# PYTHON FOR COMPUETR SCIENCE

## FILE HANDILING

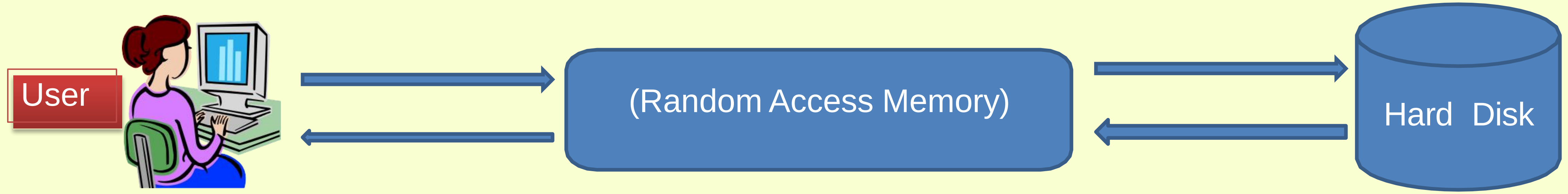
## FOR

## CLASS – XII

KAPIL SEHGAL (P.G.T. (CS) K.V. NO2 Delhi Cantt,

## NEED FOR DATA FILE

- The data stored with in a file is known as persistent data because this data is permanently stored in the system.
- Python provides reading and writing capability of data files.
- We save the data in the files for further use.
- As you save your data in files using word, excel etc. same thing we can do with python.
- “A File is a collection of characters in which we can perform read and write functions. And also we can save it in secondary storage.”



KAPIL SEHGAL

# FILE TYPES

**File are of two types –**

1. **Text File:** A text file is sequence of line and line is the sequence of characters and this file is saved in a permanent storage device. Although in python default character coding is ASCII but by using constant 'U' this can be converted into UNICODE. In Text File each line terminates with a special character which is EOL (End Of Line). These are in human readable form and these can be created using any text editor.

2. **Binary File:** Binary files are used to store binary data such as images, videos audio etc. Generally numbers are stored in binary files. In binary file, there is no delimiter to end a line. Since they are directly in the form of binary hence there is no need to translate them. That's why these files are easy and fast in working.

## DATA FILE OPERATIONS

Data File Operations

1. Opening a File
2. Perform Operations (i.e. Read or Write etc.)
3. Closing the File

Beside above operations there are some more operations can be done on files.-

- Creating of Files
- Traversing of File
- Appending Data into file.
- Inserting Data into File.
- Deleting Data from File.
- Copying of File.
- Updating Data into File.

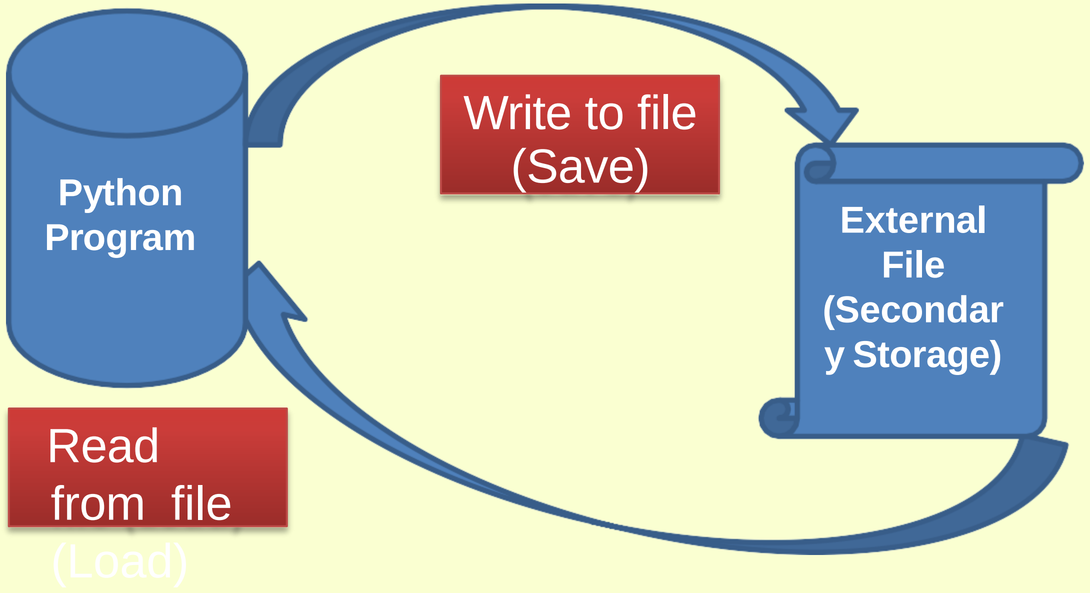
## OPENING & CLOSING FILES

- We need a *file variable* or *file handle* to work with files in Python.
- This file object can be created by using open( ) function or file( ) function.
- Open( ) function creates a file object, which is used later to access the file using the functions related to file manipulation.
- Its syntax is following -

```
<file_object>=open(<file_name>,<access_mode>)
<file_object>.close()
```

### Basic operations

1. Write into File
2. Read From File.



KAPIL SEHGAL

## SIMPLE PROGRAM

### Normal Program

```
name=input(“Enter name “)
print(nm)
```

### File Creation Programe

```
file=open(“sample.txt”,”w”)
name=input(“Enter name “)
file.write(name)
```

### Modes of open a File

“w” : Creating a new file
“r”  : Reading an existing file
“a” : Append into file.

