## Recap

| | |
|---|---|
| 1. Text file each line end with | a) Stored as characters |
| 2. To store jpg, pdf we need | b) Text file |
| 3. Character to mark end of line | c) Binary File |
| 4. Text file input is | d) \n |
| 5. read() | e) To know the position of file pointer |
| 6. writelines() | f) To move the file pointer |
| 7. tell() | g) to write sequence in file |
| 8. seek() | h) to read entire contents of file |

## Recap

1. Open a text file sample.txt in read mode
2. What is the difference between read() and read(n)
3. Open sample.txt to write contents and write ‘Today is cloudy’ and next line as ‘It is raining here’
4. Why is it necessary to close a file?
5. Which module do we use to work with binary file?
6. State any two differences between binary and text file.
7. What is the difference between read() and readlines()?
8. What does seek() and tell() methods do in file ?

## FILE HANDLING - BINARY FILES

INTRODUCTION

NEED FOR PICKLE MODULE?

HOW TO USE PICKLE MODULE ?

PICKLE.DUMP()

PICKLE.LOAD()

FILE MODES

FILE OPERATIONS IN BINARY FILE

INSERTING/APPENDING TO BINARY FILE

READING FROM BINARY FILE

SEARCHING IN BINARY FILE

UPDATION IN BINARY FILE

FILE POSITIONS

## INTRODUCTION

Files in the format :

Document files - .doc .xls
Video files - .mpg .avi
Picture files - .gif .jpg
Audio files - .mp3 .wav
Executable files - .exe .dll
Database files - .mdb .sqlite
Archive files - .zip .rar

- These files are all stored in computers in the form of binary.
- Contents of these binary files if opened in text editor, the contents will not be in human-readable form.
- This is because they are stored in binary format which can be read only by the computer.
- Moreover these binary files does not require EOL like text files, and no translation is required.
- They are fast and easy in working than compared to Text files.

## NEED FOR PICKLE MODULE

- To store and read structures like list, dictionary to a file, pickle module of Python is used.
- Pickling refers to the process of converting the structure to a byte stream before writing to the file.(Writing into file)
- Unpickling is used to convert the byte stream back to the original structure.(Reading from file)
- Generally file works with string parameters. To work on binary file, conversion of data at the time of reading as well as writing is required.
- Pickle module can be used to store any kind of object(list, dictionary…) in a file.

## HOW TO USE PICKLE MODULE?

- First we need to import pickle module which can be done by the command

  import pickle

- Pickle module provides two main methods

  dump()

  load()

- For binary file creation/writing we use:

  pickle.dump()

- For binary file reading/accessing we use :

  pickle.load()

## pickle.dump()

**Syntax :**

```
dump(object,fileobject)
```

This helps to store the contents of the object(list, dictionary…) into the binary file.

## pickle.load()

**Syntax** :

object = load(fileobject)

This helps to read data from the binary file into the object(list, dictionary…) specified on the left side of the assignment operator.

# OPEN A FILE(S)

File modes for opening in Binary files

| Mode | Description |
|---|---|
| rb | Opens a file for reading in binary format. The file handle is placed at the beginning of the file. This is the default mode. FileNotFoundError will be generated if the file does not exist. |
| wb | Opens a file for writing in binary format. Overwrites the file if the file exists. Creates a new file if does not exists. |
| ab | Opens a file for appending in binary format. The file handle/pointer is at the end of file if it exists. If the file does not exists, it creates a new file. |
| rb+ | Opens a file for both reading and writing in binary format. The file pointer is placed at the beginning of the file. |
| wb+ | Opens a file for both reading and writing in binary format. Overwrites the existing file if the file exists. If the file does not exists, creates a new file for reading and writing. |
| ab+ | Opens a file for both appending and reading in binary format.The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |

# FILE OPERATIONS USING BINARY FILES

- Inserting/Appending data in binary file
  - Adding new record in the file in beginning or end of the file.
- Read data from binary file
  - Opening the file in read mode and displaying the data
- Search a record in binary file
  - Given a record key (like rollnumber..) the record details are searched and displayed
- Update a record in binary file
  - Searching by record key and changing the record with new values.

