6-4-23

Advance

File handling:

-> It is a phenomenon of performing the functionalities inside the file system (such as creation, deletion, retrieval, updating data and managing).

-> In this concept we are going to work with the type of files called flat files [text files, and binary text file]

-> flat files are the basic file types which support only 2 operations.

1. Inserting the data
2. Retrieving the data.

-> stream is a channel that is created between the python program and file for the data transaction.

to create a stream, we have 1 way:

1. By using open function.

open(filepath_with_extension, mode): This is a function used to create a stream for the communication with a file from

a Python program and creating the file pointer and opening the file. after that it returns the file pointer address.

Syntax : fopen = open (filePath_with_extension, mode)

example :

fopen = open ('c:/users/desktop/tfile1.txt', mode) -> System understandable Path.

fopen = open ('c:\\users\\desktop\\tfile1.txt', mode) -> system understandable Path.

fopen = open (r'c:\users\desktop\tfile1.txt', mode) -> user understandable Path.

-> open function returns the address of the channel and variable 'fopen' is holds that address, it is called as 'file object'.

close( ): close the file and delete the file pointer address.

Syntax:

fopen.close( )

fopen : is nothing but file pointer object name.

-> In the file handling concept, we are given the permissions or restrictions to that file. This type of concept we call it as file modes.

-> The modes are divided into three types:

Modes : ① read ② write ③ append

read:

Modes : -'r' (flat file) and
-'rb' (binary file)

Advances modes :-

-'r+' (read and write flat file) (append)

-'rb+' (read and write binary file) (append)

Read functions :-

read (no of characters) : this is a function used to read entire content or the specified number of characters from the file. If in case, we are not defined the number of characters then it reads each and everything characters from the file.

syntax :

Var = fopen.read (character count)

character should be integer

Var = fopen.read()

Where : No of characters or Character count is optional

fopen is nothing but it will store open file pointer object addresses.

example :

```
fp = open (r"C:\users\.....", 'r')
res = fp.read()
print (res)
fp.close()
```

Wap to count the number of times given string is occured from selected text file.

```
fp = open(r"C:\.....", 'r')
res = fp.read()
print(res)
l = res.split()
print(l)
count = 0
start = 0
end = len(l)
while start < end:
    if 'language' == l[start]:
        count += 1
print(count)
fp.close().
```

Wap to count the number of times given string is occured from selected text file.

```
fp = open(r"C:\.....", 'r')
res = fp.read()
l = res.split().count('language')
```

```
print (l)
fp. close ()
```

8-4-23.

WAP that reads a file and searches for a given string, displaying all the lines from the ~~file~~ file that contains that string.

```
fp = open (r`filepath', 'r')
res = fp.read ()
l = res.split('\n')
n = eval (input ("enter the value: "))

start = 0
while start < len (l):
        if n in l [start]:
                print (l [start])

        start+ =1.
```

Wap

```
fp = open(r'file path', 'w')
data = "..... ....."
fp.write(data)
fp.close()
```

readline(): this function is going to read the first line of the file.

var = fopen.readline(character count)

-> Starting file pointer to specific character position

var = fopen.readline()

-> start file pointer to end of the line.

readlines(): this function reads all the lines of data in the form of list

var = fopen.readlines()

-> each and every line is considered as an element in the list.

write mode :- open the file in the write mode. If we perform a write

operation, old data is overridden to new data.

Modes : 'w' (flat file) and 'wb' (binary file)

Advances modes : 'w+' (write and read flat file) and 'wb+' (write and read binary file)

Write functions :

write( ) : this a function used to write the string data into the file.

writelines( ) : this function is used to write a list of strings.

Note : In writelines we should pass values in list format.

append mode : this is kind of write mode but it will not create a new file every time instead, it will add the new content to the

existing in that file.

Modes : 'a' (flat file) and 'ab' (binary file).

advances modes : 'a+' (appends and read flat file) and 'ab+' (appends and read binary file)

tell() : we can use the tell() function to return the current position of the cursor (file pointer) from the beginning of the file.

seek() : we can use seek() function to move the cursor (file pointer) to a specified location.

'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 file not found error. This is default mode.

