accenturetechnology

Learning and Knowledge Management

PYTHON PROGRAMING

DAY 5

OBJECTIVES

End of this module you will be able to understand

- **How to Work with Files in Python**
- **How to Work with openpyxl module to access excel spreadsheets**
- **How to work with PyPDF2 module to access pdf documents**
- **How to work with python-docx module to access word documents**
- **How to work with JSON and CSV files using json and csv modules**

FILE I/O FUNCTIONS

- **Python provides basic functions and methods necessary to manipulate files**
- **Most of the file manipulations is done using a ‘file’ object**
- **Before we read or write a file, we need to open it using Python's built-in *open()* function.**

Ex: file = open(“c:/Training/file.txt", “r")

- **Open takes 3 parameters : file name, access mode & buffering**

ACCESS MODES

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

EXAMPLE

```
def demofile(file):

    file=open(file,'r');

    print ("Name of the file: ", file.name)
    print ("Closed or not : ", file.closed)
    print ("Opening mode : ", file.mode)

    file.close()

demofile('c:/Training/file.txt')
```

COUNT NUMBER OF WORDS

```
def countWords(filePath):

    file=open(filePath,'r');

    numberOfWords=0;
    words=[]

    for line in file:
        for word in line.split():
            numberOfWords+=1
            words+=word


    print(numberOfWords)
```

OPENPYXL MODULE TO WORK WITH EXCEL

BASIC DEFINITIONS

- **Workbook**
- **.xlsx extension**
- **Sheets & Active Sheet**
- **Column**
- **Cell**

INSTALLING THE OPENPYXL MODULE

Download openpyxl

pip install *openpyxl (Automatically downloads dependent library)*

python setup.py install

PROCESSING EXCEL DOC

- Opening Excel Documents with OpenPyXL
- Getting Sheets from the Workbook
- Getting Cells from the Sheets
- Converting Between Column Letters and Numbers
- Getting Rows and Columns from the Sheets

PYTHON PROGRAM WITH OPENPYXL

```
 #import relevant module
 import openpyxl
 #Opening excel workbook
wb = openpyxl.load_workbook('c:/Training/PyFiles/example.xlsx')

#Accessing a specific sheet
sheet=wb.get_sheet_by_name('Sheet1')  #(Or wb.get_active_sheet()) print(sheet.title)  # Prints Sheet Title

#Different ways to get to cells
print(sheet['A1'])
print(sheet['A1'].value)

print(sheet.cell(row=1, column=2))
print(sheet.cell(row=1, column=2).value)
for i in range(1, 8, 2):
      print(i, sheet.cell(row=i, column=2).value)
```

GETTING ROWS AND COLUMNS

```
import openpyxl

wb = openpyxl.load_workbook('c:/Training/PyFiles/example.xlsx')
sheet = wb.get_sheet_by_name('Sheet1')

for rowOfCellObjects in sheet['A1':'C3']:
    for cellObj in rowOfCellObjects:
        print(cellObj.coordinate, cellObj.value)
print('--- FINISHED---')
```

WORKING WITH PYTHON-DOCX

WORD DOCUMENTS

- Python can create and modify Word documents, which have the *.docx* file extension, with

the *python-docx* module.

- To install *‘python-docx’* ececute the following command

```
$pip install python-docx
```

- *‘.docx’* files have lot of structure compared to regular text, which is represented by 3 different data types in python.
  - Document
  - Paragraph
  - Run

READING WORD DOCUMENT

```
import docx
doc = docx.Document('c:/Training/demo.docx')

print(len(doc.paragraphs))

print(doc.paragraphs[0].text)
print(doc.paragraphs[1].text)
print(len(doc.paragraphs[1].runs))
print(doc.paragraphs[1].runs[0].text)
print(doc.paragraphs[1].runs[1].text)
print(doc.paragraphs[1].runs[2].text)
print(doc.paragraphs[1].runs[3].text)
```

GETTING THE FULL TEXT FROM A .DOCX FILE

```
import docx

def getText(file):
    doc = docx.Document(file)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    return '\n'.join(fullText)



print (getText("c:/Training/demo.docx"))
```

WRITING WORD DOC

```
from docx import Document
from docx.shared import Inches

document = Document()

document.add_heading('Document Title', 0)

p = document.add_paragraph('A plain paragraph having some ')

document.add_page_break()

document.save('C:/Training/demo.docx')
```

WORKING WITH JSON AND CSV

Introduction

- CSV and JSON files are just plaintext files unlike PDF and Word which are binary in nature
- CSV stands for “comma-separated values,” and CSV files are simplified spreadsheets stored as plaintext files
- JSON is a format that stores information as JavaScript source code in plaintext files
- No need to install any additional packages

CSV MODULE

Each line in a CSV file represents a row in the spreadsheet, and commas separate the cells in the row

