

**File Handling**

As the part of programming requirement, we have to store our data permanently for future purpose. For this requirement we should go for files.

Files are very common permanent storage areas to store our data.

## Types of Files:

There are 2 types of files

#### Text Files:

Usually we can use text files to store character data eg: abc.txt

#### Binary Files:

Usually we can use binary files to store binary data like images,video files, audio files etc...

## Opening a File:

Before performing any operation (like read or write) on the file,first we have to open that file.For this we should use Python's inbuilt function open()

But at the time of open, we have to specify mode,which represents the purpose of opening file.

#### f = open(filename, mode)

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 avaialble 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 avaialble 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.**
5. **a+  To append and read data from the file.It wont override existing data.**
6. **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.

Eg: rb,wb,ab,r+b,w+b,a+b,xb

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

We are opening abc.txt file for writing data.

## 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.

#### f.close()

**Various properties of File Object:**

Once we opend 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() Retruns boolean value indicates that whether file is readable or not writable() Returns boolean value indicates that whether file is writable or not.



Eg:

|  |
| --- |
| **1) f=open("abc.txt",'w')** |
| **2) print("File Name: ",f.name)** |
| **3) print("File Mode: ",f.mode)** |
| **4) print("Is File Readable: ",f.readable())** |
| **5) print("Is File Writable: ",f.writable())** |
| **6) print("Is File Closed : ",f.closed)** |
| **7) f.close()** |
| **8) print("Is File Closed : ",f.closed)** |
| **9)** |
| **10)** |
| **11) Output** |
| **12) D:\Python\_classes>py test.py** |
| **13) File Name: abc.txt** |
| **14) File Mode: w** |
| **15) Is File Readable: False** |
| **16) Is File Writable: True** |
| **17) Is File Closed : False** |
| **18) Is File Closed : True** |

**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)

Eg:

|  |  |
| --- | --- |
| **1)** | **f=open("abcd.txt",'w')** |
| **2)** | **f.write("Durga\n")** |
| **3)** | **f.write("Software\n")** |
| **4)** | **f.write("Solutions\n")** |
| **5)** | **print("Data written to the file successfully")** |
| 6) | **f.close()** |

**abcd.txt: Durga Software Solutions**

Note: In the above program, data present in the file will be overridden everytime 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")



Eg 2:

|  |
| --- |
| **1) f=open("abcd.txt",'w')** |
| **2) list=["sunny\n","bunny\n","vinny\n","chinny"]** |
| **3) f.writelines(list)** |
| **4) print("List of lines written to the file successfully")** |
| 5) **f.close()** |

**abcd.txt: sunny bunny vinny chinny**

Note: while writing data by using write() methods, compulsory we have to provide line seperator(\n),otherwise total data should 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

Eg 1: To read total data from the file

|  |
| --- |
| **1) f=open("abc.txt",'r')** |
| **2) data=f.read()** |
| **3) print(data)** |
| **4) f.close()** |
| **5)** |
| **6) Output** |
| **7) sunny** |
| **8) bunny** |
| **9) chinny** |
| **10) vinny** |

Eg 2: To read only first 10 characters:

|  |
| --- |
| **1) f=open("abc.txt",'r')** |
| **2) data=f.read(10)** |
| **3) print(data)** |
| **4) f.close()** |
| **5)** |



|  |  |
| --- | --- |
| **6)** | **Output** |
| **7)** | **sunny** |
| **8)** | **bunn** |

Eg 3: To read data line by line:

|  |
| --- |
| **1) f=open("abc.txt",'r')** |
| **2) line1=f.readline()** |
| **3) print(line1,end='')** |
| **4) line2=f.readline()** |
| **5) print(line2,end='')** |
| **6) line3=f.readline()** |
| **7) print(line3,end='')** |
| **8) f.close()** |
| **9)** |
| **10) Output** |
| **11) sunny** |
| **12) bunny** |
| **13) chinny** |

