Date 21/12/21

Chapter → 10

File Processing

Introduction :-

भविष्य मे प्रयोग के लिए डेटा को मेमोरी मे सेव करने के लिए प्रयोग की जाने वाली प्रक्रिया File processing कहलाती है।

पायथन दो प्रकार की फाइल प्रोसेसिंग Support करता है-

1. Text File
2. Binary File

(1.) **Text File** → यह फाइल ASCII या Unicode Char. का प्रयोग करके information को Simple text के रूप मे ~~सभी~~ स्टोर करती है।

Text File मे प्रत्येक लाइन के अंत मे कैरेक्टर का प्रयोग किया जाता है। ~~all cha~~ EOL '\n'

यह फाइल किसी भी सॉफ्टवेयर जैसे- Notepad, Wordpad, Libre Office, MS-word आदि पर open की जा सकती है।

(i) **Open()** → Text File के साथ work स्टार्ट करने से पहले इसे एक विशेष मोड मे ऑपन करने की आवश्यकता होती है।

Open Function का प्रयोग फाइल को ऑपन करने के लिए किया जाता है और यह दो argument इनपुट करता है।

Syntax: <file_handle> = open ('<file name>', '<mode>')

- File handle → यह एक identifier है जिसके द्वारा फाइल को सुरक्षित रखा जाता है। इसमे डेटा इनपुट किया जाता है तथा डेटा प्राप्त किया जाता है। File से

- File name → यह फाइल का नाम है जिसकी extension होती है।
.txt

- Mode → यह फाइल ऑपन करने के उद्देश्य को निर्धारित करता है।
~~फाई~~ Mode ६: प्रकार के होते हैं —

w	write	यह मोड फाइल मे डेटा लिखने के लिए फाइल को करता है यदि फाइल डिस्क मे है open तब यह उस फाइल को हटाकर नई फाइल बनाता है। यदि फाइल डिस्क में नही है तब भी नई फाइल बनाता है।
r	Read	इस मोड का उद्देश्य फाइल से डेटा प्राप्त करना है। यदि फाइल डिस्क में नही है तो error print करता है। यदि फाइल डिस्क मे है तब इसके डेटा को read किया जा सकता है।
a	append	फाइल में डेटा जोडने के लिए इसका प्रयोग होता है। यदि फाइल डिस्क में है तब उसमें डेटा जोडा जा सकता है यदि नही है तब नई फाइल बनायी जाती है।
w+	write & read	इसका प्रयोग फाइल में डेटा लिखने तथा इसके डेटा को प्राप्त करने के लिए किया जाता है। यदि फाइल म डिस्क मे मौजूद है तो पहली फाइल डिलीट करके नई फाइल बनायेगा। और यदि फाइल नही है तो नई फाइल बनाएगा।

r+	read & write	इस मोड का उद्देश्य फाइल से डेटा प्राप्त करना तथा इसमें डेटा लिखना है। यदि फाइल डिस्क में है तो read, write कार्य किया जा सकता है और यदि नही है तो error message प्रिन्ट होगा।
a+	append & read	यह दोनो मोड में read, write फाइल को open करता है। यदि फाइल डिस्क मे मौजूद है तब उसमें डेटा add कर सकते है यदि नही है तब नई फाइल बनायी जायेगी।

Example: student = open('marks.txt','w')
shop = open('product')

NOTE: यदि मोड़ नही दिया गया है तब default mode r (read) रहता है।

(ii) close() → फाइल बनाने के बाद फाइल मे सभी कार्य करने पर इस फाइल को close करना आवश्यक है। close Function के द्वारा File close की जाती है।

Syntax: <file handle>.close()
Example: student.close()
shop.close()

(iii) write() → इस फंक्शन के द्वारा फाइल में को लिखा जाता है। यह फंक्शन record को String के रूप में लेता है तथा प्रत्येक record अंत में कैरेक्टर '\n' लगाया जाता है। लाइन के EOL

NOTE: इस फंक्शन को प्रयोग करने से पहले ~~ओपन~~ फाइल write mode में होनी चाहिए। open

Syntax: <file handle> = write (<data>)
Example: nm = 'Ronak jain', + '\n'
student.write (nm)

(iv) readline ( ) → फाइल में डेटा स्टोर होने के बाद इस फाइल से एक-एक लाइन प्राप्त करने के लिए इस फंक्शन का प्रयोग होता है।
इस फंक्शन का प्रयोग करने से पहले फाइल read mode में open होनी चाहिए।

Syntax: <file handle>.readline ( )
Example: student.readline ( )

Program: 15 छात्रों की कक्षा में छात्रों के रोल नं., नाम और तीन विषयों में प्राप्त अंको को संग्रहीत करने के लिए एक टेक्स्ट फाइल बनाने के लिए एक प्रोग्राम लिखे।

```
student = open ('s data.txt', 'w')
for a in rang (15):
    record = " "
    rno = int (input(" enter roll no."))
    record = record + rno + '\n'
    nm = input (" enter name")
    record = record + nm + '\n'
    m1 = int (input (" enter mark 1 "))
    record = record + m1 + '\n'
    m2 = int (input (" enter mark 2 "))
    record = record + m2 + '\n'
    m3 = int (input (" enter mark 3 "))
    record = record + m3 + '\n'
    student.write (record)
student.close ( )
```

More Commands to write & read the file :—

[A] **read()** → यह फंक्शन फाइल से केवल उतनी बाइट करता है जितनी argument में दी गई है। read

Syntax: <file handle>.read(<no. of bytes>)
Example: student.read(30)

[B] **readlines()** → फाइल से total infor. read करने के लिए इस फंक्शन का प्रयोग होता है। यह फंक्शन लिस्ट के रूप में डेटा प्राप्त करता है।

