Assignment No. 1

Problem Statement: Reading and writing different types of datasets.

Objective: The objective of this assignment is to familiarize ourselves with reading and writing different types of datasets including .txt, .csv, and .xml from the web and local disk storage. We will explore how to load these datasets into memory, process them, and save them to a specific location on the disk.

Prerequisite :

1. A Python environment set up with libraries like pandas, xml.etree.ElementTree, and requests (for web access).
2. Internet connection (for reading datasets from the web).
3. Text editor and basic knowledge of file handling in Python.

Theory :

In data science, reading and writing different types of datasets is an essential task.Various file formats are used to store and manage data. So the most common types are the following-

1. **Text Files (.txt) -**
   - Text files store data in plain text format.
   - Each line can represent a data entry.
   - Reading and writing data from a .txt file in Python is done using simple commands like open(), read(), and write().
   - Text files are easy to work with but don't support complex structures like tables.

2. **Comma-Separated Values (.csv) -**
   - CSV files store data in a tabular format, with rows and columns.
   - Each entry is separated by a comma.
   - These files are widely used in data science for structured data.
   - Python's pandas library is commonly used to read and write CSV files using read_csv() and to_csv() functions.

3. **Extensible Markup Language (.xml) -**
    - XML files store data in a hierarchical structure with tags to define different data elements.
    - XML is commonly used for data exchange between systems.
    - Python's xml library or external libraries like lxml are used to read and write XML files.
    - The structure is more complex compared to .txt and .csv files.

Libraries –

1. **Pandas –**

    - It is open-source Python library used for data manipulation and analysis.
    - It provides data structures like DataFrames and Series, which allow for efficient handling of structured data, similar to tables in databases or Excel spreadsheets.
    - With Pandas, users can easily read, write, filter, and process various types of datasets, including CSV, Excel, JSON, and even XML.
    - It is widely used in data science for tasks like data cleaning, transformation due to its flexibility and ease of use.

2. **xml.etree.ElementTree –**
    - It is a standard Python module used for parsing, creating, and modifying XML files.
    - It provides a simple and efficient way to navigate and manipulate XML documents through its tree structure. Elements are in a hierarchical manner.
    - The library supports reading XML from files or strings and writing XML back to files.
    - It's easy to use for basic XML handling tasks, making it suitable for smaller projects or simple data extraction.

3. **lxml –**
    - It is a Python library used for parsing, processing, and creating XML and HTML documents.
    - With lxml, users can easily manipulate XML files using XPath and XSLT, making it ideal for handling large or complex XML data.
    - It is also highly efficient and widely used in web scraping and data extraction tasks.

Algorithm (if any to achieve the objective)

1. **Reading and Writing .txt Files:**
   - Open the file in read mode. Using open("fileName.txt", "r") this syntax
   - Read the content (either line by line or all at once). Using content = file.read() this.
   - Process the content as needed.

2. **Reading and Writing .csv Files:**
   - Import pandas library like (import pandas as pd).
   - Use libraries to load the data, like df = pd.read_csv("fileName.csv")
   - For reading a .csv from the web, use url of that csv file. For ex.
     url = "http://example.com/file.csv"
     df = pd.read_csv(url)

Code & Output :

[1]:
```
# Importing the pandas library
import pandas as pd

# pd.read_csv() is used to load data from a CSV file into a DataFrame
df = pd.read_csv('C:/Users/dnyan/FODS Assignments/Datasets/iris.csv')

# df.head() shows the first 5 rows
df.head()
```