Eg 4: To read all lines into list:

|  |
| --- |
| **1) f=open("abc.txt",'r')** |
| **2) lines=f.readlines()** |
| **3) for line in lines:** |
| **4) print(line,end='')** |
| **5) f.close()** |
| **6)** |
| **7) Output** |
| **8) sunny** |
| **9) bunny** |
| **10) chinny** |
| **11) vinny** |

Eg 5:

|  |
| --- |
| **1) f=open("abc.txt","r")** |
| **2) print(f.read(3))** |
| **3) print(f.readline())** |
| **4) print(f.read(4))** |
| **5) print("Remaining data")** |
| **6) print(f.read())** |
| **7)** |
| **8) Output** |
| **9) sun** |
| **10) ny** |
| **11)** |
| **12) bunn** |
| **13) Remaining data** |



|  |
| --- |
| **14) y** |
| **15) chinny** |
| **16) vinny** |

# 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 closing of file,after completing all operations automatically even in the case of exceptions also, and we are not required to close explicitly.

Eg:

|  |
| --- |
| **1) with open("abc.txt","w") as f:** |
| **2) f.write("Durga\n")** |
| **3) f.write("Software\n")** |
| **4) f.write("Solutions\n")** |
| **5) print("Is File Closed: ",f.closed)** |
| **6) print("Is File Closed: ",f.closed)** |
| **7)** |
| **8) Output** |
| **9) Is File Closed: False** |
| **10) Is File Closed: True** |

#### The seek() and tell() methods:

**tell():**

==>We can use tell() method to return current position of the cursor(file pointer) from beginning of the file. [ can you plese telll current cursor position]

The position(index) of first character in files is zero just like string index. Eg:

|  |  |
| --- | --- |
| **1)** | **f=open("abc.txt","r")** |
| **2)** | **print(f.tell())** |
| **3)** | **print(f.read(2))** |
| **4)** | **print(f.tell())** |
| **5)** | **print(f.read(3))** |
| **6)** | **print(f.tell())** |

**abc.txt: sunny bunny chinny vinny**



#### Output:

0

su 2

nny 5

#### seek():

We can use seek() method to move cursor(file pointer) to specified location. [Can you please seek the cursor to a particular location]

f.seek(offset, fromwhere)

offset represents the number of positions

The allowed values for second attribute(from where) are 0---->From beginning of file(default value)

1---->From current position 2--->From end of the file

Note: Python 2 supports all 3 values but Python 3 supports only zero. Eg:

|  |
| --- |
| **1) data="All Students are STUPIDS"** |
| **2) f=open("abc.txt","w")** |
| **3) f.write(data)** |
| **4) with open("abc.txt","r+") as f:** |
| **5) text=f.read()** |
| **6) print(text)** |
| **7) print("The Current Cursor Position: ",f.tell())** |
| **8) f.seek(17)** |
| **9) print("The Current Cursor Position: ",f.tell())** |
| **10) f.write("GEMS!!!")** |
| **11) f.seek(0)** |
| **12) text=f.read()** |
| **13) print("Data After Modification:")** |
| **14) print(text)** |
| **15)** |
| **16) Output** |
| **17)** |
| **18) All Students are STUPIDS** |
| **19) The Current Cursor Position: 24** |
| **20) The Current Cursor Position: 17** |
| **21) Data After Modification:** |



**22) All Students are GEMS!!!**

#### How to check a particular file exists or not?

We can use os library to get information about files in our computer.

os module has path sub module,which contains isFile() function to check whether a particular file exists or not?

os.path.isfile(fname)

