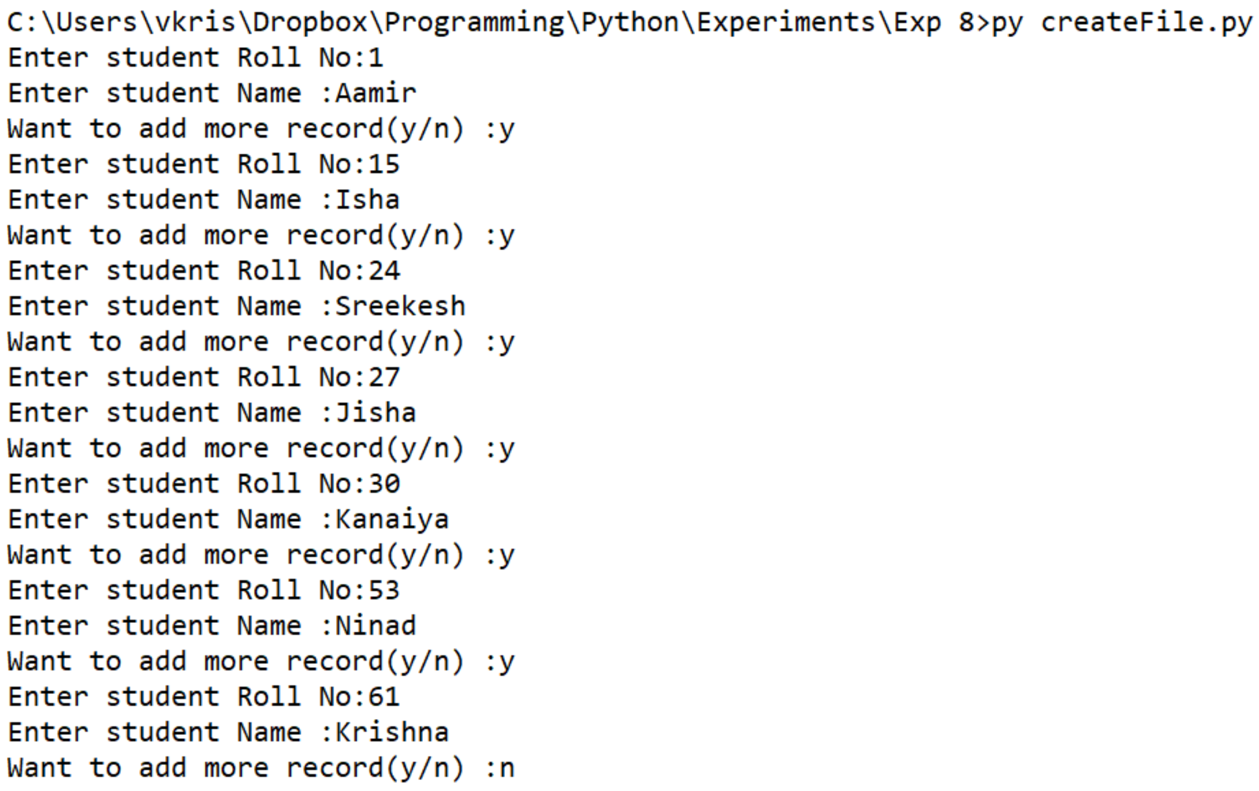
```
Hello Eddie Travis

When I take you to the Valley, you'll see the blue hills on the left and the blue hills on the right, the rainbow and the vineyards under the rainbow late in the rainy season, and maybe you'll say, "There it is, that's it!" But I'll say. "A little farther." We'll go on, I hope, and you'll see the roofs of the little towns and the hillsides yellow with wild oats, a buzzard soaring and a woman singing by the shadows of a creek in the dry season, and maybe you'll say, "Let's stop here, this is it!" But I'll say, "A little farther yet." We'll go on, and you'll hear the quail calling on the mountain by the springs of the river, and looking back you'll see the river running downward through the wild hills behind, below, and you'll say, "Isn't that the Valley?" And all I will be able to say is "Drink this water of the spring, rest here awhile, we have a long way yet to go and I can't go without you.
```

Operations on files and records:

Program in createFile.py:

```
import pickle
list =[]
while True:
    roll = input("Enter student Roll No:")
    sname  = input("Enter student Name :")
    student = {"roll":roll,"name":sname}
    list.append(student)
    choice= input("Want to add more record(y/n) :")
    if(choice=='n'):
        break
file = open("student.dat","wb")
pickle.dump(list,file)
file.close()
```

Output:

