File Handling

Tuesday, August 5, 2025 7:26 PM

- File: It is a container that holds the data.
- Based on extensions we can classify that our file belongs to which category.
- File handling is the phenomenon of reading the data from the file or writing the data into the file.
- Before performing the operation, accessibility of the file is required.
- Open(): It is the function that is used to provide the accessibility.

  Syntax:
  Var=open(filename.ext/location, mode)
  #passing mode is optional, by default it is in read mode.

- Modes are classified into 3 types:
  i. Write
     > Write(w)
     > Write + read(w+)
     > Write binary(wb)
     > Write + read binary(wb+)
  ii. Read
     > Read(r)
     > Read + write(r+)
     > Read binary(rb)
     > Read + write binary(rb+)
  iii. Append
     > Append(a)
     > Append + read(a+)
     > Append binary(ab)
     > Append + read binary(ab+)

- W R T .txt file:
  1. Write(w): Writing in the file can be done with 2 functions.\
     i. Write(): used to write single data/1 line.
        Var.write('data')
     ii. Writelines(): used to write multiple data of lines.
        Var.writelines([d1, d2, d3…..])

     Eg:
     l=open('file1.txt','w')
     l.writelines(['good morning/n','good evening/n'])
     l.close()

- If we don't want to close the file, then the syntax that is used is:
  With open('filename.txt/loc','mode') as var:

  Eg:

```
with open('file2.txt','w') as l:
   l.write('Hii guys')
```

- If we want to use the path the r (raw string) is used.
  Var=open(r'filename……loc','mode')

2. Read(r): Used to read data from the file.
    i. Read(): It is used to read everything from file
        Res=var.read()
    ii. Readline(): It is used to read the first line.
        Res=var.readline()
    iii. Readlines(): Read all the data but gives the output in list
        Res=var.readlines()

Eg:

```
'''with open('file3.txt','r') as l:
   res=l.read()
   print(res)

with open('file3.txt','r') as l:
   res=l.readlines()
   print(res)
```

3. Append(): It is also used to write.
   Difference between write and append: If working on existing file, if write function is used it will remove the existing data and then write the new data but if append is used, it will continue writing from the last existing data.
    i. Write(): used for writing one line
        Var.write('data')
    ii. Writelines(): used for writing multiple lines
        Var.writelines([d1, d2, d3…..])

- **WORKING W R T .csv file:**
    The file with .csv extension.
    - Csv: Comma Separated Value

Eg:
Sname, id, marks
'A',1,75
'B',2,97
In today time, the data that is getting shared is in form of csv file only.

1. Write():
    i. Writerow(): It is used to write a single row.
    ii. Writerows(): It is used to write multiple rows.

Syntax:

```
Import csv
With open(file.csv/loc,'w') as var:
    Var1=csv.writer(var)
    Var1.writerow([val1, val2, val3….. Val n])
        Or
    Var1.writerows([[val1, val2, val3…],[val1, val2, val3…..],……]])
```

Eg:

```
import csv
with open('file12.csv','w') as a:
```

```
b=csv.writer(a)
b.writerow(['Ename','eid','sal','des'])
b.writerows([['A','TCS123',10000,'SE'],['B','TCS345',15000,'SSE']])
```

2. Read():

   Syntax:

   Import csv

   With open('filename.csv/loc','r') as var:

   Var1=csv.reader(var)

   Print(datatype(var1))

   Eg:

```
with open('file12.csv','r') as a:
  b=csv.reader(a)
  print(list(b))
      #or
  b=[i for i in b if i!=[]][1:]
  print(b)
```

3. Append():
   i. Writerow(): It is used to write a single row.
   ii. Writerows(): It is used to write multiple rows.

Syntax:

Import csv

With open(file.csv/loc,'a') as var:

Var1=csv.writer(var)

Var1.writerow([val1, val2, val3….. Val n])

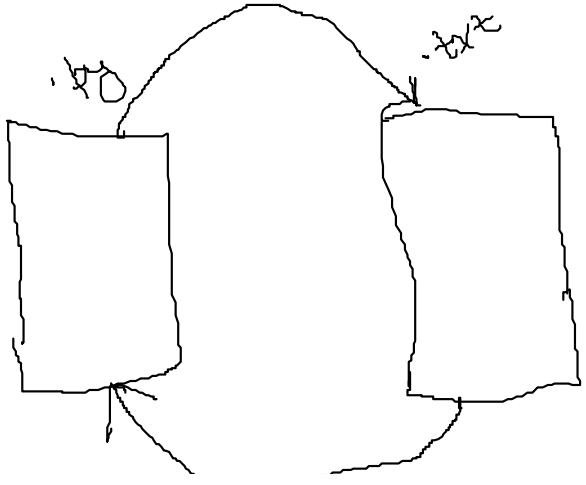
Or

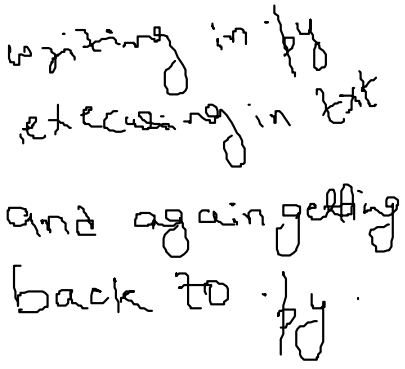
Var1.writerows([[val1, val2, val3…],[val1, val2, val3…..],……]])