'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 available, then this mode is to create a new file and whatever we are given the name of the file in file path it takes the same name and creates it.

'r+' -> open an existing file for read and append operation into the file. The previous data in the file will not be deleted.

'w+' -> to write the data and read the data. It will override existing data.

'a' -> open a existing file for write operation. It won't override existing data. It is writing the new data in the file (append data). If the specified file is not available, then this mode is to create a new file.

a+ -> to write the data and read the data. It will not override existing data. To append the new data in the file.

example :

```
fp = open('file path', 'a+')
res = fp.read()
print(res)
fp.write("ideapad 3 slim3(15)")
fp.seek(0)
res = fp.read()
print(res)

fp.write("\n,.....")
fp.seek(0)
res = fp.read()
print(res)
fp.close()
```

11-4-23.

read the content from the existing file.

```
fp = open (r" C:\users\ ", 'r')
res = fp.read()
print (res)
fp.close()
print ('_' * 30)
```

append the new content from the existing file

```
fp = open (r" demo.text", 'a')
data = "\n Hi friends
        Bye friends"
fp.write (data)
fp.close()
```

read the updated content from the file

```
print ('_' * 30)
fp = open (r" demo.text", 'r')
res = fp.read ()
print (res)
fp.close()
```

append with read the content from the file.

```
fp = open(r"demo.txt", 'a+')
fp.seek(0)
res = fp.read()
print(res)
data = "....."
fp.write(data)
print("-" * 40)
fp.seek(0)
temp = fp.read()
print(temp)
fp.close()
```

Example :- read with append

```
fp = open(r"filename", 'a+')
res = fp.read()
print(res)
fp.close()
```

automatically cursor is present in starting of the file.

```
Content (index value = 0)
print("-" * 30)
```

append with read

```
fp = open ("filename.", 'at')
res = fp.read ()
print (res)
fp.close ()
```

example :-

```
fp = open ("filename", 'at')
data = " Java "
fp.write (data)
fp.close ()
```

automatically cursor is present in Starting of the file content (Index value = 0)

here, initially we are performing the append operation, then starting position to of the content is overide to length of the content position, remaining content is same.

```
print ('_' * 30)
```

```
fp = open("filename", "a+")
data = "Java"
fp.write(data)
fp.close()
```

automatically cursor is present in end of the file content. (Index value => len(content))

Here, the new content is append at the end of file content.

example :-

```
fp = open("filename", "a+")
res = fp.read()
print(res)
data = "Java"
fp.write(data)
fp.close()
```

automatically cursor is present starting of the file content (Index value => 0)

Here, initially we are performing the read operation then the cursor is moved to end of the content, after

that, we are performing the append operation. so the new content is append end of the existing content.

```
fp = open("filename", 'w+')
data = "Python was"
fp.write(data)
fp.seek(0)
res = fp.read()
print(res)
fp.close()
# automatically cursor at (index value = 0)
# here, perform write operation, but entire existing content is override to new content.
# cursor at end of content.
# read operation can't perform due to no content at file.
# to get entire file we use seek.
# by seek we go specific location from current position, next we perform read operation, return entire file content.
```

13-04-2023

```
fp = open(r"C:\users.....", 'r')
# res = fp.read()  # it will fetch the entire data from the file.
# res1 = fp.read(15)  # it will fetch specified no of characters from the file.
# res2 = fp.readline()  # it will fetch first line from the file.
# res 2 = fp.readline(15):  # it will fetch specified no of character in first line, from the file.
res2 = fp.readlines()
# it will fetch the all the lines from the file in the form of list format, each line it consider as one data-item.
print(res2).
```

program:-

```
fp = open(r"filepath", 'r')
res2 = fp.readlines()
```

extract the specific line from the file by using readlines function with indexing concept.

```
print(res2[2])
```

extract the specific lines from the file by using readlines function with slicing concept.

```
print(res2[0:5])
fp.close()
```

write functions :

```
fp = open(r"file Path", 'w')
data = ''
fp.write(data)
fp.close()
```

only the write the content inside the file.