### File Reading Programe

```
file=open(“sample.txt”,”r”)
data=file.read()
print(data)
```

KAPIL SEHGAL

# APPEND MODE

## Normal Program

```
name=input(“Enter name “)
print(nm,ad)
```

## Append in File

```
file=open("sample.txt","a")
str="This is the line \n"
data=file.write(str)
file.close()
```

## File Creation Programe

```
file=open(“sample.txt”,”w”)
str=input(“Enter string “)
file.write(str)
file.close()
```

## File Reading Programe

```
file=open("sample.txt","r")
data=file.read()
print(data)
file.close()
```

## Modes of open a File

“w” : Creating a new file

“r” : Reading an existing file

“a” : Append into file.

## WRITELINES METHOD

```
file=open("sample.txt","w")
lst=["Computer Science \n","Physics \n","Mathematics \n","Chemistry \n","Hindi \n","English \n"]
file.writelines(lst)
print("Data Write Successfully ")
file.close()
```

## READ N METHODS

```
file=open("sample.txt","r")
data=file.read(5)
print(data, " ")
file.close()
```

```
file=open("sample.txt","r")
data=file.read(5)
print(data, " ")
data=file.read(5)
print(data, " ")
data=file.read(5)
print(data, " ")
file.close()
```

KAPIL SEHGAL

# READLINE METHOD

Readline methods read one line at a time. When file is open, by default file pointer is always at first character of file (or first line)

New.txt

```
Computer Science
Physics
Chemistry
Mathematics
These
are the
Compulsory
Subjects.
```

KAPIL SEHGAL

**Program to read a file line wise**

```
file=open("new.txt","r")
line=file.readline()
print(line)
line=file.readline()
print(line)
line=file.readline()
print(line)
file.close()
```

Output

```
>>>
Computer Science

Physics

Chemistry
```

## READLINES METHODS

Readlines methods reads all lines in one go and store in list i.e. one line as one element of list.

New.txt

```
Computer Science
Physics
Chemistry
Mathematics
```

Program to read a file with readlines method

```
file=open("new.txt","r")
line=file.readlines()
print(line)
file.close()
```

Output

```
['Computer Science \n', 'Physics
\n', 'Chemistry \n',
'Mathematics\n']
>>>
```

# COMPARES BETWEEN READ( ) AND READLINES( )

## Similarity

read() method and readlines() method both extract entire data from file in one go

## Difference

read() method extracts entire data into a single string variable. But readlines() method extracts entire data into form of list of string, it means each line of text file will be an element of List and in form of String.

New.txt

```
Computer Science
Physics
Chemistry
Mathematics
```

Output with read()

```
Computer Science
Physics
Chemistry
Mathematics
```

Output with readlines()

```
['Computer Science \n', 'Physics
\n', 'Chemistry \n',
'Mathematics\n']
>>>
```

**KAPIL SEHGAL**

## READ FUNCTIONS TO READ DATA FROM FILES

There are four read functions

1. read() :- Read the entire file in one go and store data into string form
2. read(n) :- Read the n characters from current location of file pointer in file. (When file is open that time file pointer be at first position.
3. readline() :- Read the file line by line and store line into string form
4. readlines() : Read the entire file in one go and store in form of List of String i.e. One line will be one element of list and that element in form of string.

## WRITE FUNCTIONS TO READ DATA FROM FILES

There are two write functions

1. write() :- Write the string into file, it takes only one string type parameter.
2. writelines() : Write the list into file, one element of string in one line.

**KAPIL SEHGAL**

# IMPORTANT PROGRAMS – CBSE EXAM BASED

## WRITE A PROGRAM TO WRITE FIVE NAMES INTO FILE USING WRITELINES

```
file=open("new.txt","w")
lst=[]
for i in range(0,5):
    nm=input("Enter name ")
    lst.append(nm+"\n")
file.writelines(lst)
file.close()
```

## WRITE A PROGRAM TO READ A FILE AND DISPLAY SIZE OF FILE IN BYTES

Let the file “new.txt” Contains

```
Hello Students how are you?
I hope you all are fine.
```

Program

```
file=open("new.txt","r")
data=file.read()
size=len(data)
print("Size of File in Bytes ", size)
file.close()
```

Output.

```
Size of File in Bytes  52
```

KAPIL SEHGAL

# IMPORTANT PROGRAMS

Let the file “new.txt” Contains

Hello Students how are you?
I hope you all are fine.

