File Handling in Python

Why there is a need of File Handling?

- A file in itself is a bunch of bytes stored on some storage device.
- File helps us to store the data permanently , which can be retrieved for future use.

Types of Files

Text Files

- Stores information in ASCII or Unicode characters.
- Each line of text is terminated with a special character known as EOL (End of Line).
- Extension for text files is .txt
- Default mode of file.

Binary Files

Most of the files that we see in our computer system are called binary files.

Example:

- Image files: .png, .jpg, .gif, .bmp etc.
- Video files: .mp4, .3gp, .mkv, .avi etc.
- Audio files: .mp3, .wav, .mka, .aac etc.
- Archive files: .zip, .rar, .iso, .7z etc.
- Executable files: .exe, .dll, .class etc.

Binary File

- Contains information in the same format in which the information is held in memory.
- There is no delimiter for a line.
- No translations are required.
- More secure.

Difference between Text files and Binary Files

Text File	Binary File
Text files stores information in ASCII characters.	Binary files are used to store binary data such as images, audio, video and text.
Each line of text is terminated with a special character known as EOL (End of Line)	There is no delimiter in Binary File.
Text files are easy to understand because these files are in human readable form.	Binary files are difficult to understand.
Text files are slower than binary files.	Binary files are faster and easier for a program to read and write than the text files.
Text files are having extension .txt	Binary files are having extension .dat

Two methods to open a File

- ✓ **Using open ( ) function**
- ✓ **Using with statement**

How to open a text file

FileObject = open(<filename>)

OR

FileObject = open(<filename>,<mode>)

Example:

```
F = open("myfile.txt",r)

F= open ("C:\\Users\\sony\\Desktop\\TextFile.txt", "r")

F = open (r"C:\Users\sony\Desktop\TextFile.txt", "r")
```

Using with statement

This method is very handy when you have two related operations which you would like to execute as a pair , with a block of code in between.

Syntax :

```
with open (<FileName>, <FileMode>) as <File Handle> :
        f.write(“ ….”)
```

Example

```
with open ("Output.txt","w") as f:
    f.write ("Text")
```

```
#Two Methods to Open a File.....

#Method 1 (using open() function)
f=open("MyFile.txt","w")
f.write("Welcome to My Tutorial of Python Programming..")
f.close()
print("Data written in a file..")

#Method 2 (using with statement...)
with open("File2.txt","w") as f:
    f.write("Welcome to my Tutorial of Python Programming..")

print("Data written in a file..")
```

Benefits of using with statement

- It automatically closes the file after the nested block of code.
- It also handles all the exceptions also occurred before the end of block.

File Access Modes

Mode	Description
r	Opens file for reading only. This is the Default Mode
rb	Opens file for reading only in Binary format. This is the Default Mode
r+	Opens a file for both reading and writing.
rb+	Opens a file for both reading and writing in Binary format
w	Opens a file for writing only. Overwrites the file if already exists, else creates a new file
wb	Opens a file for writing only in Binary format.
w+	Opens a file for both reading and writing
wb+	Opens a file for both reading and writing in Binary format
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in Binary format.
a+	Opens a file for both appending and reading
ab+	Opens a file for both appending and reading in Binary format

r+ mode	w+ mode
Used for both reading and writing	W+ mode is also used for both reading and writing.
In r+ mode, the file pointer is at the beginning of the file.	In w+ mode, the file pointer is at the end of the file.
If the file does not exists, it will display the FileNotFound Error.	If the file does not exist, it will create the new file.
If the file exist, it doesn't show any error.	If the file exists, it will write the data in the file (overwriting the previous content).