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

**1) import os,sys**

**3) if os.path.isfile(fname):**

**5) f=open(fname,"r")**

**7) print("File does not exist:",fname)**

**9) print("The content of file is:")**

**11) print(data)**

**13) Output**

**15) Enter File Name: durga.txt 17)**

**19) Enter File Name: abc.txt**

**21) The content of file is:**

**23) All Students are GEMS!!!**

**25) All Students are GEMS!!!**

**27) All Students are GEMS!!!**

**26) All Students are GEMS!!!**

**24) All Students are GEMS!!!**

**22) All Students are GEMS!!!**

**20) File exists: abc.txt**

**18) D:\Python\_classes>py test.py**

**16) File does not exist: durga.txt**

**14) D:\Python\_classes>py test.py**

**12)**

**10) data=f.read()**

**8) sys.exit(0)**

**6) else:**

**4) print("File exists:",fname)**

**2) fname=input("Enter File Name: ")**

**Note:**

sys.exit(0) ===>To exit system without executing rest of the program.

argument represents status code . 0 means normal termination and it is the default value.



#### Q. Program to print the number of lines,words and characters present in the given file?

|  |
| --- |
| **1) import os,sys** |
| **2) fname=input("Enter File Name: ")** |
| **3) if os.path.isfile(fname):** |
| **4) print("File exists:",fname)** |
| **5) f=open(fname,"r")** |
| **6) else:** |
| **7) print("File does not exist:",fname)** |
| **8) sys.exit(0)** |
| **9) lcount=wcount=ccount=0** |
| **10) for line in f:** |
| **11) lcount=lcount+1** |
| **12) ccount=ccount+len(line)** |
| **13) words=line.split()** |
| **14) wcount=wcount+len(words)** |
| **15) print("The number of Lines:",lcount)** |
| **16) print("The number of Words:",wcount)** |
| **17) print("The number of Characters:",ccount)** |
| **18)** |
| **19) Output** |
| **20) D:\Python\_classes>py test.py** |
| **21) Enter File Name: durga.txt** |
| **22) File does not exist: durga.txt** |
| **23)** |
| **24) D:\Python\_classes>py test.py** |
| **25) Enter File Name: abc.txt** |
| **26) File exists: abc.txt** |
| **27) The number of Lines: 6** |
| **28) The number of Words: 24** |
| 29) **The number of Characters: 149** |

**abc.txt:**

All Students are GEMS!!! All Students are GEMS!!! All Students are GEMS!!! All Students are GEMS!!! All Students are GEMS!!! All Students are GEMS!!!



**Handling Binary Data:**

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?

|  |  |
| --- | --- |
| **1)** | **f1=open("rossum.jpg","rb")** |
| **2)** | **f2=open("newpic.jpg","wb")** |
| **3)** | **bytes=f1.read()** |
| **4)** | **f2.write(bytes)** |
| **5)** | **print("New Image is available with the name: newpic.jpg")** |

**Handling csv files:**

CSV==>Comma seperated values

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

#### Writing data to csv file:

|  |
| --- |
| **1) import csv** |
| **2) with open("emp.csv","w",newline='') as f:** |
| **3) w=csv.writer(f) # returns csv writer object** |
| **4) w.writerow(["ENO","ENAME","ESAL","EADDR"])** |
| **5) n=int(input("Enter Number of Employees:"))** |
| **6) for i in range(n):** |
| **7) eno=input("Enter Employee No:")** |
| **8) ename=input("Enter Employee Name:")** |
| **9) esal=input("Enter Employee Salary:")** |
| **10) eaddr=input("Enter Employee Address:")** |
| **11) w.writerow([eno,ename,esal,eaddr])** |
| **12) 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:

|  |
| --- |
| **1) import csv** |
| **2) f=open("emp.csv",'r')** |
| **3) r=csv.reader(f) #returns csv reader object** |
| **4) data=list(r)** |
| **5) #print(data)** |
| **6) for line in data:** |
| **7) for word in line:** |
| **8) print(word,"\t",end='')** |
| **9) print()** |
| **10)** |
| **11) Output** |
| **12) D:\Python\_classes>py test.py** |
| **13) ENO ENAME ESAL EADDR** |
| **14) 100 Durga 1000 Hyd** |
| **15) 200 Sachin 2000 Mumbai** |
| **16) 300 Dhoni 3000 Ranchi** |

**Zipping and Unzipping Files:**

It is very common requirement to zip and unzip files. The main advantages are:

1. **To improve memory utilization**
2. **We can reduce transport time**
3. **We can improve performance.**

To perform zip and unzip operations, Python contains one in-bulit module zip file. This module contains a class : ZipFile

#### To create Zip file:

We have to create ZipFile class object with name of the zip file,mode and constant ZIP\_DEFLATED. This constant represents we are creating zip file.