## WRITE A PROGRAM TO READ A FILE AND PRINT NUMBER OF LINES IN A FILE

```
file=open("new.txt","r")
data=file.readlines()
nol=len(data)
print(“Number of Lines ", nol)
file.close()
```

Output.

```
Number of Lines 2
```

**KAPIL SEHGAL**

## WRITE A PROGRAM TO READ A FILE AND COUNT HOW MANY WORDS IN IT.

```
file=open("new.txt","r")
data=file.read()
lst=data.split()
totalwords=len(lst)
print(“Total words : ", totalwords)
file.close()
```

Output.

```
Total words : 11
```

## WRITE A PROGRAM TO READ A FILE AND COUNT HOW MANY WORDS STARTS WITH “A”

```
file=open("new.txt","r")
data=file.read()
lst=data.split()
count=0
for i in lst:
    if (i[0]=='a' or i[0]=='A'):
        count=count + 1
print("Total words ",count)
file.close()
```

Output.

```
Total words : 2
```

## “X” MODE TO OPEN A FILE

When we open a file in “w” mode then Python open a fresh file, if it is already exist then overwrite it without any warning. Instead of using “w” mode, we can use “x” mode.

Like “w” mode “x” mode also open the fresh file but unlike “w” mode, “x” mode do not overwrite file if it is already exist but it will show run time error.

For Ex.

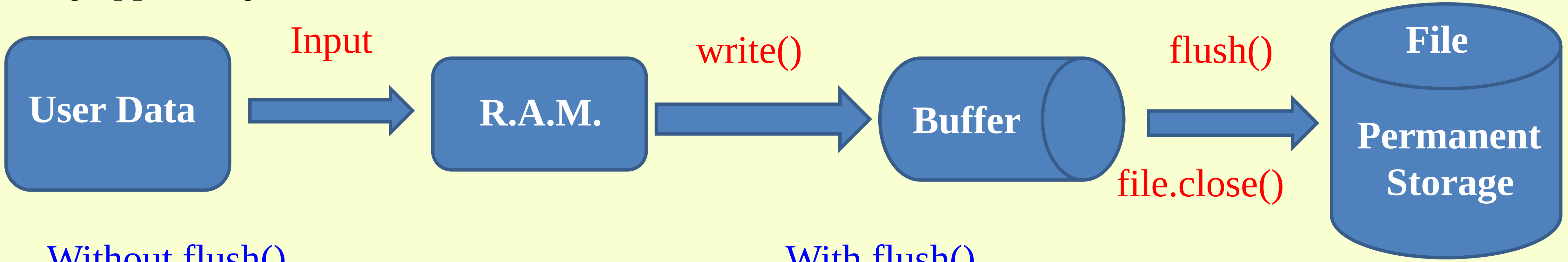
```
file=open("new.txt","x")
str=input("Enter String ")
file.write(str)
file.close()
```

Error Message

```
RESTART: C:\Users\Kapil\AppData\Local\Programs\Python\Python35\createfile.py
Traceback (most recent call last):
  File "C:\Users\Kapil\AppData\Local\Programs\Python\Python35\createfile.py", line 1, in <module>
    file=open("new.txt","x")
FileExistsError: [Errno 17] File exists: 'new.txt'
```

# FLUSH METHOD

**flush() method** is an inbuilt method in Python, it is used to clear/flush the internal buffer, it is best practice while working with file handling in Python, the internal buffer can be cleared before writing/appending the new text to the file.



## Without flush()

```
file=open("new.txt","w")
str=input(“First Interruption ")
file.write("one")
file.write("Two")
st=input(“Second Interruption")
file.write("three")
file.close()
```

## With flush()

```
file=open("new.txt","w")
str=input("First Interruption ")
file.write("one")
file.write("Two")
file.flush()
st=input(“Second Interruption ")
file.write("three")
file.close()
```

KAPIL SEHGAL

# OPEN A FILE THROUGH "WITH"

We can also open a file through "with" keyword. Using this way we don't need to close file.

Program through open method

```
file=open("new.txt","r")
line=file.readlines()
print(line)
file.close()
```

Program through with

```
with open("new.txt","r") as file :
      Line=file.readlines()
      print(line)
```

