Ex. No: 8

Aim: *Read the following file formats using pandas*

a. Text files
b. CSV files
c. Excel files
d. JSON files

8 (A):

Text files:

- Text files are one of the most common file formats to store data. Python makes it very easy to read data from text files.
- Python provides the open() function to read files that take in the file path and the file access mode as its parameters. For reading a text file, the file access mode is 'r'. Other access modes below:

  'w' – writing to a file
  'r+' or 'w+' – read and write to a file
  'a' – appending to an already existing file
  'a+' – append to a file after reading

- Python provides us with three functions to read data from a text file:
  read(n) – This function reads n bytes from the text files or reads the complete information from the file if no number is specified. It is smart enough to handle the delimiters when it encounters one and separates the sentences.
  readline(n) – This function allows you to read n bytes from the file but not more than one line of information.
  readlines() – This function reads the complete information in the file but unlike read(), it doesn't bother about the delimiting character and prints them as well in a list format.

Example:

Sample File.txt

Welcome to DS lab for how to import files in python. We will work with the following types of files:
1. Text
2. CSV
3. Excel
4. JSON

8a.py

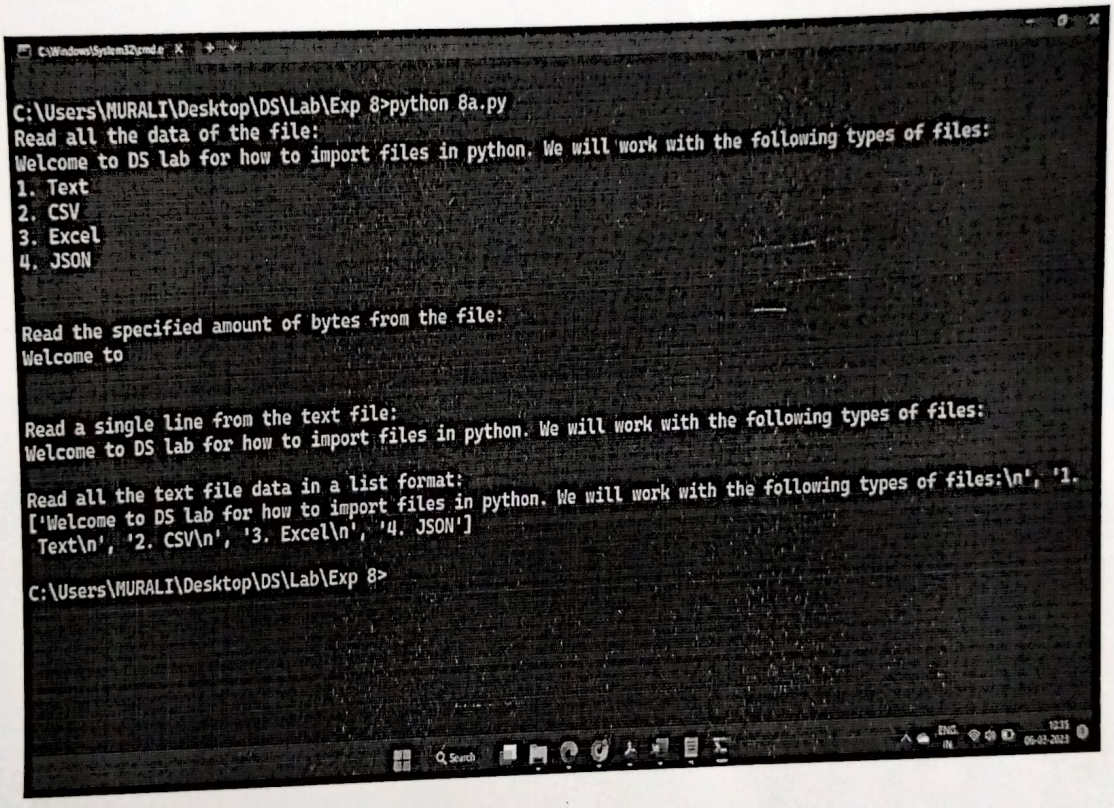
```
print("Read all the data of the file:")
with open('Sample File.txt','r') as f:
    print(f.read())
print("\n")

print("Read the specified amount of bytes from the file:")
with open('Sample File.txt','r') as f:
    print(f.read(10))
print("\n")
```

```
print("Read a single line from the text file:")
with open('Sample File.txt','r') as f:
    print(f.readline())

print("Read all the text file data in a list format:")
with open('Sample File.txt','r') as f:
    print(f.readlines())
```

