--

Unit 4:

Object oriented Programming: classes and objects – Inheritance – Polymorphism, overloading; Error handling & Exceptions – try, except and raise – exception propagation.

File Processing: reading and writing files.

File Processing:

1. What is File?
2. Types of Files.
3. File Handling
4. Why File Handling?
5. Operations we can perform

Files are named locations on disk to store information

Generally, they are used to store data permanently. [named locations means every file has some name]

Data is stored in non-volatile memory. (we can retrieve data whenever required)

Types of Files:

There are two types of files: -

- text files
- binary files

Text files: - stores data in the form of characters. It's used to store data and strings.

E.g. .txt files, .Json files

Binary files: - stores data in the form of bytes (group of 8 bits).

E.g. Audio files, video, image files, pdfs etc.

File Handling:

It refers to the ability to work with files, such as reading from or writing to them. It allows us to open, create, modify and close files in our python programming.

It's a powerful way to interact with and manipulate data stored in files.

- **[Why File Handling????]**

Operations:

There are certain operations that we can perform on files using python.

1. **open ()**

python provides an inbuilt function open () to open a file.

syntax –

f = open (filename, mode = 'r', buffering, encoding = None, errors = None, newline = None)

***Note- except filename other parameters are optional.

commonly used syntax –

f = open (filename, mode = '')

filename: - file to be accessed

mode: - access mode (purpose of opening a file)

f: - file handler or file pointer

(file pointer points to the beginning of the file (cursor))

- The allowed **modes** in Python are: -

1. **'r'** ---------- open an existing file for read operation.

The file pointer is positioned at the beginning of the file. If the specified file does not exist then we will get FileNotFoundError. This is default mode.

1. **'w'** ---------- open an existing file for write operation. If the file already contains some data, then it will be overridden. If the specified file is not already available then this mode will create that file.
2. **'a'** ---------- open an existing file for append operation. It won't override existing data. If the specified file is not already available then this mode will create a new file.
3. **'r+'** ---------- To read and write data into the file. The previous data in the file will not be deleted. The file pointer is placed at the beginning of the file.
4. **'w+'** ---------- To write and read data. It will override existing data.

1. **'a+'** ---------- To append and read data from the file. It won't override existing data.
2. **'x'** ---------- To open a file in exclusive creation mode for write operation. If the file already exists then we will get FileExistsError.

Note: All the above modes are applicable for text files. If the above modes suffixed with 'b' then these represents for binary files.

E.g.: rb, wb, ab, r+b, w+b, a+b, xb

E.g.:

f= open ("abc.txt", "w")

We are opening abc.txt file for writing data.

1. **close ()**

Closing a File:

After completing our operations on the file, it is highly recommended to close the file. For this we have to use close () function.

syntax – file_handler. close ()

- **Various properties of File Object:**

Once we opened a file and we got file object, we can get various details related to that file by using its properties.

name ------- Name of opened file

mode ------- Mode in which the file is opened

closed ------- Returns boolean value indicates that file is closed or not

readable () ------- Returns boolean value indicates that whether file is readable or not

writable () ------- Returns boolean value indicates that whether file is writable or not

E.g.:

f= open ('abc.txt', 'w')

print ('File Name: ', f. name) o/p :- File name: abc.txt

print ('File Mode: ', f. mode) o/p :- File mode: write mode

print ('Is File readable: ', f. readable ())O/p :- 0

print ('Is File writable: ', f. writable ())o/p: 1

print ('Is File closed: ', f. closed)o/p:- 0

f. close ()

print ('Is File closed: ', f. closed) o/p:-

- **Writing data to text files:**

We can write character data to the text files by using the following 2 methods.

- write (str)
- writelines (list of lines)

E.g.:

f= open ('abc.txt', 'w')

f. write ('Hello\n')

f. write ('How are you? \n')

print (‘Data written to the file successfully.’)

f. close ()

***Note:

In the above program, data present in the file will be overridden every time if we run the program. Instead of overriding if we want append operation then we should open the file as follows.

f= open ("abcd.txt", "a")

E.g.:2

f=open (‘abc.txt’, ‘w’)

list1 = [‘sunny\n’, ‘bunny\n’, ‘Vinny\n’, ‘chinny\n’]

f. writelines (list1)

print (“List of lines written to the file successfully.”)

f. close ()

**Note:

while writing data by using write () methods, compulsory we have to provide line separator (\n), otherwise total data will be written to a single line.

- **Reading Character Data from text files:**

We can read character data from text file by using the following read methods.

read () ------- To read total data from the file

read (n) ------- To read 'n' characters from the file

readline () ------- To read only one line

readlines () ------- To read all lines into a list

- **The with statement:**

The with statement can be used while opening a file.

We can use this to group file operation statements within a block. The advantage of with statement is it will take care of closing of file, after completing all operations automatically even in the case of exceptions also, and we are not required to close explicitly.

E.g.:

with open (‘anc.txt’, ‘w’) as f:

f. write (‘Hello!!\n’)

f. write (‘Everyone! \n’)

print (‘Is File closed: ‘, f. closed)

------ Check whether a particular file exists or not?

We can use ‘os’ library to get information about files in our computer. ‘os’ module has a sub module named ‘path’, which contains isFile () function to check whether a particular file exists or not?

syntax – os. path. isfile (fname)

Question1 – Write a program to check whether the given file exists or not. If it is available then print its content.

Question2 – Write a program to print the number of lines, words and characters present in the given file?

- **Handling Binary Files:**

It is very common requirement to read or write binary data like images, video files, audio files etc.

Q. Program to read image file and write to a new image file?

f1 = open (‘naruto.jpg’, ‘rb’)

f2 = open (‘hinata.jpg’, ‘wb’)

```
bytes = f1. read ()

f2. write (bytes)

print (“New Image is available with the name: hinata.jpg")
```

- **Handling csv files:**

CSV==>Comma separated values

As the part of programming, it is very common requirement to write and read data w.r.t csv files. Python provides csv module to handle csv files.

-----Writing data to csv file:

```
import csv

with open ("emp. csv", "w", newline='') as f:

w=csv. writer (f) # returns csv writer object

w. writerow (["ENO","ENAME","ESAL","EADDR"])

n=int (input ("Enter Number of Employees:"))

for i in range(n):

eno=input ("Enter Employee No:")

ename=input ("Enter Employee Name:")

esal=input ("Enter Employee salary:")

eaddr=input ("Enter Employee Address:")

w. writerow ([eno, ename, esal, eaddr])

print ("Total Employees data written to csv file successfully")
```

***Note: Observe the difference with newline attribute and without with open ("emp. csv", "w", newline='') as f: with open ("emp. csv", "w") as f:

***Note: If we are not using newline attribute then in the csv file blank lines will be included between data. To prevent these blank lines, newline attribute is required in Python-3, but in Python-2 just we can specify mode as 'wb' and we are not required to use newline attribute.

----- Reading Data from csv file:

E.g.:

```
import csv

f=open ("emp. csv", 'r')

r=csv. reader (f) #returns csv reader object

data=list(r)

#print (data)

for line in data:

for word in line:

print (word,"\t", end='')

print ()
```