READ TEXT FILE

Reading Data from a file

- read( )
- readline()
- readlines( )

read()

- Reads at most n bytes ; if no n is specified , reads the entire file.
- Returns the read bytes in the form of a string.
- Syntax:

<Filehandle>.read([n])

```
f=open('Poem.txt','r')
data=f.read()
print(data)
```

```
f=open('C:\\Users\\sony\\Desktop\\TextFile.txt','r')
data=f.read()
print(data)
```

```
f=open(r'C:\Users\sony\Desktop\TextFile.txt','r')
data=f.read()
print(data)
```

```
f=open(r'C:\Users\sony\Desktop\TextFile.txt','r')
data=f.read(150)
print(data)
```

```
f=open(r'C:\Users\sony\Desktop\TextFile.txt','r')
data=f.read(150)
print(data)
print("Type=",type(data))
```

readline()

- Reads a line of input ; if n is specified reads at most n bytes.
- Returns the read bytes in the form of a string ending with '\n' character.
- Returns a blank string if no more bytes are left for reading in the file.
- Syntax : <Filehandle>.readline()

```
f=open(r'C:\Users\sony\Desktop\TextFile.txt','r')
data=f.readline(15)
print(data)
```

OR print(data, end=' ')

readlines()

- Reads all the lines of a text file.
- Returns in the form of list.
- Syntax :
- <FileHandle>.readlines()

```
f=open(r'C:\Users\sony\Desktop\TextFile.txt','r')
data=f.readlines()
print(data)
print(type(data))
```

```
f=open(r'C:\Users\sony\Desktop\TextFile.txt','r')
data=f.readlines()
for i in data:
    print(i)
print(type(data))
```

While writing data in a file.....

- If **file doesn't exists**, it will **create the new file**.
- If **file exists**, it will **write the data** in the file.

How to write data in a file?

- write( )
- writelines( )

write(string)

- write( ) method takes a string and writes it in the file.
- For storing data, with EOL character, we have to add '\n' character to the end of the string.
- For storing numeric value, we have to either convert it into string using str( ) or write in quotes..

```
f=open("DemoWrite.txt","w")
f.write("Computer Science")
f.close()
```

```
f=open("DemoWrite.txt","w")
f.write("Computer Science")
f.write("\nClass XII")
f.close()
```

writelines(seq)

writelines( ) method is used to write sequence data types in a file(string,lis and tuple etc.)

```
f=open("DemoWrite.txt","w")
list=['Computer\n','Mathematics\n','English']
f.writelines(list)
f.close()
```

```
f=open("DemoWrite.txt","w")
t=('Computer Science\n','Mathematics\n','English\n','Physics')
f.writelines(t)
f.close()
```

Appending data to a file....

- Append means to add the data at the end of file using ‘a’ mode.
- If file doesn’t exists, it will create the new file.
- If file exists, it will append the data at the end of a file.

Closing Files

A close() function breaks the link of file - object and the file on the disk.

After close(), no tasks can be performed on that file through file object or file – handle.

