Pimpri Chinchwad Education Trust's
Pimpri Chinchwad College of Engineering, Nigdi

Assignment No.1

Problem Statement: Handling Different Dataset Formats

Objective: In this assignment, we'll get hands-on experience with reading and writing various dataset formats, specifically .txt, .csv, and .xml, from both the web and local storage. Our goal is to learn how to load these datasets into memory, process them, and save them back to disk.

Prerequisites:

1. Python Setup**:** Ensure you have Python installed along with necessary libraries such as pandas, xml.etree.ElementTree, and requests (if fetching data from the web).
2. Internet Access**:** Required to load datasets from the web.
3. Text Editor**:** Familiarity with Python file handling is a plus.

Theory:

1. Understanding File Formats:

- **.txt Files:**
  These are plain text files, typically unstructured, and often used for storing logs or simple data.
- **.csv Files:**
  Structured files with comma-separated values, often containing headers. Widely used for data tables and easy sharing of structured information.
- **.xml Files:**
  XML is a markup language allowing hierarchical data structures. It's popular for data exchange between systems due to its readability and structure.

2. Working with Pandas:

- **Pandas Library:**
  A core library for data analysis in Python. It simplifies handling data through its `DataFrame` structure, offering powerful tools for manipulation and transformation.
- **Key Functions:**
  Functions like `pd.read_csv()` and `pd.read_xml()` are essential for reading structured data directly into pandas for easy manipulation.

3. Handling Data with Pandas:

- **Data Preparation and Cleaning:**
  Cleaning and processing raw data is often the first and most critical step in any analysis.

- **Integration Across Formats:**
  Being able to combine different data formats enables you to work with more complex datasets, giving you a broader perspective.
- **Collaboration:**
  Writing data in various formats ensures easier sharing and collaboration across teams or systems

4. Basic Workflow:

1. **Import Libraries:**

   First, import pandas and other necessary libraries.

2. **File Paths:**

   Define paths for both input and output files (local or web).

3. **Reading the Data:**

   - For .csv files:data = pd.read_csv('path/to/file.csv')
   - For .txt files:data = pd.read_csv('path/to/file.txt', sep='your-delimiter')
   - For .xml files:data = pd.read_xml('path/to/file.xml')

4. **Data Processing (Optional):**

   Apply any necessary transformations or cleaning steps.

5. **Writing the Data:**
   a. To save as .csv:

      data.to_csv('path/to/save/file.csv', index=False)

   b. To save as .txt:

      data.to_csv('path/to/save/file.txt', sep='your-delimiter', index=False)

   c. To save as .xml:

      data.to_xml('path/to/save/file.xml', index=False)

6. **Confirmation:**
   Print a success message once the task is completed.

5. Code:

```
import pandas as pd
# Reading a CSV file
csv_file = pd.read_csv('path/to/your/file.csv')
print("CSV Data:")
print(csv_file.head())
# Reading a TXT file
txt_file = pd.read_csv('path/to/your/file.txt', sep='\t')
print("\nTXT Data:")
print(txt_file.head())
# Reading an XML file
xml_file = pd.read_xml('path/to/your/file.xml')
print("\nXML Data:")
print(xml_file.head())
# Save data to a new CSV file
csv_file.to_csv('path/to/save/new_file.csv', index=False)
# Save data to a new TXT file
txt_file.to_csv('path/to/save/new_file.txt', sep='\t', index=False)
# Save data to a new XML file
xml_file.to_xml('path/to/save/new_file.xml', index=False)
```

References:

- McKinney, W. *Python for Data Analysis*, O'Reilly Media, 2017.
- Pandas Official Documentation: pandas.pydata.org

Conclusion:

Mastering how to read and write different data formats using Python is crucial for any data analysis workflow. The `pandas` library offers a straightforward, efficient way to handle these tasks, allowing seamless management and sharing of data across various formats, enabling more insightful analysis.