# FILE POINTER

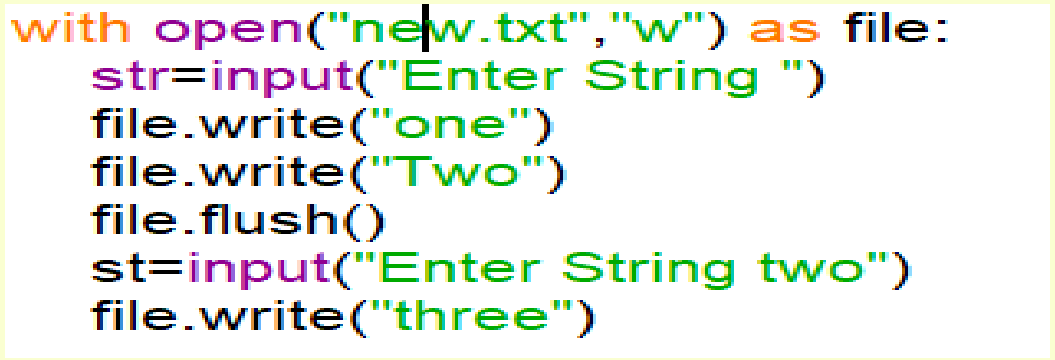
**A file pointer is simply a marker which keeps track of the number of bytes read or written in a file. This pointer automatically moves after every read or write operation.**

When a file is opened the file pointer points at the beginning of the file. The write() function begins writing at the current file position and then increments the file pointer. For example, the following figure shows the position of file pointer after each write operation.

## There are two methods to maintain file pointer

**tell() :-** tell function returns the current position of file pointer.

**seek(offset) :-** seek(offset) function moves the file pointer to the given offset from the origin



```
with open("new.txt","w") as file:
    str=input("Enter String ")
    file.write("one")
    file.write("Two")
    file.flush()
    st=input("Enter String two")
    file.write("three")
```

**In this image of file we seen the file pointer at 13**

Let the file “new.txt” Contains

# TELL & SEEK METHOD PROGRAM

Hello Students how are you?
I hope you all are fine.

## Program for tell() methods

```
file=open("new.txt","r")
location=file.tell()
print(location)
data=file.read(5)
print(data)
location=file.tell()
print(location)
data=file.read(3)
print(data)
location=file.tell()
print(location)
```

Output

```
0
Hello
5
 St
8
```

## Program using seek() method

```
file=open("new.txt","r")
data=file.read(5)
print(data)
data=file.read(3)
print(data)
file.seek(15)
data=file.read(3)
print(data)
file.seek(32)
data=file.read(3)
print(data)
file.close()
```

Output

```
Hello
 St
how
hop
```

KAPIL SEHGAL

## IMPORTANT PROGRAMS

Let the file “new.txt” Contains

TXT files are useful for storing information in plain text with no special formatting beyond basic fonts and font styles. The file is commonly used for recording notes, directions, and other similar documents that do not need to appear a certain way

KAPIL SEHGAL

### WRITE A PROGRAM TO READ A FILE AND COUNT HOW MANY LINES STARTS WITH “A”

```
file=open("new.txt","r")
data=file.readlines()
count=0
for i in data:
   if (i[0]=='a' or i[0]=='A'):
      count=count + 1
print("Total Lines ",count)
file.close()
```

Output.

```
Total Lines : 1
```

### WRITE A PROGRAM TO READ A FILE AND COUNT HOW MANY WORDS ENDS WITH “A”

```
file=open("new.txt","r")
data=file.readlines()
count=0
for i in data:
   if (i[-2]=='a' or i[-
2]=='A'):
       count=count + 1
print("Total words ",count)
file.close()
```

Output.

```
Total Lines : 0
```

# IMPORTANT PROGRAMS

Let the file “new.txt” Contains

TXT files are useful for storing information in plain text with no special formatting beyond basic fonts and font styles. The file is commonly used for recording notes, directions, and other similar documents that do not need to appear a certain way

**KAPIL SEHGAL**

## WRITE A PROGRAM TO READ A FILE AND COUNT HOW MANY WORDS OF LENGTH 5 (N)

```
file=open("new.txt","r")
data=file.read()
count=0
words=data.split()
for i in words:
    if (len(i)==5):
        count=count + 1
print("Total words of length 5  : ",count)
file.close()
```

Output.

```
Total words of length 5  :  5
```

## WRITE A PROGRAM TO READ A FILE AND AND PRINT ONLY DIGITS AND NUMBERS

```
file=open("new.txt","r")
data=file.read()
for i in data:
        if (i.isdigit()):
            print(i)
file.close()
```

Output.

No output because text file new.txt does not contains number or digits

# IMPORTANT PROGRAMS

Let the file “new.txt” Contains

TXT files are useful for storing information in plain text with no special formatting beyond basic fonts and font styles. The file is commonly used for recording notes, directions, and other similar documents that do not need to appear a certain way

## WRITE A PROGRAM TO READ A FILE AND PRINT THOSE LINES WHICH STARTS WITH ‘S’ ALONG WITH LINE NUMBER

```
file=open("new.txt","r")
data=file.readlines()
count=0
for i in data:
    count=count+1
    if (i[0] in ['s','S']):
        print(count,i)
file.close()
```