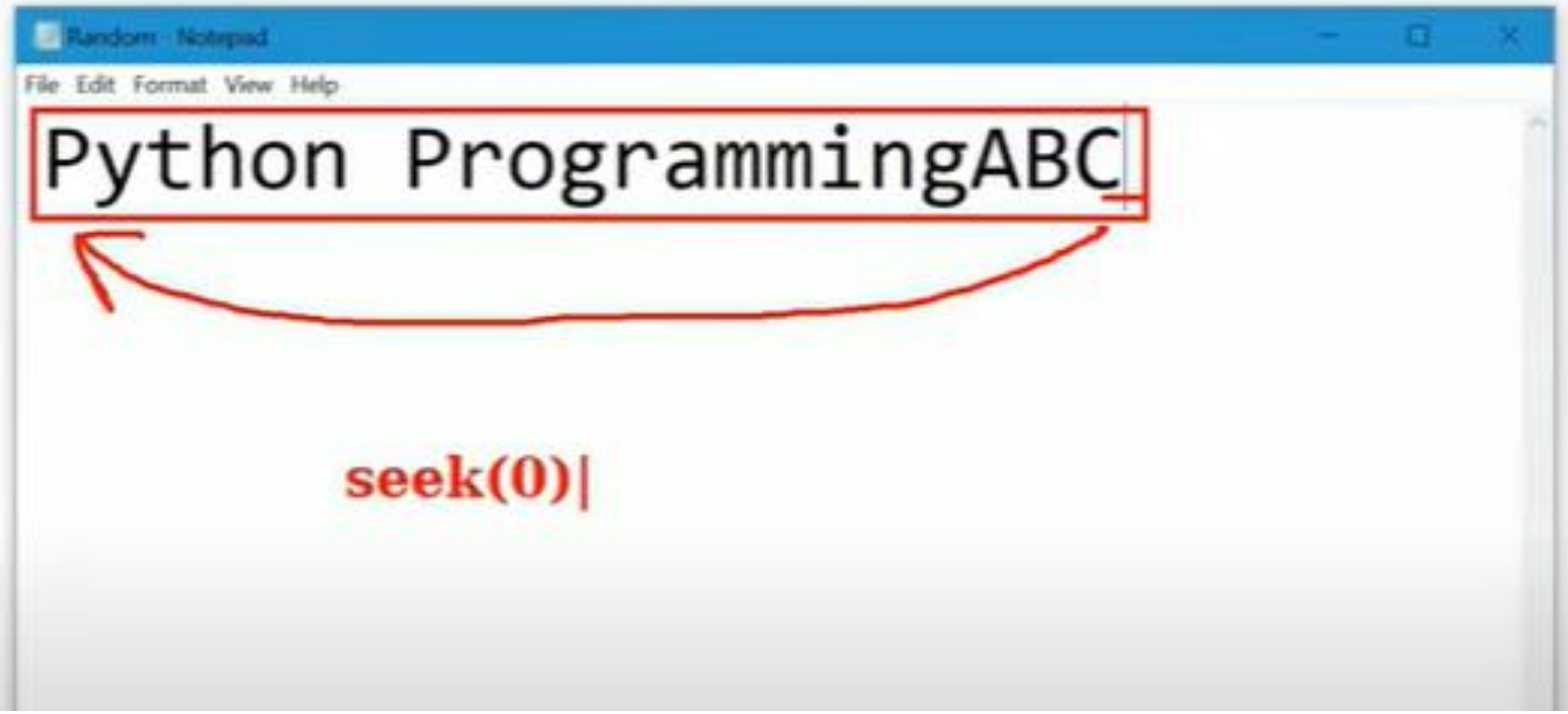
Syntax:

<FileHandle>.close()

Example:

F.close()

```
f=open(r"C:\Users\USER\Desktop\Random.txt","r+")
print(f.read())
f.write("ABC")
print(f.read())
f.close()
```