write function will accept only string data.

```
fp = open(r"filepath", 'x')
res = fp.readlines()
print(len(res))
```

writelines function :-

```
fp = open(r'filepath', 'w')
fp.writelines([{'python\n': "programming",
'java\n': "language", 'django': "framework"}])
# fp.writelines([{'python\n', 'java\n',
'django'}])
# fp.writelines(['python\n', 'java\n', 'django'])
# fp.writelines("python\nis\n programming
\n language\n")
# only the write the content inside
the file. (in the form of either
single line or multiline)
# fp.writelines(['python\n', 'java\n',
'django'])
# writelines function will accept only
collection data types.
fp.close()
```

Program :-

write a program that reads a file and searches for a given word, and writes the new file and the old word is replaced with another given word. (manipulate the file).

```
fp = open (r"filepath", 'r')
res = fp.read()
r = res.replace('python', 'java')
print(r)
fp.close()
fp = open (r"filepath", 'w')
fp.write(r)
fp.close().
```

Program :-

write a program to read a file, writes the lines from 5th to 10th lines from input file into another file.

```
fp = open (r"filepath", 'r')
res = fp.readlines()
r = res[5:10+1]
fp.close()

fp.open (r"newfilepath", 'w')
fp.writelines (r)
fp. close ()
```

15-04-23.

Program :- (Manipulation inside a file).

```
fp = open (r"filepath", 'a+')
data = ['\n RAMU is Model\n', 'RAMU is a super Model RAMU is a bad boy\n', 'RAMU is extra Model']
fp.writelines (data)
fp. seek (0)
res = fp.read()
# print (res)
fp. seek (0)
res1 = fp. readlines()
# print(res1)
```

```
fp.close()
def replace(coll, start=0, end=0):
    old = input("enter the old word:")
    new = input("enter the new word:")
    if end == 0:
        end = len(coll)
    while start < end+1:
        print(start)
        l = coll[start].split()
        i = 0
        while i < len(l):
            if l[i] == old:
                l[i] = new
            i += 1
        coll[start] = ' '.join(l)
        start += 1
    return coll

fp = open(r'filepath', 'w+')
fp.writelines(replace(res1, 3, 5))

fp.seek(0)
res = fp.read()
print(res)
fp.close()
```

Program :-

```
fp = open(r"filePath", 'a+')
data = ['ramu is model\n', 'ramu is a
        super model ramu is a bad boy\n',
        'ramu is extra model']
fp.writelines(data)
fp.seek(0)
res = fp.read()
fp.close()
res1 = res.replace('ramu', 'siva')

fp = open(r"filepath", 'a+')
fp.write(res1)
fp.close()
```

Program :-

```
fp = open(r"filepath", 'a+')
data = ['ramu is model\n', 'ramu is
        super model ramu is a bady\n',
        'ramu is extra model']
fp.writelines(data)
fp.seek(0)
res = fp.read()
```

```
fp.seek(0)
res1 = res.replace('rahul', 'siva')
fp.write(res1)
fp.close()
```

18-4-23.

File handling to extract CSV file Content.

CSV → Comma Separated Values

Program 1 :-

```
import csv
fp = open('filePath', 'r')
res = csv.reader(fp)
print(res)
fp.close()
```

By using file handling to extract CSV file specific columns.

```
with open('filePath', 'r') as fp:
    res = fp.readlines()
```

```
for i in res:
    li = i.split(',')
print(li[0], li[1], li[2])
```

CSV file handling :

read the content from the CSV file using CSV module

-> if we want to use CSV module, first we have to import the CSV module from below syntax.

```
import csv
with open(r 'file path', 'r') as fp
    fr = csv.reader(fp)
    res = next(fr)
```

-> Before performing any operation, first open the CSV file by using open function with 'with' method and by using alias name.

Syntax :-

with open (CSV file path, operation mode) as alias name or variablename.

-> Read the content
-> (write) the content.

with statement block.

-> if we use with: method we don't want to close the file. Dynamically it will only save the content and close the file.

-> we can perform read the content or write the content.