#### f = ZipFile("files.zip","w","ZIP\_DEFLATED")

Once we create ZipFile object,we can add files by using write() method.

#### f.write(filename)



Eg:

|  |  |
| --- | --- |
| **1)** | **from zipfile import \*** |
| **2)** | **f=ZipFile("files.zip",'w',ZIP\_DEFLATED)** |
| **3)** | **f.write("file1.txt")** |
| **4)** | **f.write("file2.txt")** |
| **5)** | **f.write("file3.txt")** |
| **6)** | **f.close()** |
| **7)** | **print("files.zip file created successfully")** |

#### To perform unzip operation:

We have to create ZipFile object as follows

#### f = ZipFile("files.zip","r",ZIP\_STORED)

ZIP\_STORED represents unzip operation. This is default value and hence we are not required to specify.

Once we created ZipFile object for unzip operation,we can get all file names present in that zip file by using namelist() method.

#### names = f.namelist()

Eg:

|  |  |
| --- | --- |
| **1)** | **from zipfile import \*** |
| **2)** | **f=ZipFile("files.zip",'r',ZIP\_STORED)** |
| **3)** | **names=f.namelist()** |
| **4)** | **for name in names:** |
| **5)** | **print( "File Name: ",name)** |
| **6)** | **print("The Content of this file is:")** |
| **7)** | **f1=open(name,'r')** |
| **8)** | **print(f1.read())** |
| 9) | **print()** |

#### Working with Directories:

It is very common requirement to perform operations for directories like

1. **To know current working directory**
2. **To create a new directory**
3. **To remove an existing directory**
4. **To rename a directory**
5. **To list contents of the directory etc...**



To perform these operations,Python provides inbuilt module os,which contains several functions to perform directory related operations.

#### Q1. To Know Current Working Directory:

import os cwd=os.getcwd()

print("Current Working Directory:",cwd)

#### Q2. To create a sub directory in the current working directory:

import os os.mkdir("mysub")

print("mysub directory created in cwd")

#### Q3. To create a sub directory in mysub directory:

cwd

|-mysub

|-mysub2

import os os.mkdir("mysub/mysub2")

print("mysub2 created inside mysub")

Note: Assume mysub already present in cwd.

#### Q4. To create multiple directories like sub1 in that sub2 in that sub3:

import os os.makedirs("sub1/sub2/sub3")

print("sub1 and in that sub2 and in that sub3 directories created")

#### Q5. To remove a directory:

import os os.rmdir("mysub/mysub2") print("mysub2 directory deleted")

#### Q6. To remove multiple directories in the path:

import os os.removedirs("sub1/sub2/sub3")

print("All 3 directories sub1,sub2 and sub3 removed")



#### Q7. To rename a directory:

import os os.rename("mysub","newdir")

print("mysub directory renamed to newdir")

#### Q8. To know contents of directory:

os module provides listdir() to list out the contents of the specified directory. It won't display the contents of sub directory.

Eg:

|  |  |
| --- | --- |
| **1)** | **import os** |
| **2)** | **print(os.listdir("."))** |
| **3)** | |
| **4)** | **Output** |
| **5)** | **D:\Python\_classes>py test.py** |
| **6)** | **['abc.py', 'abc.txt', 'abcd.txt', 'com', 'demo.py', 'durgamath.py', 'emp.csv', '** |
| **7)** | **file1.txt', 'file2.txt', 'file3.txt', 'files.zip', 'log.txt', 'module1.py', 'myl** |
| **8)** | **og.txt', 'newdir', 'newpic.jpg', 'pack1', 'rossum.jpg', 'test.py', ' pycache '** |
| 9) | **]** |

The above program display contents of current working directory but not contents of sub directories.