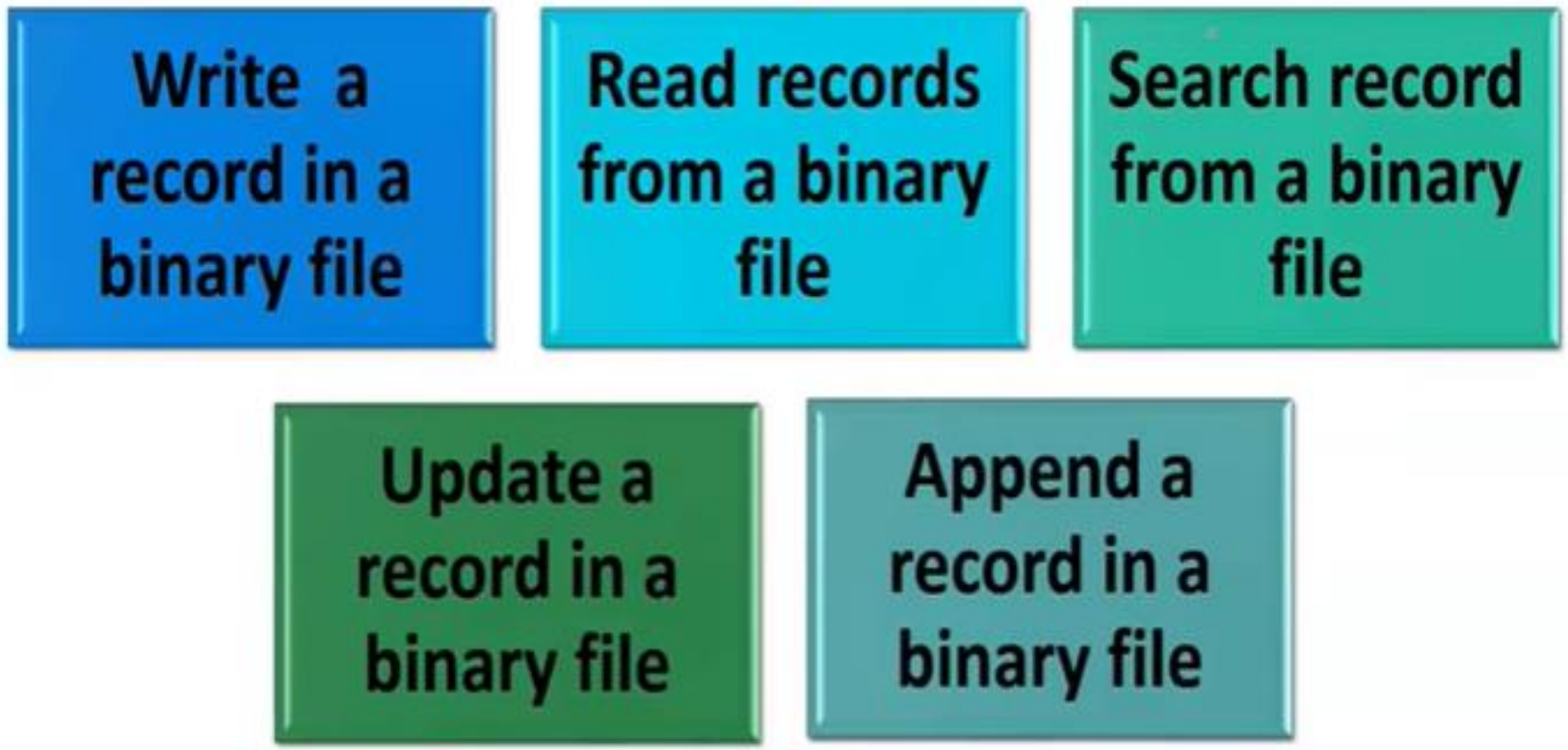

```
f=open(r"C:\Users\USER\Desktop\Random.txt","r+")
print(f.read())
f.write("ABC")
f.seek(0)
print(f.read())
f.close()
```

```
f=open(r"C:\Users\USER\Desktop\Random.txt","w+")
f.seek(0)
print(f.read())
f.close()
```


```
f=open(r"C:\Users\USER\Desktop\Random.txt","w+")
f.write("ABC")
f.seek(0)
print(f.read())
f.close()
```

```
f=open(r"C:\Users\USER\Desktop\Random.txt","a+")
f.write("ABC")
f.seek(0)
print(f.read())
f.close()
```

Operations performed on Binary File



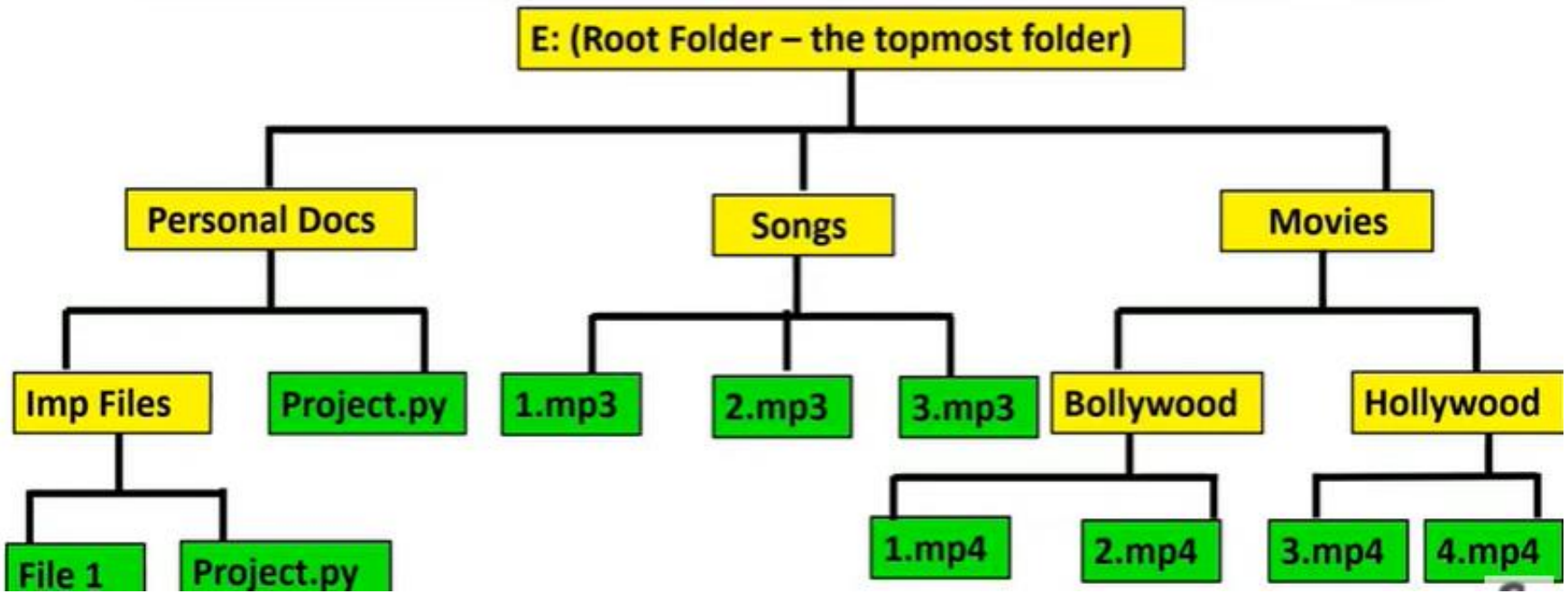
To read data , from the keyboard, we need	Keyboard (Standard Input Device
To display the output on the screen, we need	Monitor (Standard Output Device)
An error, if occurs is also displayed on the	Monitor (Standard Error Device)

