Experiment 08

Write python programs to understand different File Handling operations in Python.

1)Python implements 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.	01
Name	Aamir Ansari
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

Aim: Write python programs to understand different File Handling operations in Python

Introduction:

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.
The key function for working with files in Python is the open() function. The open() function takes two parameters; filename, and mode.
There are four different methods (modes) for opening a file:

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

Pickling

1. Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening.
2. 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.
3. Later on, this character stream can then be retrieved and de-serialized back to a Python object.
4. Pickling is not to be confused with compression
5. The former is the conversion of an object from one representation (data in Random Access Memory (RAM)) to another (text on disk), while the latter is the process of encoding data with fewer bits, in order to save disk space

We can pickle objects with the following data types:

1. Booleans,
2. Integers,
3. Floats,
4. Complex numbers,
5. (normal and Unicode) Strings,
6. Tuples,
7. Lists,
8. Sets, and
9. 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.

To pickle a dictionary, we first define a dictionary `buggers` as follow and save it in a file named `debuggers.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 'wb'. 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 `debuggers` should have appeared in the same directory

Unpickling files
The process of loading a pickled file back into a Python program is similar to the one you saw previously: use the open() function again, but this time with 'rb' as second argument (instead of wb). The r stands for read mode and the b stands for binary mode. You'll be reading a binary file. Assign this to infile. Next, use pickle.load(), with infile as argument, and assign it to new_dict. The contents of the file are now assigned to this new variable. Again, you'll need to close the file at the end.

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 the body of the mail and a list of names that we merge together to form all the mails.
For this program, we have written all the names in separate lines in the file "names.txt". The body is in the "body.txt" file.
We open both the files in reading mode and iterate over each name using a for loop. A new file with the name "[name].txt" is created, where name is the name of that person.
We use strip() method to clean up leading and trailing whitespaces (reading a line from the file also reads the newline '\n' character). Finally, we write the content of the mail into this file using the write() method.

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()
```

Create 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 through python,
Delete a File: `os.remove()` function
To delete a file, you must import the OS module, and run its os.remove() function.
For example, 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:
"a" - Append: will append to the end of the file
"w" - Write: will overwrite any existing content

Write: append to the end of the file

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

Write: 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

For example removing 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)
```

input:

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

output:

```
foo
bar
44 foo
```

Functions of Directories

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

File: **pickle_debuggers.py**

```
import pickle

buggers = {"Aamir":1, "Isha":15, "Sreekesh":24, "Jisha":27, "Kanaiya":31, "Ninad":53, "Krishna":61}
filename = "debuggers"
outfile = open(filename, "wb")
pickle.dump("buggers", outfile)
outfile.close()
```

File: **unpickle_debuggers.py**

```
import pickle

buggers = {"Aamir":1, "Isha":15, "Sreekesh":24, "Jisha":27, "Kanaiya":31, "Ninad":53, "Krishna":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:

```
E:\Sem-4\Lab_Assignments\Python_Lab\Experiment_08\code>python pickle_debuggers.py

E:\Sem-4\Lab_Assignments\Python_Lab\Experiment_08\code>python unpickle_debuggers.py
{'Aamir': 1, 'Isha': 15, 'Sreekesh': 24, 'Jisha': 27, 'Kanaiya': 31, 'Ninad': 53, 'Krishna': 61}
True
<class 'dict'>
```

Mail merge

File: **names.txt**

```
Aamir
Isha
Sreekesh
Jisha
Kanaiya
Ninad
Krishna
```

File: **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.

File: **mail_merge.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)
```

Example of output file generated `**Aamir.txt**`

Hello Aamir
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.

```
>>> import os
>>> os.chdir("E:\\Sem-4\\Lab_Assignments\\Python_Lab\\Experiment_08")
>>> os.getcwd()
'E:\\Sem-4\\Lab_Assignments\\Python_Lab\\Experiment_08'
>>> f = open("EverythingAtOnce.txt", "w")
>>> f.write("")
0
>>> f.write("As warm as the sun, as silly as fun\nAs cool as a tree, as scare as the sea\n")
75
```

EverythingAtOnce.txt - Notepad

File Edit Format View Help

```
As warm as the sun, as silly as fun
As cool as a tree, as scare as the sea
```

****

```
>>> f = open("EverythingAtOnce.txt", "a")
>>> f.write("As hot as fire, cold as ice\nSweet as sugar and everything nice!")
63
>>> f.close()
```

EverythingAtOnce.txt - Notepad

File Edit Format View Help

```
As warm as the sun, as silly as fun
As cool as a tree, as scare as the sea
As hot as fire, cold as ice
Sweet as sugar and everything nice!
```

****

```
>>> with open("EverythingAtOnce.txt") as f:
        lines = f.readlines()
        for index, line in enumerate(lines):
                if line.startswith("As hot"):
                        lines[index] = ""
                with open("EverythingAtOnce.txt", "w") as f:
                        f.writelines(lines)


>>>
```

EverythingAtOnce.txt - Notepad

File Edit Format View Help

```
As warm as the sun, as silly as fun
As cool as a tree, as scare as the sea
Sweet as sugar and everything nice!
```

```
>>> os.remove("EverythingAtOnce.txt")
>>>
```

Directories:

```
>>> import os
>>> curDir = os.getcwd()
>>> print(curDir)
C:\Users\1104a\AppData\Local\Programs\Python\Python39
>>>
>>> contents = os.listdir()
>>> print(contents)
['DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.
dll', 'python39.dll', 'pythonw.exe', 'Scripts', 'tcl', 'Tools', 'vcruntime140.dll', 'vcrunti
me140_1.dll']
>>>
>>> os.chdir("E:\\Sem-4\\Lab_Assignments\\Python_Lab\\")
>>> os.getcwd()
'E:\\Sem-4\\Lab_Assignments\\Python_Lab'
>>>
>>> os.mkdir("Temp")
>>> os.listdir
<built-in function listdir>
>>>
>>> os.listdir()
['body.txt', 'Experiment_01', 'Experiment_02', 'Experiment_03', 'Experiment_04', 'Experiment
_05', 'Experiment_06', 'Experiment_07', 'Experiment_08', 'names.txt', 'Temp']
>>> os.rename("Temp", "Not_temp_anymore")
>>> os.listdir()
['body.txt', 'Experiment_01', 'Experiment_02', 'Experiment_03', 'Experiment_04', 'Experiment
_05', 'Experiment_06', 'Experiment_07', 'Experiment_08', 'names.txt', 'Not_temp_anymore']
>>> os.rmdir("Not_temp_anymore")
>>> os.listdir()
['body.txt', 'Experiment_01', 'Experiment_02', 'Experiment_03', 'Experiment_04', 'Experiment
_05', 'Experiment_06', 'Experiment_07', 'Experiment_08', 'names.txt']
```

Conclusion:

Hence we have successfully studied and used **File Handling operations** in Python.