Output:



8 (B):

CSV files:

- A CSV (Comma Separated Value) file is the most common type of file that a data scientist will ever work with.
- These files use a "," as a delimiter to separate the values and each row in a CSV file is a data record.
- These are useful to transfer data from one application to another and is probably the reason why these are so common place in the world of data science.
- The Pandas library makes it very easy to read CSV files using the read_csv() function:

Example:

Products.txt

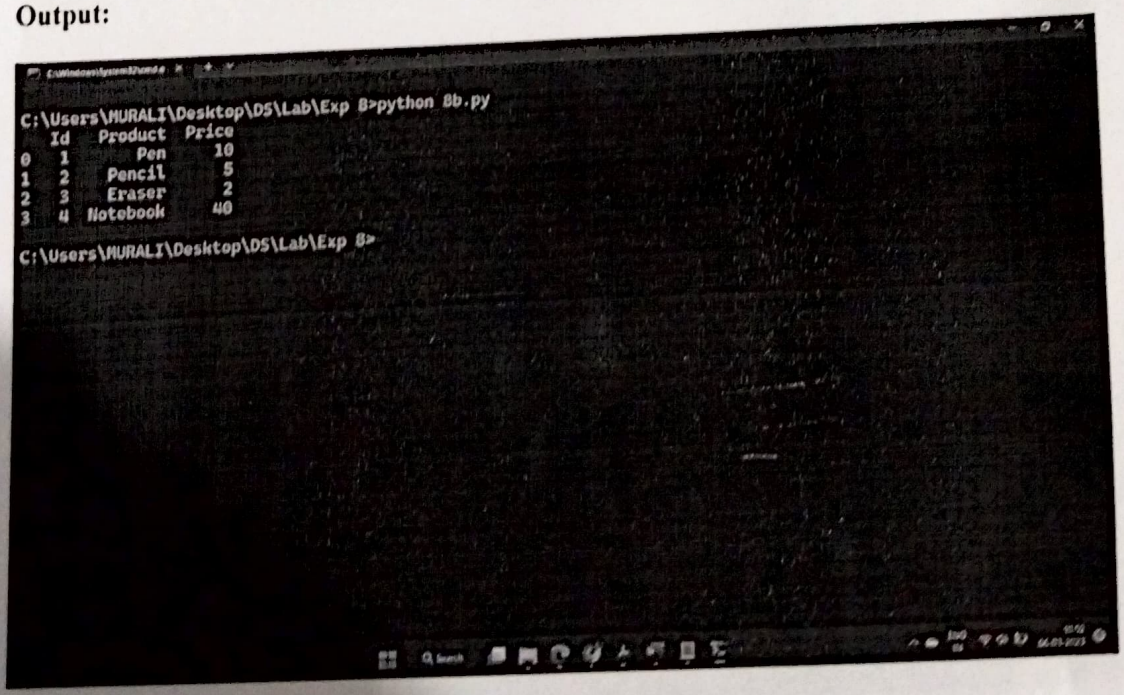
```
Id,Product,Price
1,Pen,10
2,Pencil,5
3,Eraser,2
4,Notebook,40
```

8b.py

```
import pandas as pd

df = pd.read_csv('Products.txt',delimiter=',')
print(df)
```

Output:



8 (C):

