Module 6:

File Handling

By Ninad Gaikwad

Reference - “Core Python Programming”
Dr. R. Nageshwara Rao
Dreamtech Press

17.1 Text File Handling

- File Opening -
  File handler = open(“file name”, “open mode”)
  Eg. f = open(“myfile.txt”, “w”)
- Modes - write(w), read(r), append(a), write+read(w+), read+write(r+), append+read(a+), exclusive creation (x)
- File closing -
  Eg. f.close()
- Reading from file
  Eg. f.read() # read string from file
  f.readlines() # reads all the lines into a list
- Program to store group of strings in a text file
  Eg.

```
f = open(‘myfile.txt’, ‘w’)

print(‘Enter text (@ at the end): ’)
While str != ‘@’:
        str = input()
        if str != ‘@’:
                f.write(str+’\n’)
f.close()
```

17.1 Text File Handling

- To move specified number of bytes from beginning, end or current position

  f.seek(offset, fromwhere)

  f.seek(10,0) # 10 byte from beginning

  f.seek(10,1) # 10 byte from current position

  f.seek(-10,2) # 10 byte from end
- Knowing whether a file exists or not
  - os module has a submodule by the name path which contains a method isfile().
  - This method can be used to know whether the file we are trying to open really exists or not

    Eg.

```
if os.path.isfile(fname):
        f.open(fname,'r')
else:
        print(fname+' does not exist')
        sys.exit()
```

17.2 Binary File Handling

- Binary files handle data in the form of bytes. They can be used to handle text, images, audio and video files.
- To open file for reading purpose we can use mode as ‘rb’. Here b represents the binary file
- A program to copy an image to another file
  Eg.

```
f1 = open(‘cat.jpg’,’rb’)
f2 = open(‘new.jpg’,’wb’)

bytes=f1.read()
f2.write(bytes)

f1.close()
f2.close()
```

- It is very important to close the files that are opened using open statement

17.3 The with statement

- With statement can be used while opening a file
- Advantage of using a with statement is that we dont have to worry about closing the file opened using the with statement
- Eg

```
with open('sample.txt','r') as f:
        for line in f:
                print(line)
```

17.4 Pickle in python

- When we want to store structured data in files we can use pickle
- pickle.dump(object, file) is used to store data in files
- Eg.

```
# Define  Emp class
class Emp:
    def __init__(self, id, name, sal):
        self.id = id
        self.name = name
        self.sal = sal
    def display(self):
        print("{:5d} {:20s} {:10.2f}".format(self.id, self.name, self.sal))
```

pickle.dump

```
# Store Emp class in emp.dat file
f=open('emp.dat','wb')
n=int(input('How many employees'))
for i in range(n):
    id = int(input('enter id '))
    name = input('enter name ')
    sal = float(input('enter salary'))
    e = Emp(id,name, sal)
    pickle.dump(e,f)
f.close()
```

pickle.load

- pickle.load(file) is used to load data from file
- Eg.

```
# Loading class from binary file using pickle
f = open('emp.dat','rb')
print('Employee details: ')
while True:
    try:
        obj =pickle.load(f)
        obj.display()
    except EOFError:
        print('end of file reached')
        break
f.close()
```

- f.tell() will give you the current position of the file pointer
- f.seek(offset, fromwhere) will take you to the position specified

17.5 Zipping and unzipping files

- File contents are compressed and hence size is reduced
- Format of data is changed and hence made unreadable
- Eg.

```
# Zipping Files
from zipfile import *
f = ZipFile('test.zip', 'w', ZIP_DEFLATED)
f.write('file1.txt')
f.write('file2.txt')
f.write('file3.txt')
print('test.zip file created')
f.close()

# Unzipping Files
form zipfile import *
z=ZipFile('test.zip','r')
z.extractall()
```

17.6 Working with Directories

- os.getcwd() gives the current working directory
- os.mkdir(“dirname”) creates a directory by the name dirname
- os.mkdir(“dirname/subdirs”) creates a directory by the name dirname and subdirs if it does not exist
- os.rmdir(dirname) removes the directory
- Display all the contents of current directory
- Eg.

```
import os
for dirpath, dirname, filename in os.walk(“.”):
        print(“Current Path”, dirpath)
        print(“Direcrories”, dirname)
        print(“File name”, filename)
        print()
```

Experiment 11:

Exploring Files and directories

- Python Program to read the content of file and write it in another file
- Python program to append data to existing file and then display the entire fille
- Python program to count the number of lines, words and characters in a file.
- Python program to display file available in current directory