Output.

```
2 storing information in

8 similar documents that do not
```

## WRITE A PROGRAM TO READ A FILE CALLED “NEW.TXT” AND COPY ALL THE CONTENT TO “NEW1.TXT”

```
file=open("new.txt","r")
file1=open("new1.txt","w")
data=file.read()
file1.write(data)
file.close()
file1.close()
```

KAPIL SEHGAL

# IMPORTANT PROGRAMS

Let the file “new.txt” Contains

TXT files are useful for storing information in plain text with no special formatting beyond basic fonts and font styles. The file is commonly used for recording notes, directions, and other similar documents that do not need to appear a certain way

KAPIL SEHGAL

## WRITE A FUNCTION TO READ A FILE CALLED “NEW.TXT” AND COPY ALL THE WORDS WHICH START WITH ‘S’ COPY TO “NEW1.TXT”

```
file=open("new.txt","r")
file1=open("new1.txt","w")
data=file.read()
words=data.split()
for i in words:
    if (i[0] in ['s','S']):
        file1.write(i)
        file1.write(" ")
file.close()
file1.close()
```

## WRITE A FUNCTION TO READ A FILE CALLED “NEW.TXT” AND COPY ALL THE WORDS WHICH START WITH ‘S’ CONTENT TO “NEW1.TXT”

```
def copyfile():
    file=open("new.txt","r")
    file1=open("new1.txt","w")
    data=file.readlines()
    for i in data:
        if (i[0] in ['s','S']):
            file1.write(i)
            file1.write(" ")
    file.close()
    file1.close()
copyfile()
```

# MORE MODES TO OPEN A FILE

## More modes to open a file

1. "w+" : Open a file for write and read. (But first Write then Read in same file same program)
2. "r+" : Open a file for read and write (But first Read and then Write in same file same program

### Example of "w+" mode

```
file=open("new.txt","w+")
file.write("I am a students of Class 12 C Science")
file.seek(0)
data=file.read()
print(data)
file.close()
```

### Example of "r+" mode

Program for overwrite first five character by *****

```
file=open("new.txt","r+")
data=file.read()
file.seek(0)
file.write("*****")
file.close()
```

KAPIL SEHGAL

# IMPORTANT PROGRAMS FOR ASSIGNMENT

1. Write a function to read a file called “new.txt” and count how many “the” word in it.
2. Write a function to read a file and count how many word ‘Do”. “Do” will be in any case [“DO”,”do”,’Do”,”dO”]
3. Write a function to read a file “new.txt” and copy only those lines which ends with ‘s’ or “S” to another file “new1.txt”
4. Write a program to read a file “new.txt” print number of characters in each line.
5. Write a function counteven() to read a file and count how many even number in file “new.txt”
6. Write a program to read a file and count & print only vowels store in it.

**KAPIL SEHGAL**

## IMPORTANT PROGRAMS FOR ASSIGNMENT

Write a program to read a file and count & print only vowels store in it.

```
file=open("new.txt","r")
data=file.read()
count=0
v=[]
for i in data:
    if i in ['a','e','i','o','u','A','E','I','O','U']:
        if i not in v:
            v.append(i)
            count=count+1
file.close()
print(v)
print("Number of Vowel ",count)
```

## BINARY FILE

1. To handle binary file operation, we need a special library called “pickle”

2. Installation of “pickle” library, we write :

   pip install pickle-mixin

3. In binary file using pickle library, we can read and write various objects (like : list, tuple, dictionary etc)

4. Method to write different objects in binary file : dump()

5. Method to read different objects from binary file : load()

6. Mode for Creating a binary file (Fresh File) : wb

7. Mode for Reading a binary file (Existing File ) : rb # here b indicate binary

KAPIL SEHGAL

# BINARY FILE PROGRAMS

## WRITE A PROGRAM TO CREATE A BINARY FILE USING DICTIONARY DATA

```
import pickle
with open("new.dat","wb") as file:
    s1={"name":"Ram","Class":"12C"}
    s2={"name":"Mohan","Class":"12C"}
    pickle.dump(s1,file)
    pickle.dump(s2,file)
```

## WRITE A PROGRAM TO CREATE A BINARY FILE USING LIST AND TUPLE

```
import pickle
with open("new.dat","wb") as file:
    l1=[10,20,30,40,50]
    t1=(50,60,70,80)
    pickle.dump(l1,file)
    pickle.dump(t1,file)
```

## WRITE A PROGRAM TO READ BINARY FILE USING DICTIONARY DATA

```
import pickle
with open("new.dat","rb") as file:
    d1=pickle.load(file)
    d2=pickle.load(file)
print(d1)
print(d2)
```

## WRITE A PROGRAM TO READ A BINARY FILE USING LIST AND TUPLE

```
import pickle
with open("new.dat","rb") as file:
   list=pickle.load(file)
   tuple=pickle.load(file)
print(list)
print(tuple)
```