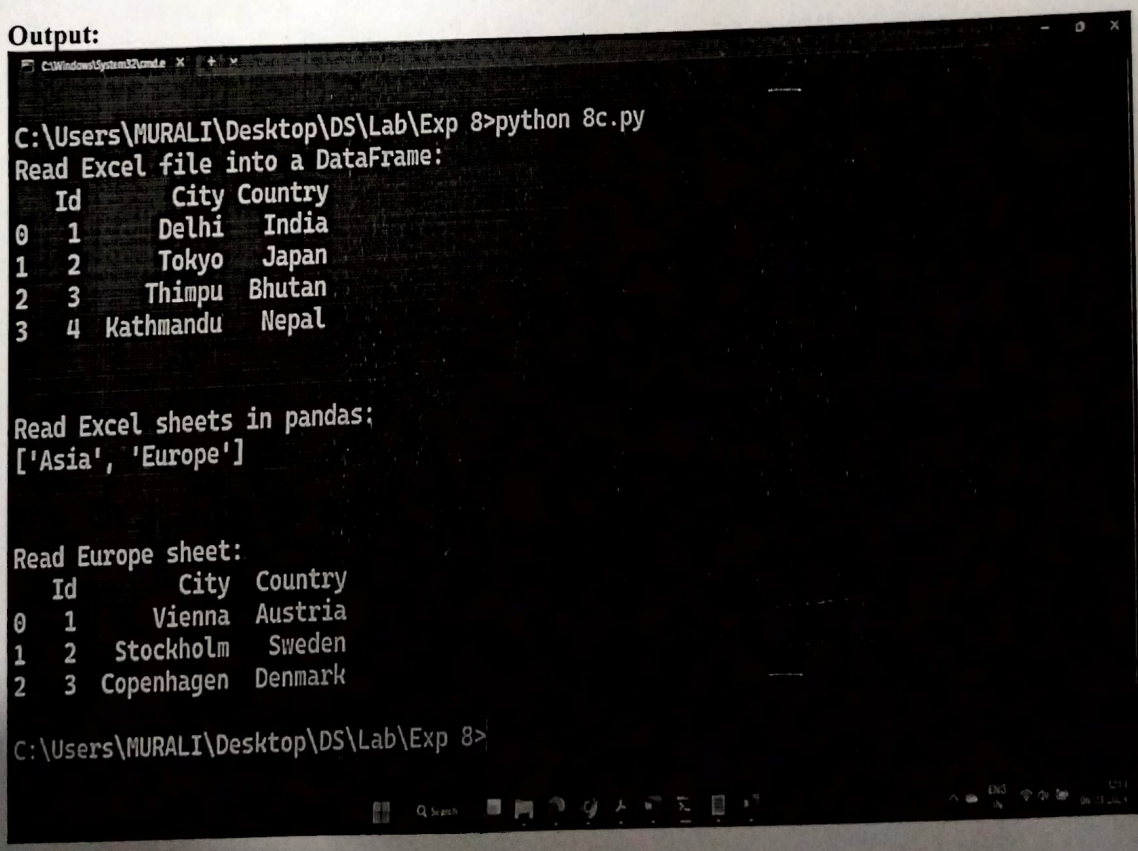
Excel files:

- Pandas has a very handy function called read_excel() to read Excel files.
- But an Excel file can contain multiple sheets, so we can use the Pandas' ExcelFile() function to print the names of all the sheets in the file.
- After printing that, we can easily read data from any sheet we wish by providing its name in the sheet_name parameter in the read_excel() function.

Example:

```
import pandas as pd
print("Read Excel file into a DataFrame:")
df = pd.read_excel('World_city.xlsx')
print(df)
print("\n")
print("Read Excel sheets in pandas:")
xl = pd.ExcelFile('World_city.xlsx')
print(xl.sheet_names)
print("\n")
print("Read Europe sheet:")
df = pd.read_excel('World_city.xlsx',sheet_name='Europe')
print(df)
```

Output:



8 (D):

JSON files:

- JSON (JavaScript Object Notation) files are lightweight and human-readable to store and exchange data. It is easy for machines to parse and generate these files and are based on the JavaScript programming language.
- JSON files store data within {} similar to how a dictionary stores it in Python. But their major benefit is that they are language-independent, meaning they can be used with any programming language – be it Python, C or even Java!
- Python provides a json module to read JSON files. You can read JSON files just like simple text files. However, the read function, in this case, is replaced by json.load() function that returns a JSON dictionary.
- Once you have done that, you can easily convert it into a Pandas dataframe using the pandas.DataFrame() function.
- But you can even load the JSON file directly into a dataframe using the pandas.read_json() function.

Example:

sample_json.json

```
{"Name": {"0": "Akash", "1": "Jill"}, "Company":{"0": "Analytics", "1": "Google"}, "Job": {"0": "Intern",
"1": "Full time"}}
```

8d.py

```
import json
import pandas as pd
# open json file
with open('sample_json.json','r') as file:
   data = json.load(file)
# json dictionary
print(type(data))
# loading into a DataFrame
df_json = pd.DataFrame(data)
print(df_json)
print("\n")
print("Reading directly into a DataFrame:")
df = pd.read_json("sample_json.json")
print(df)
```

Output:

```
C:\Users\MURALI\Desktop\DS\Lab\Exp 8>python 8d.py
<class 'dict'>
    Name     Company       Job
0  Akash  Analytics    Intern
1   Jill     Google  Full time


Reading directly into a DataFrame:
    Name     Company       Job
0  Akash  Analytics    Intern
1   Jill     Google  Full time

C:\Users\MURALI\Desktop\DS\Lab\Exp 8>
```