Ex:

```
4/5/2015 13:34,Apples,73
4/5/2015 3:41,Cherries,85
4/6/2015 12:46,Pears,14
4/8/2015 8:59,Oranges,52
4/10/2015 2:07,Apples,152
4/10/2015 18:10,Bananas,23
4/10/2015 2:40,Strawberries,98
```

The advantage of CSV files is simplicity. CSV files are widely supported by many types of programs, can be viewed in text editors

READ & WRITING CSV

```
import csv

stuCSV=open('c:/Training/Student.csv')
csvRd=csv.reader(stuCSV)
csvData=list(csvRd)
print(csvData)

for rec in csvData:
    print(rec)
```

```
import csv
outputFile = open('c:/Training/output.csv', 'w', newline='')
outputWriter = csv.writer(outputFile)
outputWriter.writerow(['Amar', 'Raj', 'Simran', 'Johnny'])
outputWriter.writerow(['SE', 'SSE', 'TL', 'AM'])
outputWriter.writerow([11, 10, 9, 8])
outputFile.close()
```

JSON MODULE

Python’s json module handles all the details of translating between a string and JSON data with the help of json.loads() and json.dumps()

JSON can’t store *every* kind of Python value. It can handle strings, integers, floats, Booleans, lists, dictionaries, and NoneType only

READ & WRITE JSON

Reading JSON with the loads() Function

```
import json

jsonData='{"PID":"11203094","Name":"Mohan","Project":"LKM","Designation":"TL"}'

data = json.loads(jsonData)

print(data)
```

Writing JSON with the dumps() Function

```
import json

pyData={'PID':11203094,'Name':'Mohan','Project':'LKM','Designation':'TL'}

jsonData = json.dumps(pyData)

print(jsonData)
```

PYPDF2 MODULE

INTRODUCTION

PDF **stands for** ***Portable Document Format*** **and uses the** ***.pdf*** **file extension.**

Processing PDF documents is one of the tedious task considering their complex structure that are difficult to understand

Python provides several packages that can help. The following list displays some of the most popular ones

- **pdfrw**
- **slate**
- **PDFQuery**
- **PDFMiner**
- **PyPDF2**

In this training we will understand how to manipulate PDF documents using PyPDF2 library which still being continued to develop.

INSTALLATION

- **Download PyPDF2 from PyPDF2_1.26.0**
- **Get into the downloaded folder and execute pip command as follows**
  - ***Cd C:\Training\Python\PyPDF2-1.26.0***
  - ***C:\Training\Python\PyPDF2-1.26.0>pip install pypdf2***

PROCESSING PDF

You can manipulate PDF files in a variety of ways using the pure-Python PyPDF2 toolkit. We can perform following operations.

- Merge (layer/append)
- Delete
- Split
- Slices
- Add Metadata
- Crop
- Encrypt

MERGE (APPEND)

```
from PyPDF2 import PdfFileMerger, PdfFileReader
import os

merger = PdfFileMerger()
files = [x for x in os.listdir('c:/mypdfs') if x.endswith('.pdf')]
for fname in sorted(files):
    merger.append(PdfFileReader(open(os.path.join('c:/mypdfs', fname), 'rb')))

merger.write("c:/Training/output.pdf")
```

MERGE (LAYER)

```
from PyPDF2 import PdfFileWriter, PdfFileReader
output = PdfFileWriter()

ipdf = PdfFileReader(open('c:/mypdfs/Acn.pdf', 'rb'))
wpdf = PdfFileReader(open('c:/mypdfs/Wm.pdf', 'rb'))
watermark = wpdf.getPage(0)

for i in range(ipdf.getNumPages()):
    page = ipdf.getPage(i)
    page.mergePage(watermark)
    output.addPage(page)

with open('c:/Training/output.pdf', 'wb') as f:
   output.write(f)
```

DELETE

```
from PyPDF2 import PdfFileWriter, PdfFileReader
infile = PdfFileReader('c:/mypdfs/PoetryWBP.pdf', 'rb')
output = PdfFileWriter()

for i in range(infile.getNumPages()):
    p = infile.getPage(i)

    text=p.extractText()

    if len(text) > 10: # getContents is None if  page is blank
       # print(i,len(text),text)
        output.addPage(p)

with open('c:/training/output.pdf', 'wb') as f:
   output.write(f)
```

SPLIT

```
from PyPDF2 import PdfFileWriter, PdfFileReader
infile = PdfFileReader(open('c:/mypdfs/Poetry.pdf', 'rb'))

for i in range(infile.getNumPages()):
    p = infile.getPage(i)
    outfile = PdfFileWriter()
    outfile.addPage(p)
    with open('c:/Training/page-%02d.pdf' % i, 'wb') as f:
        outfile.write(f)
```

SUMMARY

In this module we have understood:

- **How to Work with Files in Python**
- **How to Work with openpyxl module to access excel spreadsheets**
- **How to work with PyPDF2 module to access pdf documents**
- **How to work with python-docx module to access word documents**
- **How to work with JSON and CSV files using json and csv modules**