III B.Tech.

Computer Science & Engineering

CSE304: PYTHON PROGRAMMING WITH WEB FRAMEWORKS

UNIT-II: File I/O

By

Mrs. S. KAMAKSHI, AP-III / CSE

School of Computing

Types of Files

- Text file
  - CSV Files, XML files, JSON files, HTML files
- Binary File
  - Program files, image files, Audio files, video files, database files, compressed files
- File Operations
  - open, close, read, write

File Open & Close Operations

- Syntax for opening a file
  - fileobject = open(file, mode='r')
  - File: path to the file
  - Mode:
    - r – to read from file, default
    - w – to write into file
    - a – to append to a file
    - b – for binary file along with read or write mode
  - Used along with "with" statement

    with open(file, mode) as fileobject

        statements …
- Syntax for closing the file
  - fileobject.close()
- Example
  - fp = open("text.txt", 'r')
    - Opens the file text.txt in the current working directory for reading, if exists. Otherwise raises exception
  - with open("text.txt", "r") as fp:

    statements …
    - Opens the file text.txt in the current working directory for reading, if exists. Otherwise raises exception

File Read & Write operations

- str = read()
  - reads entire file and returns its contents as a string
- strlist = readlines()
  - reads entire file and returns it as a list of strings
- str = readline()
  - reads the next line from the file and returns it as a string
- File object is an iterable and can be used iterate through its contents line by line directly

Examples

- **Eg. 1**

```
fp = open('myfile.txt')
print(fp.read())
```

- **Eg. 2**

```
fp = open('myfile.txt')
print(fp.readlines())
```

- **Eg. 3**

```
with open('myfile.txt') as fp:
    line = fp.readline()
    while line:
        print(line,end='')
        line = fp.readline()
```

- **Eg. 4**

```
for line in open('myfile.txt'):
    print(line, end='')
```

Reading and Writing Lists

List of Strings

```
MyStrings = ['This is First Line',
             'This is Second Line',
             'This is Third Line']
with open('mylist.txt', 'w') as fp:
   for mystr in MyStrings:
      fp.write(mystr+'\n')
contents = []
for line in open('mylist.txt'):
   contents.append(line)
print(contents)
```

List of Numbers

```
MyNumbers = [45, 56, 90, 23, 11, 8]
with open('mylist.txt', 'w') as fp:
   for num in MyNumbers:
      fp.write(str(num)+'\n')
contents = []
for num in open('mylist.txt'):
   contents.append(int(num))
print(contents)
```

```
MyNumbers = [45, 56, 90, 23, 11, 8]
with open('mylist.txt', 'w') as fp:
   fp.write(str(MyNumbers)+'\n')
fp = open('mylist.txt')
numbers = fp.read()
print(type(numbers))
print(numbers)
```

```
MyNumbers = [45, 56, 90, 23, 11, 8]
with open('mylist.txt', 'w') as fp:
   fp.write(str(MyNumbers)+'\n')
fp = open('mylist.txt')
numbers = eval(fp.read())
print(type(numbers))
print(numbers)
```

Reading and Writing CSV files

- csv Module contains the reader and writer methods to read and write comma separated values (csv) files
- import csv # to make use of reader and writer methods
- Examples:
- **For Writing:**

```
with open('myfile.csv', 'w', newline='') as fp:
    w_obj = csv.writer(fp)
    mylist =[['Ram', 'Manager', 75000], ['Lakshmi', 'Asst. Manager', 60000],
             ['Kumar', 'Clerk', 30000], ['Raja', 'Peon', 10000]]
    w_obj.writerows(mylist)
```

- **For Reading:**

```
with open('myfile.csv', 'r', newline='') as fp:
    r_obj = csv.reader(fp)
    for row in r_obj:
        print(row)
```

Read / Write with optional arguments

Argument	Description
quoting=csv.QUOTE_MINIMAL	Specifies when quotes are written and read. QUOTE_MINIMAL: adds quotes to columns that contain special characters such as the delimiter, quote, or \n. Other options are: QUOTE_ALL, QUOTE_NONNUMERIC, QUOTE_NONE
quotechar=“”’	Specifies the character that is used to quote columns
delimiter=‘,’	Specifies a one-character string used to separate fields

```
with open('employee_file.csv', mode='w',newline='') as ef:
    ew = csv.writer(ef, delimiter=',',  quotechar='"',  quoting=csv.QUOTE_MINIMAL)
    ew.writerow(['Kumar', 'Marketing', 'January'])
    ew.writerow(['Raja', 'Finance', 'May'])

with open('employee_file.csv') as ef:
    er = csv.reader(ef, delimiter=',', quotechar='"',  quoting=csv.QUOTE_MINIMAL)
    for row in er:
        print(row)
```

Read / Write as Dictionary

```
with open('employee_file2.csv', mode='w', newline='') as csv_file:
    fieldnames = ['emp_name', 'dept', 'birth_month']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)

    writer.writeheader()
    writer.writerow({'emp_name': 'Ram', 'dept': 'Accounting', 'birth_month': 'November'})
    writer.writerow({'emp_name': 'Lakshmi', 'dept': 'IT', 'birth_month': 'March'})

with open('employee_file2.csv') as employee_file:
    employee_reader = csv.DictReader(employee_file)
    for row in employee_reader:
        print(row)

with open('employee_file2.csv') as employee_file:
    employee_reader = csv.DictReader(employee_file)
    for row in employee_reader:
        print('Name:', row['emp_name'], '\nDepartment:', row['dept'], '\nBirth Month:',
row['birth_month'])
```

Reading as Dictionary that has no field names


```
fieldnames=['emp_name', 'dept', 'birth_month']
with open('employee_file.csv') as ef:
    er = csv.DictReader(ef, fieldnames)
    for row in employee_reader:
        print('Name:', row['emp_name'])
        print('Department:', row['dept'])
        print('Birth Month:', row['birth_month'])
```

Creating Binary Files using pickle module

- import pickle *#to use dump and load methods*
- Use dump() and load() methods of pickle module to write into and read from binary files
- Example

```
MyStrings = ['This is First Line', 'This is Second Line', 'This is Third Line']
with open('myfile.bin', 'wb') as fp:
        pickle.dump(MyStrings, fp)

with open('myfile.bin', 'rb') as fp:
      MyStrings = pickle.load(fp)
      print(MyStrings)
```