-> if we want to perform read the content from csv file by using csv module, we have to use function called 'reader'

-> The reader function it will take the content from csv file and generates and return the reader object address.

syntax :-

csv.reader(csv file pointer address)

program :-

```
import csv
with open(r'filePath', 'r') as fp:
    fr = csv.reader(fp)
    for i in fr:
        print(i)
```

sales.csv

Name	age	date	item	item type	price	Total price
Sunil	28	06-4-2020	jeans	Casual wear	900	1100
Naga	28					
Ganesh	28	17-4-2020	tShirt	"	400	600.

WAP to extract the specific row details from given CSV file.

```
import csv
with open(r'filepath', 'r') as fp:
    fr = csv.reader(fp)
    line_row = 1
    for i in fr:
        if line_row == 5:
            print(i)
        line_row += 1
```

o/p:-

['Ganesh', '26', '17-04-2020', 'Tshirts', 'Semi formal', '400', '600']

WAP to extract the specific person details by using person name from the csv file.

```
import csv
with open(r"filePath", 'r') as fp:
    fr = csv.reader(fp)
    for i in fr:
        if i[0].upper() == 'kumar'.upper():
            print(i)
```

WAP to display the total purchase of specific person by using person name.

```
import csv
with open(r'filepath', 'r') as fp:
    fr = csv.reader(fp)
    sum1 = 0
```

```
for i in fr:
    if i[0].upper()=='KUMAR':
        sum1+=int(i[-1])
print(sum1)
```

WAP to display the purchase date's with total purchase of specific person by using person name:

```
import csv
fp=open(r'filepath','rt')
fr=csv.reader(fp)
sum1=0
for i in fr:
    if i[0].upper()=='KUMAR'.upper():
        print(i[2])
```

WAP to display the purchase date's and items with total purchase price of specific person by using person name.

```
fp=open(r'filepath','rt')
fr=csv.reader(fp)
```

```
name = input("enter the name:")
sum1 = 0
for i in fr:
    if i[0].upper() == name.upper():
        print(i[2], i[3], i[-1])
        sum1 += int(i[-1])
print("total price is", sum1)
fp.close()
```

Questions:

1. WAP to count the number of items in a file.
2. WAP to display the details of all the people, itemtype is casual wear.
3. WAP to display the total profit for casual wear.

20-4-23.

write content in CSV file.

```
import CSV
li = [['sunil', '28', '760122020',
'sunil@gmail.com'], ['Naga', '18',
'96012020', 'Naga@gmail.com']]
```

['mahesh', '16', '8601202o', 'mahesh@ghail.
Coh'] ['Ganesh', '26', '7601202o',
'Ganesh@ghail.Coh'], ['kumar', '30',
'9601202424', 'kumar@ghail.Coh']

```
fp = open(r'filepath', 'w', newline='')
write = csv.writer(fp)
write.writerow(['name', 'age', 'phone', 'email'])
write.writerow(li)
fp.close()

# for i in li:
#     write.writerow(i)
#
write.writerows(li)
fp.close()
```

reading csv file :-

-> If we want to read the content from the csv file we have to use the function name called reader.

-> reader is a pre-defined function it accept file pointer.

-> It will fetch the content from the file and store it inside the file reader object.

-> It returns file reader object.

-> If we want to get the content from the file reader object, we have to use dynamic iterator (for loop).

-> This will store the content in form of list with string format.

-> Each row it consider as one list.

-> Each value it will store in form of string inside list.

Syntax :-

variablename = CSV.reader(CSV file Pointer object address).

->Default delimeter is comma(,)

example :-

```
import CSV
fp = open(r'filepath', 'r')
res = CSV.reader(fp)
print(res)
for i in res:
    print(i)
```

writer :-

If we want to create a content in csv file we have to use function name called writer.

-> writer is a pre-defined function the writer function will give the permission to write the content inside the csv file.

-> the writer function will accept csv file pointer object address.

Note :- the csv file pointer object permission should be write.

-> It returns csv writer object address. by using writer object address we can store the content inside the writer object address by the help of writerow function. or writerows function.

writerow :- It is a pre-defined function this function will accept list iterable value. The list of string values are stored inside the writer object.

-> It will create only one row at a time.

writerows :- It is a pre-defined function this function will accept list of list rows.

-> all the list of list rows stored inside the writer object.

