File Handling in Python

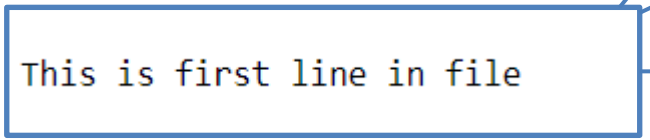
Course Contents

- Opening a File – Read , Write & Append Mode
- Read & Write Operations on File
- Other File Operations
- The Pickle (Serialize and De-serialize Python Objects)

Opening a File

- In Python , open() is the function used to open a file.

- There are 3 modes of file opening
  - ❑ Read – r , rb
  - ❑ Write – w , wb
  - ❑ Append – a , ab



File Name : abc.txt

```
f = open("abc.txt" , "r")
print(f.read())
f.close()
```

```
This is first line in file.
```

```
f = open("abc.txt" , "w")
f.write('This is new line now')
f.close()
```

```
f = open('abc.txt' , 'a')
f.write('\nThis is second line in file')
f.close()
```

Reading Operation

- In Python , after opening the file in Read mode one can perform following 3 reading operations.
  - ❑ f.read()
  - ❑ f.readline()
  - ❑ f.readlines()

```
This is new line now
This is second line in file
```

File Name : abc.txt

```
f = open("abc.txt" , "r")
s = f.read(6)
print(s)
f.close()
```

```
This i
```

```
f = open("abc.txt" , "r")
s = f.readline()
print(s)
f.close()
```

```
This is new line now
```

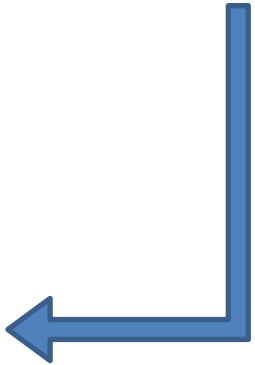
```
f = open("abc.txt" , "r")
s = f.readlines()
print(s)
f.close()
```

```
['This is new line now\n',
 'This is second line in file']
```

```
f = open("abc.txt" , "r")
for line in f:
    print(line)
f.close()
```

```
This is new line now

This is second line in file
```



Writing Operation

- In Python , after opening the file in Write mode one can perform following 2 writing operations.

❑ f. write()
❑ f. writelines()

- If the file is not present , new file will be created.

- If the file is already present with same name , it is over written.

```
f = open("abc.txt" , "w")
f.write("This is new file.")
f.close()
```

```
f = open("abc.txt" , "w")
lst  = ['This is first line in file\n', 'This is second line in file']
f.writelines(lst)
f.close()
```

Other File Operations

- The **tell()** method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.
- The **seek(offset[, from])** method changes the current file position. The **offset** argument indicates the number of bytes to be moved. The **from** argument specifies the reference position from where the bytes are to be moved.
- If from is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

tell() and seek()

- f.tell() returns an integer giving the file object's current position in the file
- To change the file object's position, use f.seek()

```
f = open("abc.txt", "rb")
s = f.read(10)
s
```

```
b'This is fi'
```

```
# Check current position
position = f.tell()
position
```

```
10
```

```
# Reposition pointer at the beginning once again
position = f.seek(0,0)
s = f.read(10)
s
```

```
b'This is fi'
```

```
f.close()
```

Using ‘with’ clause

- The advantage is that the file will be automatically closed after the indented block after the With has finished execution:

```
with open("abc.txt", "w") as fh:
    fh.write("To write or not to write \nthat is the question!\n")
```

```
with open("abc.txt") as fobj:
    for line in fobj:
        print(line.rstrip())
```

```
To write or not to write
that is the question!
```

- The rstrip() method removes any trailing characters (characters at the end a string), space is the default trailing character to remove.

The Pickle

- With the algorithms of the pickle module we can serialize and de-serialize Python object structures.
- "Pickling" denotes the process which converts a Python object hierarchy into a byte stream, and "unpickling" on the other hand is the inverse operation, i.e. the byte stream is converted back into an object hierarchy.
- What we call pickling (and unpickling) is also known as "serialization" or "flattening" a data structure.
- Objects which have been dumped to a file with pickle.dump can be reread into a program by using the method pickle.load(file).

```
import pickle
```

```
tour = {"ciies":["Paris", "New Delhi", "Mumbai", "Surat"],
        "hotels":["The Taj","Pride","Radison"]}
```

```
fh = open("data.pkl", "bw")
pickle.dump(tour, fh)
fh.close()
```

The Pickle

- pickle.load() recognizes automatically, which format had been used for writing the data.
- The file data.pkl can be read in again by Python in the same or another session or by a different program

```
import pickle
```

```
f = open("data.pkl", "rb")
new_tour= pickle.load(f)
print(new_tour)
f.close()
```

```
{'ciies': ['Paris', 'New Delhi', 'Mumbai', 'Surat'], 'hotels': ['The Taj', 'Pride', 'Radison']}
```