BINARY FILES

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 Files

- We can open some binary files in the normal text editor but we **can't read the content** present inside the file.
- That's because all the binary files will be encoded in the binary format, which can be **understood only by a computer or a machine.**
- In binary files, 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.

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 we want 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 we want to write.

Operations performed on Binary File

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

How to write records in a Binary File?

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

How to read records from a Binary File?

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

How to search records from a Binary File?

```
def search():
    f=open("BinaryRecord.dat",'rb')
    s=pickle.load(f)
    found=0
    rno=int(input("Enter roll no:"))
    for i in s:
        if i[0]==rno:
            print("Record Found...")
            print(i[0],i[1],i[2])
            found=1
    if found==0:
        print("Record not found...")
```

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_what" 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.