Dictionary reader :- It ~~is~~ will accept csv pointer object address and it return csv reader object address.

-> dictionary reader object address contain dictionary data.

ex :-

{"column name1": "data1", "column name2": "data2"}

rows.

program :-

```
import csv
fp = open(r'filepath', 'r')
res = csv.DictReader(fp)
for i in res:
    print(i)
```

O/p :-

{'name': 'Sujith', 'age': 28}

{'name': 'Naga', 'age': 18}

{'name': 'Mahesh', 'age': 24}

Dictionary writer :- Dictionary writer function will accept CSV file pointer address with [write mode] and it create write object for storing the data and it returns dictionary writer object address.

-> dictionary writer ~~object~~ function will accept one more argument called fieldnames in the form of collection.

-> By using writerow or writerows we can store the dictionary data inside the writer object address.

-> dictionary writer object data will store it inside the CSV file.

Program :-

```
li =[{'name':'sunil','age':'28',
    'branch':'ece'},
   {'name':'shashi','age':'23',
    'branch':'civil'},
   {'name':'ganesh','age':24
    'branch':'cse'}]
import csv
fp = open(r'filepath','w',newline="")
write = csv.dictwriter(fp,['name','age',
                           'branch'])
write.writeheader()
write.writerows(li)
fp.close().
```

Program (2) :-

```
def collect():
    name = input()
    age = input()
    phone = input()
    email = input()
```

```
return {'name': name, 'age': age,
        'phone': phone, 'email': email}

l = [ ]
start = 0
while start < 3:
    l += [collect()]
    start += 1
print(l)
```

program :-

```
li = [{'name': 'raja', 'age': '24', 'phone':
      '9988774455', 'email': 'raja@gmail.com'},
     {'name': 'rani', 'age': '22', 'phone':
      '9977446622', 'email': 'rani@gmail.com'}]
import csv
fp = open(r'filepath', 'w+', newline='')
write = csv.DictWriter(fp, ['name', 'email',
        'age', 'phone'])
write.writeheader()
write.writerows(li)
fp.seek(0)
```

```
res = csv.reader(fp)
for i in res:
    print(i)
fp.close()
```

25-4-23.

SQL Connection (MYSQL)

How to create database. (syntax)

>>> create database databasename;

to install pymysql in python:

->open CMD

--> pip install pymysql

mysql command line client:

-> open mysql command line client

-> given sql server user password

-> create a database

Cmd create database databasename;

Cmd use databasename;

Program :-

```
import pymysql
dbconn = pymysql.connect(host='127.0.0.1',
        port=3306, user='root',
        password='root', db='interview')
Print(dbconn)
Cus = dbconn.cursor()
print(cus)

def create-table(dbconn, Cus):
    Cus.execute("create table employee
        (id int, name varchar(50), age int,
        email varchar(50), phone bigint,
        gender varchar(10))")
    print("query ok, %.s rows affected"
        %.(cus.rowcount))
    db conn.commit()
#create-table(dbconn, Cus)

def insert-record(dbconn, Cus):
    id = int(input())
    name = input()
    age = int(input())
    email = input()
```

```
    Phone = int( )
    gender = input( )
    val = [id, name, age, email, Phone, gender]
    cus.execute("insert into employee
        (id, name, age, email, Phone, gender)
        values (%s, %s, %s, %s, %s,
            %s)", val)
    print("query ok, %s rows affected"
        %(cus.rowcount))
    dbconn.commit( )
insert_record(dbconn, cus)

def select_records(dbconn, cus):
    cus.execute("select * from employee")
    res = cus.fetchall( )
    for i in res:
        print(i)
# select_records(dbconn, cus)
```

```
def detail_record(dbConn, cus, pk):
    cus.execute("select * from employee")
    res = cus.fetchall()
    for i in res:
        if i[0] == pk:
            return i
    return "record is not available"
print(detail_record(dbConn, cus, 1))

def delete_record(dbConn, cus, pk):
    cus.execute("delete from employee
                 where id = ?", [pk])
    print("Query ok, %s rows
          affected" % (cus.rowCount))
    dbconn.commit()

delete_record(dbconn, cus, 1)
```