# BINARY FILE PROGRAMS

## CREATE A BINARY FILE WITH MULTIPLE OBJECTS

```
import pickle
with open("new.dat","wb") as file:
    l1=[10,20,30,40,50]
    l2=[60,70,80,90,100]
    l3=[110,120]
    l4=[130,140,150]
    pickle.dump(l1,file)
    pickle.dump(l2,file)
    pickle.dump(l3,file)
    pickle.dump(l4,file)
```

## READ A BINARY FILE WITH MULTIPLE OBJECTS

```
import pickle
mainlist=[]
with open("new.dat","rb") as file:
    while True:
        try:
            list=pickle.load(file)
            mainlist.append(list)
            print(list)
        except EOFError:
            break
print(mainlist)
```

Output.

```
[10, 20, 30, 40, 50]
[60, 70, 80, 90, 100]
[110, 120]
[130, 140, 150]
[[10, 20, 30, 40, 50], [60, 70, 80, 90, 100], [110, 120], [130, 140, 150]]
```

KAPIL SEHGAL

# APPEND IN BINARY FILE

## W.A.P. TO APPEND A LIST INTO BINARY FILE

```
import pickle
with open("new.dat",“ab") as file:
    l1=[110,210,310,410,510]
    pickle.dump(l1,file)
    pickle.dump(t1,file)
```

## READ OBJECT FROM BINARY FILE

```
import pickle
with open("new.dat","rb") as file:
    while True:
        try:
            var=pickle.load(file)
            print(var)
        except EOFError:
            break
```

## READ A BINARY FILE WITH MULTIPLE OBJECTS

```
import pickle
with open("new.dat","rb") as file:
   list=pickle.load(file)
   tuple=pickle.load(file)
   list1=pickle.load(file)
print(list)
print(tuple)
print(list1)
```

KAPIL SEHGAL

## STRING WRITE READ AND APPEND IN BINARY FILE

### WRITE

```
import pickle
with open("new.dat","wb") as file:
    s1="This is Mumbai"
    s2="This is Delhi"
    s3="This is Bhopal"
    pickle.dump(s1,file)
    pickle.dump(s2,file)
    pickle.dump(s3,file)
```

### READ

```
import pickle
with open("new.dat","rb") as file:
    while True:
        try:
            str=pickle.load(file)
            print(str)
        except EOFError:
            break
```

### APPEND

```
import pickle
with open("new.dat","ab") as file:
    s1="This is Vidisha"
    s2="This is Jhansi"
    s3="This is Ujjain"
    pickle.dump(s1,file)
    pickle.dump(s2,file)
    pickle.dump(s3,file)
```

KAPIL SEHGAL

# SEARCH IN BINARY FILE

## WRITE A PROGRAM TO SEARCH OBJECT IN TO THE BINARY FILES

```
import pickle
rlist=[110,210,310,410,510]  # list for search
with open("new.dat","rb") as file:
    while True:
        try:
            list=pickle.load(file)
            if (list==rlist):
                print(list)
                print("Data Found ")
            else:
                print(list)
        except EOFError:
            break
```

KAPIL SEHGAL

## WRITE A PROGRAM TO SEARCH STRING IN BINARY FILE CONTAINS STRINGS.

```
import pickle
with open("new.dat","rb") as file:
    str=input("Enter String to Search ")
    flag=0
    while True:
        try:
            str1=pickle.load(file)
            if (str1==str):
                print ("Search Successful ")
                flag=1
                break
        except EOFError:
            break
    if flag==0:
        print("Search Unsuccessful")
```

## BINARY FILE PROGRAMS

### WRITE A PROGRAM TO READ A BINARY FILE "NEW.DAT" WHICH CONTAINS LIST AND TUPLE AND FIND THE SUM OF THE ELEMENTS OF LIST AND TUPLE.

```
import pickle
with open("new.dat","rb") as file:
    sum=0
    while True:
        try:
            var=pickle.load(file)
            for i in var:
                sum = sum + i
        except EOFError:
            break
    print("Sum of elements of list ",sum)
```

```
import pickle
with open("new.dat","wb") as file:
    l1=[10,20,30,40,50]
    l2=(60,70,80,90,100)
    l3=[110,120]
    l4=(130,140,150)
    pickle.dump(l1,file)
    pickle.dump(l2,file)
    pickle.dump(l3,file)
    pickle.dump(l4,file)
```

Output.

```
Sum of elements of list  410
```

# BINARY FILE PROGRAMS

## WRITE A PROGRAM TO READ A BINARY FILE "NEW.DAT" MODIFY THE SPECIFIC OBJECT

### CREATE

```
import pickle
with open("new.dat","wb") as file:
    s1="This is Vidisha"
    s2="This is Jhansi"
    s3="This is Ujjain"
    s4=[10,20,30,"Ram","Krishna"]
    s5=(50,60,70)
    s6={"Name":"Ram","Age":25}
    pickle.dump(s1,file)
    pickle.dump(s2,file)
    pickle.dump(s3,file)
    pickle.dump(s4,file)
    pickle.dump(s5,file)
    pickle.dump(s6,file)
```