If we want the contents of a directory including sub directories then we should go for walk() function.

#### Q9. To know contents of directory including sub directories:

We have to use walk() function

[Can you please walk in the directory so that we can aware all contents of that directory] os.walk(path,topdown=True,onerror=None,followlinks=False)

It returns an Iterator object whose contents can be displayed by using for loop

path-->Directory path. cwd means . topdown=True --->Travel from top to bottom

onerror=None --->on error detected which function has to execute. followlinks=True -->To visit directories pointed by symbolic links



Eg: To display all contents of Current working directory including sub directories:

|  |
| --- |
| **1) import os** |
| **2) for dirpath,dirnames,filenames in os.walk('.'):** |
| **3) print("Current Directory Path:",dirpath)** |
| **4) print("Directories:",dirnames)** |
| **5) print("Files:",filenames)** |
| **6) print()** |
| **7)** |
| **8)** |
| **9) Output** |
| **10) Current Directory Path: .** |
| **11) Directories: ['com', 'newdir', 'pack1', ' pycache ']** |
| **12) Files: ['abc.txt', 'abcd.txt', 'demo.py', 'durgamath.py', 'emp.csv', 'file1.txt'** |
| **13) , 'file2.txt', 'file3.txt', 'files.zip', 'log.txt', 'module1.py', 'mylog.txt', '** |
| **14) newpic.jpg', 'rossum.jpg', 'test.py']** |
| **15)** |
| **16) Current Directory Path: .\com** |
| **17) Directories: ['durgasoft', ' pycache ']** |
| **18) Files: ['module1.py', ' init .py']** |
| **19)** |
| **20) ...** |

Note: To display contents of particular directory,we have to provide that directory name as argument to walk() function.

os.walk("directoryname")

#### Q. What is the difference between listdir() and walk() functions?

In the case of listdir(), we will get contents of specified directory but not sub directory contents. But in the case of walk() function we will get contents of specified directory and its sub directories also.

#### Running Other programs from Python program:

os module contains system() function to run programs and commands. It is exactly same as system() function in C language.

os.system("commad string")

The argument is any command which is executing from DOS.

Eg: import os

os.system("dir \*.py") os.system("py abc.py")



#### How to get information about a File:

We can get statistics of a file like size, last accessed time,last modified time etc by using stat() function of os module.

stats = os.stat("abc.txt")

The statistics of a file includes the following parameters: st\_mode==>Protection Bits

st\_ino==>Inode number st\_dev===>device st\_nlink===>no of hard links st\_uid===>userid of owner st\_gid==>group id of owner st\_size===>size of file in bytes

st\_atime==>Time of most recent access st\_mtime==>Time of Most recent modification st\_ctime==> Time of Most recent meta data change

#### Note:

st\_atime, st\_mtime and st\_ctime returns the time as number of milli seconds since Jan 1st 1970 ,12:00AM. By using datetime module fromtimestamp() function,we can get exact date and time.

#### Q. To print all statistics of file abc.txt:

|  |  |
| --- | --- |
| **1)** | **import os** |
| **2)** | **stats=os.stat("abc.txt")** |
| **3)** | **print(stats)** |
| **4)** | |
| **5)** | **Output** |
| **6)** | **os.stat\_result(st\_mode=33206, st\_ino=844424930132788, st\_dev=2657980798, st\_nlin** |
| **7)** | **k=1, st\_uid=0, st\_gid=0, st\_size=22410, st\_atime=1505451446, st\_mtime=1505538999** |
| 8) | **, st\_ctime=1505451446)** |