Path

The full name of a file or directory or folder consists of path.

Path is a sequence of directory names which give you the hierarchy to access a particular directory or file name.

Directory Structure



Absolute Path

The absolute paths are from the topmost level of directory structure.

E.g.

E:\Personal Docs

E:\Personal Docs\Imp Files\Project.py

E:\Personal Docs\Project.py

E:\Movies\Bollywood\2.mp4

Two files having same name but different location can exist.

Relative Path

The relative paths are relative to the current working directory, denoted as a dot (.) , while its parent directory is denoted with two dots(..)

```
f=open(r'C:\Users\USER\Desktop\Folder 1\Folder 2\File
print(f.read())
f.close()
```

```
f=open(r'.\Folder 2\File 1.txt',"r")
print(f.read())
f.close()
```

Random Access in Files

- seek( )
- tell( )

seek()

seek() function is used to change the position of the file handle (file pointer) to a given specific position.

File pointer is like a cursor, which defines from where the data has to be read or written in the file.

Syntax :

```
f.seek(offset, from_what)
```

#where f is file pointer

seek()

The reference point is defined by the "from_wha
argument. It can have any of the three values:

0: sets the reference point at the beginning of the file, which is by default.

1: sets the reference point at the current file position.

2: sets the reference point at the end of the file.

But, in Python 3.x and above, we can seek from beginning only, if opened in text mode. We can overcome from this by opening the file in b mode.

Tell() function return the current location of file pointer in file

```
f=open("Random.txt","r")
print(f.tell())
```

Return 0 i.e at the beggining

```
f=open("Random.txt","r")
print(f.tell())
f.seek(3)
print(f.read())
f.seek(6)
print(f.read(5))
print("Current location of pointer=",f.tell())
```

File pointer shifted to at 3rd location start from the beginning

Again, file pointer shifted at 6th location start from the beginning

```
f=open("Random.txt","rb")
print(f.tell())
f.seek(3)
print(f.read())
f.seek(-6,1)
print(f.read(5))
print("Current location of pointer=",f.tell())
```

```
f=open("Random.txt","rb")
print(f.tell())
f.seek(3)
print(f.read(5))
print("Current position of pointer=",f.tell())
f.seek(2,1)
print(f.read(5))
print("Current position of pointer=",f.tell())
```

```
f=open("Random.txt","rb")
f.seek(4)
print(f.read(3))
f.seek(-5,2)
print(f.read())
print(f.tell())
f.seek(5,1)
print(f.read())
```

MOST IMPORTANT QUESTIONS BASED ON TEXT FILE

Write a method in Python to count the total number of words in a file.

```
def CountWords():
    f=open("India.txt","r")
    count=0
    text=f.read()
    words=text.split()
    for i in words:
        count+=1
    print("Total number of words=",count)


CountWords()
```