### READ

```
import pickle
with open("new.dat","rb") as file:
    while True:
        try:
            str=pickle.load(file)
            print(str)
        except EOFError:
            break
```

### UPDATE

```
import pickle
record=[]
oldstr=input("Enter String to Replace ")
newstr=input("Enter New String ")
with open("new.dat","rb+") as file:
    while True:
        try:
            str=pickle.load(file)
            record.append(str)
        except EOFError:
            break
    count=-1
    for i in record:
            count=count+1
            if (i==oldstr):
                record[count]=newstr
    file.seek(0)
    for i in record:
        pickle.dump(i,file)
```

KAPIL SEHGAL

# BINARY FILE PROGRAMS

## WRITE A PROGRAM TO READ A BINARY FILE “NEW.DAT” MODIFY THE SPECIFIC OBJECT

### CREATE

```
import pickle
with open("new.dat","wb") as file:
    s1="This is Vidisha"
    s2="This is Jhansi"
    s3="This is Ujjain"
    pickle.dump(s1,file)
    pickle.dump(s2,file)
    pickle.dump(s3,file)
```

### READ

```
import pickle
with open("new.dat","rb") as file:
    while True:
        try:
            str=pickle.load(file)
            print(str)
        except EOFError:
            break
```

### UPDATE

```
import pickle
import os
oldobject="This is Vidisha"
newobject=["This","is","Vidisha"]
tfile=open("temp.dat","wb")
with open("new.dat","rb") as file:
    while True:
        try:
            objread=pickle.load(file)
            if (oldobject==objread):
                pickle.dump(newobject,tfile)
            else:
                pickle.dump(objread,tfile)
        except EOFError:
            break
tfile.close()
os.remove("new.dat")
os.rename("temp.dat","new.dat")
```

**KAPIL SEHGAL**

# BINARY FILE PROGRAMS

## WRITE A PROGRAM TO READ A BINARY FILE "NEW.DAT" DELETE THE SPECIFIC OBJECT

### CREATE

```
import pickle
with open("new.dat","wb") as file:
    s1="This is Vidisha"
    s2="This is Jhansi"
    s3="This is Ujjain"
    s4=[10,20,30,"Ram","Krishna"]
    s5=(50,60,70)
    s6={"Name":"Ram","Age":25}
    pickle.dump(s1,file)
    pickle.dump(s2,file)
    pickle.dump(s3,file)
    pickle.dump(s4,file)
    pickle.dump(s5,file)
    pickle.dump(s6,file)
```

### READ

```
import pickle
with open("new.dat","rb") as file:
    while True:
        try:
            str=pickle.load(file)
            print(str)
        except EOFError:
            break
```

### DELETE

```
import pickle
record=[]
t=(50,60,70)
with open("new.dat","rb+") as file:
    while True:
        try:
            str=pickle.load(file)
            record.append(str)
        except EOFError:
            break
    count=-1
    for i in record:
            count=count+1
            if (i==t):
                break
    del record[count]
    file.seek(0)
    for i in record:
        pickle.dump(i,file)
        file.truncate()
```

KAPIL SEHGAL

# BINARY FILE PROGRAMS

## WRITE A PROGRAM TO READ A BINARY FILE "NEW.DAT" DELETE THE SPECIFIC OBJECT

### CREATE

```
import pickle
with open("new.dat","wb") as file:
    s1="This is Vidisha"
    s2="This is Jhansi"
    s3="This is Ujjain"
    pickle.dump(s1,file)
    pickle.dump(s2,file)
    pickle.dump(s3,file)
```

### READ

```
import pickle
with open("new.dat","rb") as file:
    while True:
        try:
            str=pickle.load(file)
            print(str)
        except EOFError:
            break
```

### DELETE

```
import pickle
import os
oldobject="This is Jhansi"
tfile=open("temp.dat","wb")
with open("new.dat","rb") as file:
    while True:
        try:
            objread=pickle.load(file)
            if (oldobject!=objread):
                pickle.dump(objread,tfile)
        except EOFError:
            break

tfile.close()
os.remove("new.dat")
os.rename("temp.dat","new.dat")
```

KAPIL SEHGAL

# C.S.V. FILE

A **CSV** is a comma-separated values **file**, which allows data to be saved in a tabular format. A **CSV file** is a human readable text **file** where each line has a number of fields, separated by commas or some other delimiter. The **CSV** file is opened as a text file

## Reading data from C.S.V. File

(1) Import csv
(2) Get Data using Reader Method into object.
(3) Fetch data from object and display on the screen

## Methods for reading a file

(1) reader()
(2) DictReader()

## Methods for Writing into file

(1) writer() :
(2) writerow()
(3) writerows()
(4) DictWriter()
(5) writeheader()

## Creating C.S.V. File

(1) Import csv
(2) Get Data from keyboard or any other way
(3) Make a object to store data using writer() method
(4) Store the object using writerow() and writerows() methods

KAPIL SEHGAL

# METHODS OF C.S.V. FILE

## reader() method

- The csv.reader() is used to read the file, which returns an ittrable reader object.