## INSERTING / CREATING TO FILES

**Eg 1: (rjk29061.py)**

```
#creating a binary file to store employee details
# empno,name,desig,salary as fields
import pickle
def binary_insertion(fname,fmode):
    fhandle = open(fname,fmode)
    emprec = []
    while True:
        empid=int(input('Enter employee id '))
        empname = input('Enter the employee name ')
        desig = input('Enter the designation ')
        salary = float(input('Enter salary amount '))
        data =[empid,empname,desig,salary]
```

# INSERTING / CREATING TO FILES

```
        emprec.append(data)
        choice = input(' Enter Y to continue ')
        if choice == 'Y':
            continue
        else:
            break
    pickle.dump(emprec,fhandle)
    fhandle.close()

fname = input('Enter file name ')
fmode = 'wb'
binary_insertion(fname,fmode)
```

## READING FROM FILES(rjk29062.py)

```
import pickle
def binary_reading(fname,fmode):
    fhandle = open(fname,fmode)
    emprec = pickle.load(fhandle)
    print('Employee details \n')
    for rec in emprec:
        empid=rec[0]
        empname = rec[1]
        desig = rec[2]
        salary = rec[3]
```

## READING FROM FILES

```
        print('\nEmployee ID ',empid)
        print('\nEmployee Name ',empname)
        print('\nDesignation ',desig)
        print('\nSalary ',salary)
        print()
    fhandle.close()

fname = input('Enter file name ')
fmode = 'rb'
binary_reading(fname,fmode)
```

## SEARCHING IN BINARY FILES(rjk29063.py

```
import pickle
def binary_reading(fname,fmode):
    fhandle = open(fname,fmode)
    emprec = pickle.load(fhandle)
    empid = int(input(‘Emp id to search '))
    print('Employee details \n')
    flag = 0
    for rec in emprec:
        if empid == rec[0]:
            empname = rec[1]
            desig = rec[2]
            salary = rec[3]
            print('\nEmployee ID ',empid)
            print('\nEmployee Name ',empname)
            print('\nDesignation ',desig)
            print('\nSalary ',salary)
            flag = 1
            break
    if flag == 0:
        print('Record not found')
    fhandle.close()

fname = input('Enter file name ')
fmode = 'rb'
binary_reading(fname,fmode)
```

## UPDATING IN BINARY FILES(rjk29064.py)

```
import pickle
def binary_reading(fname,fmode):
    fhandle = open(fname,fmode)
    emprec = pickle.load(fhandle)
    empid = int(input('Empid to update '))
    flag = 0
    for rec in emprec:
        if empid == rec[0]:
            empname = rec[1]
            desig = rec[2]
            print('Employee name ',empname,'Designation ',desig)
            desig = input('Enter new desig ')
            rec[2] = desig
            flag = 1
            break
```

## UPDATING IN BINARY FILES

```
    if flag == 0:
        print('Record not found')
    else:
        fhandle.seek(0) # moving record pointer to beginning
        pickle.dump(emprec,fhandle)
    fhandle.close()

fname = input('Enter file name ')
fmode = 'rb+'
binary_reading(fname,fmode)
```

## FILE POSITIONS

- File pointer is associated with every file by the file management system, that facilitates the movement across the file for reading and/or writing data.
- The file pointer specifies a location from where the current read or write operation is initiated.
- Once a read/write operation is completed the file pointer is automatically updated.

```
# text file handling updation(rjk29065.py)
fhandle = open('sample.txt','w')
fhandle.write('Mere India Mahaan'+'\n')
fhandle.write('Is India self sufficient'+'\n')
fhandle.close()
# text file updation
fhandle = open('sample.txt','r+')
new =''
x1 = fhandle.read()
x2 = x1.split(' ')

for i in x2:
    if i == 'India':

        new +='Bharat '
    else:
        new +=i+' '
print(new)
fhandle.seek(0)
fhandle.write(new)
fhandle.close()
```

## FILE POSITIONS

Methods in Python :

- tell()
- seek( )

tell() – tells the current position within the file at which the next read or write operation will occur.

It is specified as number of bytes from the beginning of the file.

Generally when the file is opened the file pointer is positioned at location 0, which is the beginning of the file

## FILE POSITIONS

seek(offset[,from]) – method is used to set the position of the file pointer or move the file pointer to a new location.

The **offset** indicates the number of bytes to be moved and the **from** argument specifies the reference position from where the bytes are to be moved.

from is important when we need to randomly move file pointer to required position.