Write a method in Python to read lines from a text file 'India.txt', to find and display the occurrence of the word 'India'.

```
def CountWords():
    f=open("India.txt","r")
    count=0
    text=f.read()
    words=text.split()
    for i in words:
        if i=='India':
            count+=1
    print("Total occurences of word India are=",count


CountWords()
```

Write a method BIGWORDS() in Python to read contents from a text file and display the occurrence of those words which are having 5 or more alphabets.

```
def BIGWORDS():
    f=open("India.txt","r")
    count=0
    text=f.read()
    words=text.split()
    for i in words:
        if len(i)>=5:
            count+=1
    print("Total big words >=5=",count)


BIGWORDS()
```

Write a Python statement to open a text file “Demo.txt” in read mode.

```
F=open(“Demo.txt”,’r’)
```

Write a function definition to search and display all those records of the students whose marks are between 50 and 70 (bo values included)

To open a text file "Book.txt" in read mode.

f=open("Book.txt","r")

(b) To open a binary file "Book.dat" in write mode

f=open("Book.dat","w")

Consider the following code:

```
f = open("test", "w+")
f.write("0123456789abcdef")
f.seek(-3,2) //Statement 1
print(f.read(2)) //Statement 2
```

Explain statement 1 and give out
statement 2.

Yogendra intends to position the file pointer to the beginning of a text file. Write Python statement for the same assuming "F" is the Fileobject.

F.seek(4,2)

What is the output of the following code?

```
fh = open("test.txt", "r")
Size = len(fh.read())
print(fh.read(5))
```

Pickling and Unpickling

Pickling refers to the process of converting the structure (such as list or dictionary) to a byte stream before writing to the file

Pickling

Pickling and Unpickling

Unpickling refers to the process of converting the byte stream back to the original structure.

Unpickling

pickle.dump()

- Used to write the object in a file.
- Syntax:

  pickle.dump(<Structure>,FileObject)

- Where,
- Structure can be any sequence of Python. It can be either list or dictionary.
- FileObject is the file handle of the file, in which to write.

pickle.load()

- Used to read the data from a file.
- Syntax:

  Structure = pickle.load(FileObject)

- Where,
- Structure can be any sequence of Python. It can be either list or dictionary.
- FileObject is the file handle of the file, in which to write.

```
import pickle
def write():
    f=open("BinaryWrite.dat",'wb')
    list=['Comp Sci','Maths','English','Physics','Chemsitry']
    pickle.dump(list,f)
    f.close()

write()
print("Data written in file sucessfully....")
```

```
def write():
    f=open("BinaryWrite.dat",'wb')
    list=['Comp Sci','Maths','English','Physics','Chemsitry']
    dic={'Comp':100,'Maths':98,'English':74,'Physics':40,'Chem':80}
    pickle.dump(list,f)
    pickle.dump(dic,f)
    f.close()

def read():
    f=open("BinaryWrite.dat","rb")
    lst=pickle.load(f)
    d=pickle.load(f)
    print(lst)
    print(d)
    f.close()
write()
read()
```

```
import pickle
def write():
    f=open("SampleData.dat","wb")
    record=[]
    while True:
        rno=int(input("Enter roll no:"))
        name=input("Enter name:")
        grade=input("Enter grade:")
        marks=int(input("Enter marks:"))
        data=[rno,name,grade,marks]
        record.append(data)
        ch=input("Do you want to enter more records?(y/n)")
        if ch=='n':
            break
    pickle.dump(record,f)

def read():
    f=open("SampleData.dat",'rb')
    s=pickle.load(f)
    for i in s:
        rno=i[0]
        name=i[1]
        grade=i[2]
        marks=i[3]
        print(rno,name,grade,marks)
    f.close()

        marks=i[3]
        print(rno,name,grade,marks)
    f.close()

def search():
    f=open("SampleData.dat","rb")
    s=pickle.load(f)
    print("Displaying records having marks in range of 50 and 70:")
    for i in s:
        if i[3]>=50 and i<[3]<=70:
            print(i)


write()
read()
search()
```