BINARY FILES IN PYTHON (Read, Write and Append)

INTRODUCTION

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

Examples of binary files include

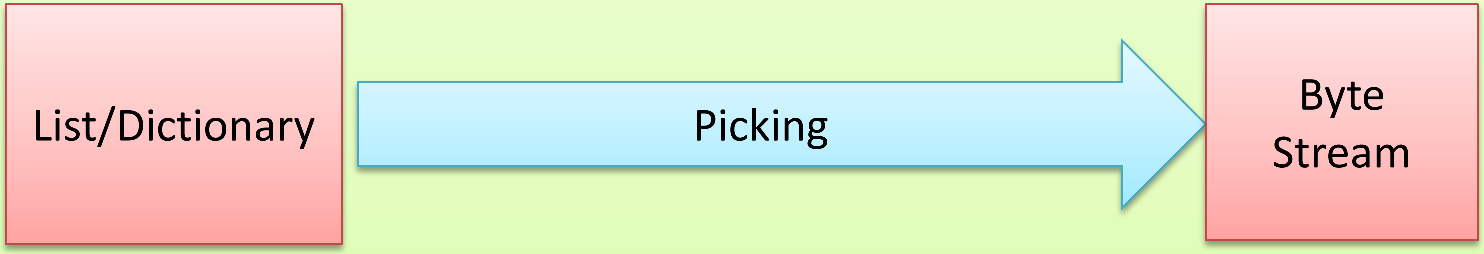
- Image files such as .jpg, .png, .bmp etc
- Audio files such as .mp3, .wav etc
- Video files such as .mp4, .avi etc
- Archive files such as .zip, .rar
- Executable files .exe, .dll

Characteristics

- Binary file is present in machine readable format.
- It is encoded in binary format in the same way as it is stored in computer memory
- There is no delimiter to end a line.
- No translation is required.
- It is easy and faster to work
- It is highly secured

PICKLING

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



List/Dictionary

Picking

Byte Stream

PICKLE MODULE

PICKLE module has to be imported before reading and writing data from and to a binary file

Syntax:

```
import pickle
```

File access modes

- rb – read only –
- wb – write only –
- ab - append
- rb+ - read and write
- wb+ - write and read
- ab+ - append and read

Pickle - dump()

Syntax: pickle.dump(structure, fileobj)

Used to write the object in a file

Program to write data into a binary file

```
import pickle
def bwrite():
    F= open("bfile.dat","wb")
    L= ["1.Python", "2.Java","3.C++","4.Oracle"]
    pickle.dump(L,F)
    F.close()

bwrite()
print("Data returned successfully")
```

Pickle – load()

Used to read data from file

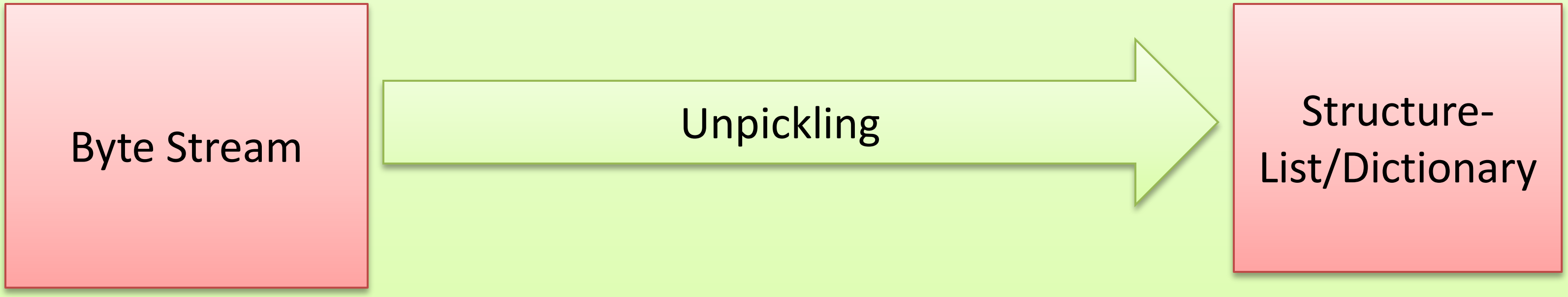
Syntax: structure = pickle.load(Fileobj)

Structure can be a list or dictionary

File handle is the file handle of the file

UNPICKLING

Process of converting the byte stream back to the original structure is called as Unpickling.



Byte Stream

Unpickling

Structure-
List/Dictionary

Example

```
import pickle
def bwrite():
    F= open("bfile.dat","wb")
    L= ["1.Python", "2.Java","3.C++","4.Oracle"]
    pickle.dump(L,F)
    F.close()


def bread():
    F= open("bfile.dat","rb")
    list = pickle.load(F)
    print(list)
    F.close()

bwrite()
print("Data returned successfully")

bread()
print("Data read successfully")
```

OPERATIONS PERFORMED IN A BINARY FILE

- Read records from a binary file
- Write record in a binary file
- Append record in a binary file
- Update record in a binary file
- Search records from a binary file

NESTED LIST

List inside another list.

Eg: [[7201, “Abhishek”,85], [7201, “Balaram”,90]]

Example program to write to a binary file

```
import pickle
def Binarywrite():
    B = open("studrec.dat","wb")
    stud = []
    while True:
        rno=int(input("Enter the rollnumber"))
        name = input("Enter the name")
        mark = int(input("Enter the mark"))
        data = [rno,name,mark]
        stud.append(data)
        ch = input("Do you want to enter more records? (y/n)")
        if ch=='n':
            break
    pickle.dump(stud,B)

Binarywrite()
print("Records written in the file successfully")
```

Example program to read from a binary file

```
import pickle

def Binaryread():
    B= open("studrec.dat", "rb")
    stud= pickle.load(B)
    print("Contents of binary file ")
    for R in stud:
        print(R)
        '''Rno = R[0]
        Rname= R[1]
        Rmark= R[2]
        print(Rno, Rname, Rmark)'''
    B.close()

Binaryread()
```

Example program to append to a binary file

```
import pickle
def Binarywrite():
    B = open("studrec.dat",“ab")
    stud = []
    while True:
        rno=int(input("Enter the rollnumber"))
        name = input("Enter the name")
        mark = int(input("Enter the mark"))
        data = [rno,name,mark]
        stud.append(data)
        ch = input("Do you want to enter more records? (y/n)")
        if ch=='n':
            break
    pickle.dump(stud,B)

Binarywrite()
print("Records written in the file successfully")
```

Try and except method

After appending, when you try to read records from a binary file, you will not have the appended records shown in the output.

To resolve this, we have to use try and except method

Example

```
import pickle

def Binaryread():
    B= open("studrec.dat", "rb")
    print("Contents of binary file ")
    while True:
        try:
            stud= pickle.load(B)
            for R in stud:
                print(R)
        except EOFError:
                 break
B.close()

Binaryread()
```