Assignment on Pandas Library

1. Introduction to the Pandas Library

Pandas is one of the most powerful and widely used Python libraries for data analysis and manipulation. It provides a flexible and efficient way to handle structured data—such as data from spreadsheets, databases, and CSV files.

The name “pandas” is derived from “Panel Data”, a term used in economics to refer to multi-dimensional structured datasets.

Pandas makes working with large data much easier by offering high-level data structures and tools for cleaning, analyzing, and manipulating data. It is open-source and developed primarily by Wes McKinney in 2008, later joined by Sien Chang. It is a key library in data science, machine learning, finance, and business analysis.

2. Features and Importance of Pandas

Pandas was built on top of the NumPy library, which provides fast mathematical and numerical operations. Thanks to this foundation, Pandas inherits NumPy’s speed and efficiency while adding its own specialized features, such as:
- Easy handling of missing data
- Label-based indexing
- Tools to read/write data from multiple sources (CSV, Excel, SQL, JSON)
- Group-by operations for summarizing data
- Compatibility with other libraries like Matplotlib, SciPy, and Scikit-learn

In short, Pandas combines the simplicity of spreadsheets like Excel with the power of Python programming.

3. Installation of Pandas

There are several ways to install Pandas depending on your system and setup.

3.1 Installing Pandas Using Anaconda

Anaconda is the easiest method for installing Pandas since it comes with all dependencies. To check if Pandas is already installed:
conda list pandas

If not installed, run:
conda install pandas
To upgrade Pandas:
conda update pandas

3.2 Installing Pandas Using pip (PyPI)

You can also install it using Python’s package manager pip:
pip install pandas

3.3 Installation on Linux

For Ubuntu/Debian:
sudo apt-get install python-pandas
For Fedora:
sudo dnf install python-pandas

3.4 Installing from Source (Advanced Users)

You can compile Pandas manually from GitHub:
git clone git://github.com/pandas-dev/pandas.git
cd pandas
python setup.py install

3.5 Verifying Installation

After installation, verify with:
import pandas as pd
print(pd.__version__)

4. Getting Started with Pandas

To use Pandas, first import it:
import pandas as pd
import numpy as np

The two core data structures in Pandas are:
1. Series – for one-dimensional labeled data
2. DataFrame – for two-dimensional tabular data

5. The Pandas Series

A Series is a one-dimensional labeled array capable of holding any data type. It can be compared to a column in a spreadsheet.

Example 1: Creating a Series
```
import pandas as pd
data = pd.Series([10, 20, 30, 40])
print(data)
```

Output:
```
0 10
1 20
2 30
3 40
dtype: int64
```

Example 2: Custom Index
```
marks = pd.Series([85, 90, 75], index=["Math", "Science", "English"])
print(marks)
```

Output:
```
Math 85
Science 90
English 75
dtype: int64
```

6. The Pandas DataFrame

A DataFrame is a two-dimensional labeled data structure like a spreadsheet or SQL table. It contains rows and columns, where each column is a Pandas Series.

Example:
```
import pandas as pd
data = {
"Name": ["Arun", "Priya", "Kiran"],
"Age": [22, 24, 21],
"Marks": [85, 90, 88]
}
df = pd.DataFrame(data)
print(df)
```

Output:
Name Age Marks
0 Arun 22 85
1 Priya 24 90
2 Kiran 21 88

7. DataFrame Operations

Pandas provides many methods for working with DataFrames:
df.head() - Shows first rows
df.tail() - Shows last rows
df.info() - Displays info
df.describe() - Summary statistics
df["Marks"].mean() - Calculates mean

Example:
high_marks = df[df["Marks"] > 85]
print(high_marks)

8. Indexing and Selection

Pandas allows access using:
- .loc[] (by label)
- .iloc[] (by index position)

Example:
print(df.loc[0])
print(df.iloc[1, 2])
print(df.loc[:, "Age"])

9. Reading and Writing Data

Pandas can read/write multiple file types:

CSV - pd.read_csv(), df.to_csv()
Excel - pd.read_excel(), df.to_excel()
JSON - pd.read_json(), df.to_json()

Example:
df = pd.read_csv("students.csv")

```
df.to_excel("output.xlsx", index=False)
```

10. Hierarchical Indexing (Multi-Level Indexing)

Hierarchical indexing allows multiple levels of indices.

Example:

```
data = pd.Series(
[10, 20, 30, 40],
index=[["Class A", "Class A", "Class B", "Class B"],
["Math", "Science", "Math", "Science"]]
)
print(data)
```

Output:

```
Class A Math 10
Science 20
Class B Math 30
Science 40
```

11. Conclusion

Pandas is a vital tool in data analysis. Its combination of performance, simplicity, and flexibility makes it the backbone of Python-based data science. Whether cleaning, analyzing, or preparing data, Pandas provides all functionalities required for efficient data manipulation.