**object= csv.reader(fileobject[,delimiter]) # by default “,”**

Let the file “firstcsv.txt” Contains

```
Name,Age
Sourabh,16
Abhishek,17
Krishna,12
Arnav,13
```

Note:-

**Elements of CSV file fetch in form of list # here row is a type of LIST &**
**Do not access directly object**

**KAPIL SEHGAL**

Read this csv file as a list

```
import csv
with open("firstcsv.csv","r") as file:
    obj=csv.reader(file,delimiter=’,’)
    for row in obj:
        print(row)
```

Output:-

```
['Name', 'Age']
['Sourabh', '16']
['Abhishek', '17']
['Krishna', '12']
['Arnav', '13']
```

Read this csv file as individual elements

```
import csv
with open("firstcsv.csv","r") as file:
    obj=csv.reader(file)
    for row in obj:
        print(row[0],",",row[1])
```

Output:-

```
Name , Age
Sourabh , 16
Abhishek , 17
Krishna , 12
Arnav , 13
```

## WRITE A PROGRAM TO READ A CSV FILE AND STORE DATA INTO SEPARATE LIST

Let the file “firstcsv.txt” Contains

```
Name,Age
Sourabh,16
Abhishek,17
Krishna,12
Arnav,13
```

```python
import csv
listnm=[]
listage=[]
with open("firstcsv.csv","r") as file:
    obj=csv.reader(file,delimiter=',')
    for row in obj:
        listnm.append(row[0])
        listage.append(row[1])
    print(listnm)
    print(listage)
```

Output:-

```
['Name', 'Sourabh', 'Abhishek', 'Krishna', 'Arnav']
['Age', '16', '17', '12', '13']
```

**KAPIL SEHGAL**

## METHODS OF C.S.V. FILE

### writer() method

To write to a CSV file in Python, we can use the csv.writer() function. The csv.writer() function returns a writer object that converts the user's data into a delimited string. This string can later be used to write into CSV files using the writerow() function.

**writerobject=csv.writer(fileobject)**

### writerrow() method

writerow() method will write object row into the csv file one by one.

**writerobject.writerow(row-of-object)**

### writerrows() method

writerows() method will write entire data object in one go into the csv file.

**writerobject.writerows(data-object)**

KAPIL SEHGAL

## WRITE A PROGRAM TO WRITE DATA INTO CSV FILE

Let the file "firstcsv.txt" Contains

```
Name,Age
Sourabh,16
Abhishek,17
Krishna,12
Arnav,13
```

### Using writerow() Method

```
import csv
list=[["Name","Age"],["Sourabh",16],["Abhishek",17],["Krishna",12],["Arnav",13]]
with open("secondcsv.csv","w") as file:
    obj=csv.writer(file)
    for row in list:
       obj.writerow(row)
```

### Using writerows() Method

```
import csv
list=[["Name1","Age"],["Sourabh",16],["Abhishek",17],["Krishna",12],["Arnav",13]]
with open("secondcsv.csv","w") as file:
    obj=csv.writer(file)
    obj.writerows(list)
```

**KAPIL SEHGAL**

# METHODS OF C.S.V. FILE

## DictReader() method

The csv.DictReader class operates like a regular reader but maps the information read into a dictionary. The keys for the dictionary can be passed in with the fieldnames parameter or inferred from the first row of the CSV file.

**DicReaderObject=csv.DictReader(fileobject [,fieldnames=filenamelist])**

### When CSV file contains heading

```
import csv
with open("secondcsv.csv","r") as file:
    obj=csv.DictReader(file)
    for row in obj:
        print(row)
print(row["Name"],",",row["Age"])
```

### When CSV file does not contains heading

```
import csv
with open("secondcsv.csv","r") as file:
    colname=['Name','Age']
    obj=csv.DictReader(file,fieldnames=colname)
    for row in obj:
        print(row)
        print(row["Name"],",",row["Age"])
```

KAPIL SEHGAL

## METHODS OF C.S.V. FILE

### DictWriter() method

csv.DictWriter writes the values from Python dictionaries into the CSV file.

**DicWriterObject=csv.DictWriter(fileobject [,fieldnames=filenamelist])**

### writeheader() method

The writeheader() method writes the headers to the CSV file. For ex. object.writeheader()

```
import csv
with open("newcsv.csv","w") as file:
    sdict={'Stname':'Krishna','Rollno':151}
    colname=['Stname','Rollno']
    cwriter=csv.DictWriter(file,fieldnames=colname)
    cwriter.writeheader()
    cwriter.writerow(sdict)
    cwriter.writerow({'Stname':'Ram','Rollno':150})
```

KAPIL SEHGAL