## FILE POSITIONS

from and its position

| from | Reference position |
|---|---|
| 0 | From the beginning of the file |
| 1 | From the current position of the file |
| 2 | From the end of the file |

## FILE POSITIONS

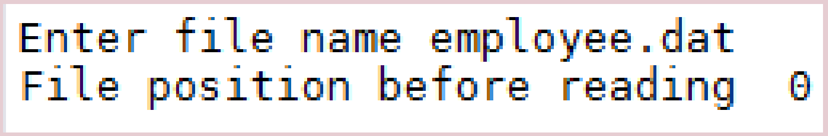
Write a program that tells the position of the file pointer.

**Program**

```
import pickle
def binary_reading(fname,fmode):
    fhandle = open(fname,fmode)
    print('File position before reading ',fhandle.tell())

fname = input('Enter file name ')
fmode = 'rb+'
binary_reading(fname,fmode)
```

Output



```
Enter file name employee.dat
File position before reading  0
```

# FILE POSITIONS(rjk0507.py)

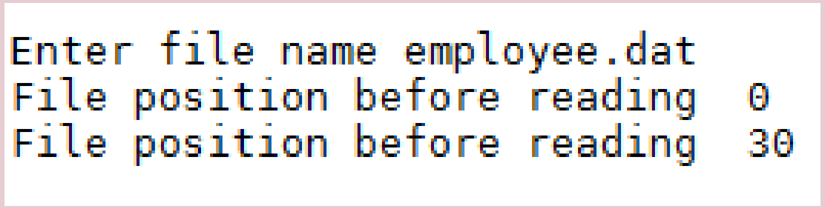
Write a program that sets the position of the file pointer to a specified location.

**Program**

```
import pickle
def binary_reading(fname,fmode):
    fhandle = open(fname,fmode)
    print('File position before reading ',fhandle.tell())
    emprec = pickle.load(fhandle)
    fhandle.seek(30,0)
    print('File position before reading ',fhandle.tell())
    fhandle.close()

fname = input('Enter file name ')
fmode = 'rb+'
binary_reading(fname,fmode)
```

Output



```
Enter file name employee.dat
File position before reading  0
File position before reading  30
```

## PRACTICAL 7

Write a program to use a binary file to store details of student – rollno, name, marks.  The program should create the file and also allow the user to update a particular record with new marks.

## PRACTICAL 7 A program to create binary file for students details and allow to update

```
# Practical problem 7(rjk02071.py)
import pickle
def binary_creation(fname,fmode):
    fhandle = open(fname,fmode)
    sturec = []
    while True:
        stuid=int(input('Enter Rollno '))
        stuname = input('Enter the Student name ')
        marks = float(input('Enter the Marks '))
        data =[stuid,stuname,marks]
        sturec.append(data)
```

## PRACTICAL 7

```
        choice = input(' Enter Y to continue ')
        if choice == 'Y':
            continue
        else:
            break
    pickle.dump(sturec,fhandle)
    fhandle.close()

fname = input('Enter file name ')
fmode = 'wb'
binary_creation(fname,fmode)
```

# PRACTICAL 7

```
def binary_updation(fname,fmode):
    fhandle = open(fname,fmode)
    sturec = pickle.load(fhandle)
    stuid = int(input('Rollno to update '))
    flag = 0
    for rec in sturec:
        if stuid == rec[0]:
            stuname = rec[1]
            marks = rec[2]
            print('Student name ',stuname,'Marks ',marks)
            marks = float(input('Enter updated marks '))
            rec[2] = marks
            flag = 1
            break
```

## PRACTICAL 7

```
    if flag == 0:
        print('Record not found')
    else:
        fhandle.seek(0) # moving file pointer to beginning
        pickle.dump(sturec,fhandle)
    fhandle.close()

fname = input('Enter file name ')
fmode = 'rb+'
binary_updation(fname,fmode)
```

## PRACTICAL 6

Write a program to use a binary file to search for rollno of a student and display the details of student otherwise display ‘record not found’ as message.

```
#Practical 6(rjk02072.py)
def binary_searching(fname,fmode):
    fhandle = open(fname,fmode)
    sturec = pickle.load(fhandle)
    stuid = int(input('Rollno to search '))
    print('Student details \n')
    flag = 0
    for rec in sturec:
        if stuid == rec[0]:
            stuname = rec[1]
            marks = rec[2]
            print('\nStudent ID ',stuid)
            print('\nStudent Name ',stuname)
            print('\nMarks ',marks)
            flag = 1
            break
```

## #Practical 6

```
  if flag == 0:
        print('Record not found')
    fhandle.close()

fname = input('Enter file name ')
fmode = 'rb'
binary_searching(fname,fmode)
```