[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[2]:
```
# Opening and reading a text file
file_path = 'C:/Users/dnyan/FODS Assignments/Datasets/iris.txt'

# Using 'with' ensures the file is properly closed after reading
with open(file_path, 'r') as file:
    #reads the entire content of the file into a string and stores it in the variable data.
    data = file.read()

# Print the contents of the file
print(data)
```

```
Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species
1,5.1,3.5,1.4,0.2,Iris-setosa
2,4.9,3.0,1.4,0.2,Iris-setosa
3,4.7,3.2,1.3,0.2,Iris-setosa
4,4.6,3.1,1.5,0.2,Iris-setosa
5,5.0,3.6,1.4,0.2,Iris-setosa
6,5.4,3.9,1.7,0.4,Iris-setosa
7,4.6,3.4,1.4,0.3,Iris-setosa
8,5.0,3.4,1.5,0.2,Iris-setosa
9,4.4,2.9,1.4,0.2,Iris-setosa
10,4.9,3.1,1.5,0.1,Iris-setosa
11,5.4,3.7,1.5,0.2,Iris-setosa
12,4.8,3.4,1.6,0.2,Iris-setosa
13,4.8,3.0,1.4,0.1,Iris-setosa
14,4.3,3.0,1.1,0.1,Iris-setosa
15,5.8,4.0,1.2,0.2,Iris-setosa
16,5.7,4.4,1.5,0.4,Iris-setosa
17,5.4,3.9,1.3,0.4,Iris-setosa
18,5.1,3.5,1.4,0.3,Iris-setosa
19,5.7,3.8,1.7,0.3,Iris-setosa
20,5.1,3.8,1.5,0.3,Iris-setosa
21,5.4,3.4,1.7,0.2,Iris-setosa
22,5.1,3.7,1.5,0.4,Iris-setosa
23,4.6,3.6,1.0,0.2,Iris-setosa
24,5.1,3.3,1.7,0.5,Iris-setosa
25,4.8,3.4,1.9,0.2,Iris-setosa
26,5.0,3.0,1.6,0.2,Iris-setosa
27,5.0,3.4,1.6,0.4,Iris-setosa
28,5.2,3.5,1.5,0.2,Iris-setosa
29,5.2,3.4,1.4,0.2,Iris-setosa
30,4.7,3.2,1.6,0.2,Iris-setosa
31,4.8,3.1,1.6,0.2,Iris-setosa
32,5.4,3.4,1.5,0.4,Iris-setosa
33,5.2,4.1,1.5,0.1,Iris-setosa
34,5.5,4.2,1.4,0.2,Iris-setosa
35,4.9,3.1,1.5,0.1,Iris-setosa
```

[10]:
```python
# Importing the xml.etree.ElementTree library
# Helps in parsing and working with XML data by treating the XML document as a tree structure.
import xml.etree.ElementTree as ET

# function is used to load and parse the XML file
tree = ET.parse('C:/Users/dnyan/FODS Assignments/Datasets/iris.xml')

# retrieves the root element of the XML document
root = tree.getroot()

# To print the content or navigate through the XML tree
for elem in root.iter():
    print(elem.tag, elem.attrib, elem.text)
```

```
root {}

row {}

Id {} 1
SepalLengthCm {} 5.1
SepalWidthCm {} 3.5
PetalLengthCm {} 1.4
PetalWidthCm {} 0.2
Species {} Iris-setosa
row {}

Id {} 2
SepalLengthCm {} 4.9
SepalWidthCm {} 3
PetalLengthCm {} 1.4
PetalWidthCm {} 0.2
Species {} Iris-setosa
```

[9]:
```
# Reading xml file through panda library
import pandas as pd

# Load the XML file into a pandas DataFrame
df = pd.read_xml('C:/Users/dnyan/FODS Assignments/Datasets/iris.xml')

# can also use, it doesn't require lxml library
# df = pd.read_xml('C:/Users/dnyan/FODS Assignments/Datasets/iris.xml', parser='etree')


# Display the DataFrame
print(df.head())
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

[11]:
```
# Create a DataFrame for the Iris dataset
data = {
    'sepal_length': [5.1, 4.9, 4.7, 4.6],
    'sepal_width': [3.5, 3.0, 3.2, 3.1],
    'petal_length': [1.4, 1.4, 1.3, 1.5],
    'petal_width': [0.2, 0.2, 0.2, 0.2],
    'species': ['setosa', 'setosa', 'setosa', 'setosa']
}

df = pd.DataFrame(data)
```

[12]:
```
df
```

[12]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa

[13]:
```
# Write the DataFrame to a CSV file
df.to_csv('C:/Users/dnyan/FODS Assignments/Datasets/iris.csv', index=False)
```

[14]:
```
# Write the DataFrame to a TXT file (using tab as a delimiter)
df.to_csv('C:/Users/dnyan/FODS Assignments/Datasets/iris.txt', sep='\t', index=False)
```

[15]:
```
# Write the DataFrame to an XML file
df.to_xml('C:/Users/dnyan/FODS Assignments/Datasets/iris.xml', index=False)
```

References :

https://www.kaggle.com/code/hamelg/python-for-data-10-reading-and-writing-data

https://www.kaggle.com/code/lalitharajesh/iris-dataset-model-building

Conclusion :

In this assignment, we have explored how to read and write different types of datasets using Python. We successfully loaded text files, CSV files, and XML files from local storage. By utilizing libraries such as pandas for CSV manipulation and xml.etree.ElementTree for XML handling, we simplified the process of data management.