Syntax: <file handle>.readlines()
Example: n = student.readlines
print(n)

[C] **writelines()** → यह फंक्शन स्ट्रिंग की एक लिस्ट को argument के रूप में लेता है तथा स्ट्रिंग को फाइल में अलग-अलग लाइन्स में लिखता है।

Syntax: <File handle>, writelines(<list>)
Example: student.writelines(slist)

Program: निम्न प्रोग्राम यूजर से इनपुट के रूप में फलों के नाम लेता है और सभी नामों की सूची बनाते हैं। एक बार सूची तैयार होने के बाद, लिखित मोड में एक फाइल खोलता है और सूची को फाइल में संग्रहीत करता है।

```
fruitfile = open('names.txt','w')
flist = []
for i in range(5):
    nm = input("enter a fruit name:")
    flist.append(nm+'\n')
fruitfile.writelines(flist)
fruitfile.close()
```

Random reading in File :-

एक फाइल की एक stream होती है। जिसमे एक bites के द्वारा डेटा read किया जाता है। File pointer

जब फाइल ऑपन की जाती है तब File pointer की कुछ position शुरुआत में होती है तथा फाइल से डेटा read किया जाता है तब FP की position उस डेटा से आगे हो जाती है।

```
f1 = open('Fruites.txt', 'r')
```

A	P	P	L	E	\n	B	A	N	A	N	A	\n	O	R	A	N	G	E	\n
↑																			
fp																			

```
rm = f1.read(7)
```

A	P	P	L	E	\n	B	A	N	A	N	A	\n	O	R	A	N	G	E	\n
							↑												
							fp												

पायथन में fp की position को खोजने तथा बदलने के लिए दो Function बनाए गये है -

1. **tell()** - इस फंक्शन का प्रयोग fp की current position Find करने के लिए किया जाता है।

Syntax: <File handle>.tell()

Example:
```
f1 = open('Fruites.txt','r')
print('position of fp :', f1.tell())
r = f1.read(3)
print('Position of fp after reading 3 bytes :', f1.tell())
```

2. Seek() → इस फंक्शन का प्रयोग fp की position बदलने के लिए किया जाता है।

Syntax: <File handle>.seek(<offset>)

Example: fi.seek(10)
print('position of fp after change:', f1.tell())

(2.) Binary File processing :-

बाइनरी फाइल डेटा को उस फॉर्मेट में स्टोर करती है जिस फॉर्मेट में इसे मेमोरी में स्टोर किया जाता है।
बाइनरी फाइल में लाइन के अंत में delimiter नही होता।
पायथन के द्वारा बनाई गई binary file केवल python सॉफ्टवेयर में ओपन होती है।
बाइनरी फाइल को .dat extension के साथ सेव किया जाता है।

[1] 'pickle' module : बाइनरी फाइल में डेटा स्टोर करने तथा एक्सेस करने के लिए 'pickle' का प्रयोग किया जाता है।
pickle module इसके द्वारा record को Serialize तथा De-Serialize किया जाता है।

Serialize → यह वह प्रक्रिया है जिसमें डेटा को bytes में बदला जाता है।

De-Serialize → यह वह प्रक्रिया है जिसमे bytes के डेटा को simple format में बदला जाता है।

इसके दो module प्रयोग किये जाते है —

dump ( )
load ( )

Open ( ) → बाइनरी फाइल के साथ work स्टार्ट करने से पहले इसे एक विशेष mode में ऑपन किया जाता है। मोड फाइल के उद्देश्य को दर्शाता है।

Open ( ) के द्वारा फाइल ऑपन की जाती है।

x: <file handle> = open ('<file name>', '<mode>')

Mode निम्न प्रकार के होते है।

File mode	Purpose	Description
wb	write	Same as text file
rb	read	"
ab	append	"
wb+	write & read	"
rb+	read & write	"
ab+	append & read	"

ample: school = open ('student.dat', 'wb')
employ = open ('income.dat', 'rb')

[3] **close ( )** → बाइनरी फाइल में work पूरा होने के बाद इसे close किया जाता है।

Syntax: <file handle>.close ( )
Example: school.close ( )

[4] dump() → यह फंक्शन pickle module में define किया गया है। जिसका प्रयोग बाइनरी फाइल में डेटा लिखने के लिए किया जाता है।

Syntax: pickle.dump(<field>, <file handle>)
Example: pickle.dump(rno, school)

[5] load() → यह फंक्शन भी pickle module में define किया गया है। इसका प्रयोग बाइनरी फाइल से डेटा प्राप्त करने के लिए किया जाता है।

<field> = pickle.load(<file handle>)
rno = pickle.load(school)

Checking End of File :-

पायथन में एक तरीका है जिसका प्रयोग करके फाइल के अंत का पता लगाया जाता है।

```
try:
    while True:
        <read, write, process>
except EOF Error:
        <file handle>.close()
```

Deletion of records :-

फाइल से record हटाने के लिए पायथन में direct command नही है। Record को हटाने के लिए एक logic का प्रयोग किया जाता है जिसमें दो फाइल ऑपन की जाती है। एक फाइल r मोड में तथा दूसरी फाइल w मोड में ऑपन की होती है।

एक फाइल से डेटा प्राप्त किया जाता है तथा दूसरी फाइल में स्टोर किया जाता है।
पहली फाइल को remove कर दी जाती है तथा दूसरी फाइल का नाम बदल दिया जाता है।
इस प्रोसेस के लिए os module प्रयोग किया जाता है।

- **remove ( )**

Syntax: os. remove ('<file name>')
Example: os. remove ('salary.dat')

- **rename ( )**

Syntax: os. rename ('<old file>', '<new file>')
Example: os. rename ('<temp.dat>', '<salary.dat>')