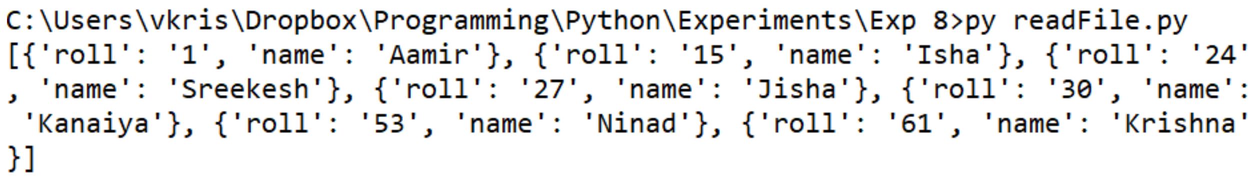


```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py createFile.py
Enter student Roll No:1
Enter student Name :Aamir
Want to add more record(y/n) :y
Enter student Roll No:15
Enter student Name :Isha
Want to add more record(y/n) :y
Enter student Roll No:24
Enter student Name :Sreekesh
Want to add more record(y/n) :y
Enter student Roll No:27
Enter student Name :Jisha
Want to add more record(y/n) :y
Enter student Roll No:30
Enter student Name :Kanaiya
Want to add more record(y/n) :y
Enter student Roll No:53
Enter student Name :Ninad
Want to add more record(y/n) :y
Enter student Roll No:61
Enter student Name :Krishna
Want to add more record(y/n) :n
```

Program in readFile.py:

```
import pickle
file = open("student.dat", "rb")
list = pickle.load(file)
print(list)
file.close()
```

Output:



```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py readFile.py
[{'roll': '1', 'name': 'Aamir'}, {'roll': '15', 'name': 'Isha'}, {'roll': '24'
, 'name': 'Sreekesh'}, {'roll': '27', 'name': 'Jisha'}, {'roll': '30', 'name':
 'Kanaiya'}, {'roll': '53', 'name': 'Ninad'}, {'roll': '61', 'name': 'Krishna'
}]
```

Program in modifyRecord.py:

```
import pickle
name = input('Enter name that you want to update in binary file : ')
file = open("student.dat", "rb+")
list = pickle.load(file)
found = 0
lst = []
for x in list:
    if name in x['name']:
        found = 1
        x['name'] = input('Enter new name: ')
    lst.append(x)
if(found == 1):
    file.seek(0)
    pickle.dump(lst, file)
    print("Record Updated!")
else:
    print('Name does not exist!')
file.close()
```

<u>Output:</u>

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py modifyRecord.py
Enter name that you want to update in binary file : Krishna
Enter new name: Krishnasubramaniam
Record Updated!
```

<u>Program in deleteRecord.py:</u>

```
import pickle
name = input('Enter name that you want to DELETE from binary file: ')
file = open("student.dat", "rb+")
list = pickle.load(file)
found = 0
lst = []
for x in list:
    if name not in x['name']:
        lst.append(x)
    else:
        found = 1
if(found == 1):
    file.seek(0)
    pickle.dump(lst, file)
    print("Record Deleted!")
else:
    print('Name does not exist!')
file.close()
```

<u>Output:</u>

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 8>py deleteRecord.py
Enter name that you want to DELETE from binary file: Krishna
Record Deleted!
```

<u>Directory Functions:</u>

```
>>> import os
>>> os.getcwd()
'C:\\Users\\vkris\\Dropbox\\Programming\\Python\\Experiments'
>>> os.getcwdb()
b'C:\\Users\\vkris\\Dropbox\\Programming\\Python\\Experiments'
>>> os.chdir('C:\\Users\\vkris\\Dropbox\\Programming\\Python\\Experiments\\Exp 8')

>>> os.getcwd()
'C:\\Users\\vkris\\Dropbox\\Programming\\Python\\Experiments\\Exp 8'

>>> os.listdir()
['body.txt', 'createFile.py', 'datafile.txt', 'debuggers', 'deleteRecord.py', 'exa
mplePickle', 'mailMerge.py', 'modifyRecord.py', 'names.txt', 'pickle1.py', 'pickle
2.py', 'pickle3.py', 'readFile.py', 'student.dat', 'unpickle1.py', 'unpickle3.py']

>>> os.mkdir('MyNewDirectory')
>>> os.listdir()
['body.txt', 'createFile.py', 'datafile.txt', 'debuggers', 'deleteRecord.py', 'exa
mplePickle', 'mailMerge.py', 'modifyRecord.py', 'MyNewDirectory', 'names.txt', 'pi
ckle1.py', 'pickle2.py', 'pickle3.py', 'readFile.py', 'student.dat', 'unpickle1.py
', 'unpickle3.py']

>>> os.remove('datafile.txt')
>>> os.listdir()
['body.txt', 'createFile.py', 'debuggers', 'deleteRecord.py', 'examplePickle', 'ma
ilMerge.py', 'modifyRecord.py', 'names.txt', 'NewDirectory', 'pickle1.py', 'pickle
2.py', 'pickle3.py', 'readFile.py', 'student.dat', 'unpickle1.py', 'unpickle3.py']

>>> os.rmdir('NewDirectory')
>>> os.listdir()
['body.txt', 'createFile.py', 'debuggers', 'deleteRecord.py', 'examplePickle', 'ma
ilMerge.py', 'modifyRecord.py', 'names.txt', 'pickle1.py', 'pickle2.py', 'pickle3.
py', 'readFile.py', 'student.dat', 'unpickle1.py', 'unpickle3.py']
```

Conclusion:

Thus, we have understood basics of file handling in Python using concepts of pickling and mail merge. We also learnt about CRUD operations on binary files and built-in directory management functions of Python, by performing hands-on practical programs for each of these concepts.