Eg:

```
import csv
with open('file12.csv','a') as a:
  b=csv.writer(a)
  b.writerow(['Ename','eid','sal','des'])
  b.writerows([['A','TCS123',10000,'SE'],['B','TCS345',15000,'SSE']])
```

Parsing Technique



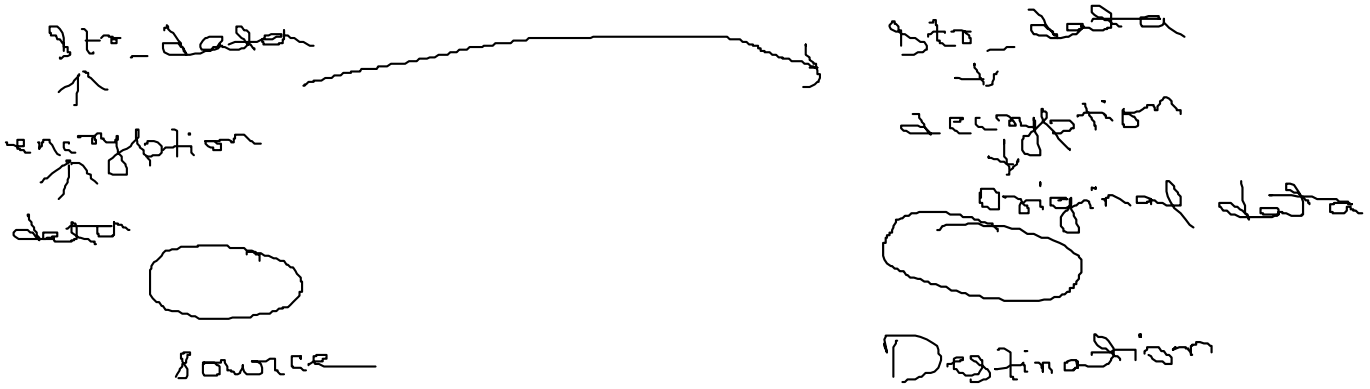


It is the phenomenon of providing the security to the data while sharing from one file to other file.

There are 2 types of parsing techniques:

i. Json -> java script object notation
ii. pickle

i. Json: It is a parsing technique where the original data gets converted into str and then the sharing process happens and again after reaching the destination, it gets converted to its original datatype.



We are converting the data into str because it restricts addition of new data, deletion of old data as well as accessibility with help of indexing which is not possible in any other datatypes.

Syntax:
Sender:
Import json
Var=json.dumps(data)

Reciever:
Import json
Var=json.loads(str_data)

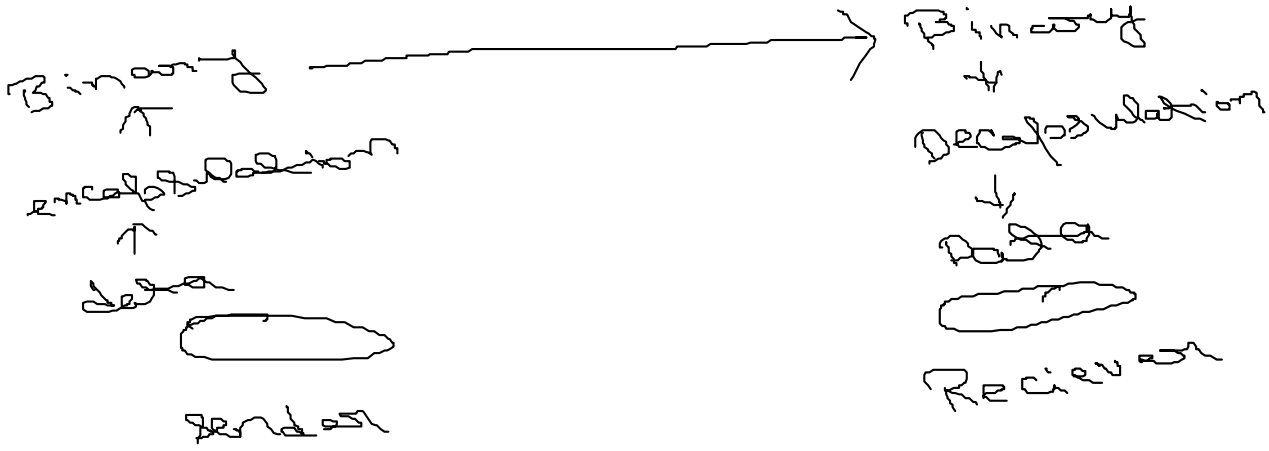
Eg:

```
import json
data=[10,20,30,40]
enc_data=json.dumps(data)
with open('file6.py','w') as a:
   a.write(enc_data)
   print(type(enc_data))


import json
with open('file6.py','r') as a:
```

```
res=a.read()
data=json.loads(res)
print(data)
print(type(data))
print(res)
print(type(res))
```

ii. Pickle: Here it converts the original data into binary form for sharing.



Syntax:
Sender:
Import pickle
Var=pickle.dumps(data)

Reciever:
Import pickle
Res=pickle.loads(bin_data)

Eg:

```
import pickle
data=[10,20,30,40]
enc_data=pickle.dumps(data)
with open('file7.txt','wb') as a:
    a.write(enc_data)
    print(type(enc_data))

import pickle
with open('file7.txt','rb') as a:
    enc_data=a.read()
    data=pickle.loads(enc_data)
    print(data)
```

Mostly we use json but when it comes to data related to transactions or finance then we use pickle.