**Q. To print specified properties:**

|  |  |
| --- | --- |
| **1)** | **import os** |
| **2)** | **from datetime import \*** |
| **3)** | **stats=os.stat("abc.txt")** |
| **4)** | **print("File Size in Bytes:",stats.st\_size)** |
| **5)** | **print("File Last Accessed Time:",datetime.fromtimestamp(stats.st\_atime))** |
| **6)** | **print("File Last Modified Time:",datetime.fromtimestamp(stats.st\_mtime))** |
| **7)** | |



|  |
| --- |
| **8) Output** |
| **9) File Size in Bytes: 22410** |
| **10) File Last Accessed Time: 2017-09-15 10:27:26.599490** |
| **11) File Last Modified Time: 2017-09-16 10:46:39.245394** |

# Pickling and Unpickling of Objects:

Sometimes we have to write total state of object to the file and we have to read total object from the file.

The process of writing state of object to the file is called pickling and the process of reading state of an object from the file is called unpickling.

We can implement pickling and unpickling by using pickle module of Python. pickle module contains dump() function to perform pickling.

pickle.dump(object,file)

pickle module contains load() function to perform unpickling

obj=pickle.load(file)

**pickling**

**e1**

**eno: 100**

**ename: Durga esal: 10000 eaddr: HYD**

**pickle.dump**

**(e1, f)**

**eno: 100**

**ename: Durga esal: 10000 eaddr: HYD**

**eno: 100**

**ename: Durga esal: 10000 eaddr: HYD**

**abc.txt**

**pickle load (f)**

**unpickling**

**e2**



#### Writing and Reading State of object by using pickle Module:

|  |  |
| --- | --- |
| **1) import pickle** | |
| **2) class Employee:** | |
| **3)** | **def init (self,eno,ename,esal,eaddr):** |
| **4)** | **self.eno=eno;** |
| **5)** | **self.ename=ename;** |
| **6)** | **self.esal=esal;** |
| **7)** | **self.eaddr=eaddr;** |
| **8)** | **def display(self):** |
| **9)** | **print(self.eno,"\t",self.ename,"\t",self.esal,"\t",self.eaddr)** |
| **10) with open("emp.dat","wb") as f:** | |
| **11) e=Employee(100,"Durga",1000,"Hyd")** | |
| **12)** | **pickle.dump(e,f)** |
| **13) print("Pickling of Employee Object completed...")** | |
| 14) | |
| **15) with open("emp.dat","rb") as f:** | |
| **16) obj=pickle.load(f)** | |
| **17) print("Printing Employee Information after unpickling")** | |
| **18)** | **obj.display()** |

**Writing Multiple Employee Objects to the file:**

**emp.py:**

|  |  |
| --- | --- |
| **1)** | **class Employee:** |
| **2)** | **def init (self,eno,ename,esal,eaddr):** |
| **3)** | **self.eno=eno;** |
| **4)** | **self.ename=ename;** |
| **5)** | **self.esal=esal;** |
| **6)**  **7)** | **self.eaddr=eaddr;**  **def display(self):** |
| **8)** | |
| **9)** | **print(self.eno,"\t",self.ename,"\t",self.esal,"\t",self.eaddr)** |

**pick.py:**

|  |
| --- |
| **1) import emp,pickle** |
| **2) f=open("emp.dat","wb")** |
| **3) n=int(input("Enter The number of Employees:"))** |
| **4) for i in range(n):** |
| **5) eno=int(input("Enter Employee Number:"))** |
| **6) ename=input("Enter Employee Name:")** |
| **7) esal=float(input("Enter Employee Salary:"))** |
| **8) eaddr=input("Enter Employee Address:")** |
| **9) e=emp.Employee(eno,ename,esal,eaddr)** |
| **10) pickle.dump(e,f)** |
| **11) print("Employee Objects pickled successfully")** |



#### unpick.py:

|  |
| --- |
| **1) import emp,pickle** |
| **2) f=open("emp.dat","rb")** |
| **3) print("Employee Details:")** |
| **4) while True:** |
| **5) try:** |
| **6) obj=pickle.load(f)** |
| **7) obj.display()** |
| **8) except EOFError:** |
| **9) print("All employees Completed")** |
| **10) break** |
| **11) f.close()** |