What is Pandas?

- **Pandas** is a powerful Python library used for data manipulation and analysis.
- It provides data structures like **Series** and **DataFrames** that make it easy to work with structured data.

Getting Started with Pandas

1. Importing Pandas

To use Pandas in your Python programs, you first need to import it:

```
import pandas as pd
```

Here, `pd` is a common alias for Pandas, making it easier to reference in your code.

Pandas Data Structures

1. Series

- A **Series** is like a one-dimensional array or a column in a table.
- It can hold any type of data: integers, strings, floats, etc.

Example:

```
import pandas as pd

data = pd.Series([10, 20, 30, 40])
print(data)
```

Output:

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

This creates a Series with four elements, where the index on the left is automatically assigned.

---

2. DataFrame

- A DataFrame is a two-dimensional, table-like structure with rows and columns.
- Think of it as a spreadsheet or a SQL table.

Example:

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']}

df = pd.DataFrame(data)
print(df)
```

Output:

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35     Chicago
```

This creates a DataFrame with three rows and three columns.

Reading and Writing Data

1. Reading Data from a CSV File

Pandas makes it easy to load data from various file formats, including CSV.

Example:

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.head())
```

1. **`pd.read_csv('filename.csv')` loads the CSV file into a DataFrame.**
2. **`.df.head()` shows the first five rows of the DataFrame.**

2. Writing Data to a CSV File

You can also save a DataFrame to a CSV file.

Example:

```
import pandas as pd

df.to_csv('output.csv', index=False)
```

This saves the DataFrame `df` to a file named `output.csv` without including the index.

Basic Operations on DataFrames

1. Selecting Columns

You can select one or more columns from a DataFrame.

Example:

```python
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']}

df = pd.DataFrame(data)
print(df['Name'])
```

Output:

```
0     Alice
1       Bob
2   Charlie
Name: Name, dtype: object
```

This selects the `Name` column from the DataFrame.

2. Filtering Rows

You can filter rows based on conditions.

Example:

```python
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']}

df = pd.DataFrame(data)
filtered_df = df[df['Age'] > 30]
print(filtered_df)
```

Output:

```
      Name  Age     City
2  Charlie   35  Chicago
```

This filters the DataFrame to include only rows where `Age` is greater than 30.

Handling Missing Data

1. Checking for Missing Data

You can check for missing values in a DataFrame.

Example:

```python
import pandas as pd

data = {'Name': ['Alice', 'Bob', None],
        'Age': [25, None, 35],
        'City': ['New York', 'Los Angeles', None]}

df = pd.DataFrame(data)
print(df.isnull())
```

Output:

```
    Name    Age   City
0  False  False  False
1  False   True  False
2   True  False   True
```

This shows `True` where data is missing (`None`).

2. Filling Missing Data

You can fill in missing values with a specific value.

Example:

```python
import pandas as pd

data = {'Name': ['Alice', 'Bob', None],
        'Age': [25, None, 35],
        'City': ['New York', 'Los Angeles', None]}

df = pd.DataFrame(data)
df_filled = df.fillna('Unknown')
print(df_filled)
```

Output:

```
      Name   Age         City
0    Alice  25.0     New York
1      Bob   NaN  Los Angeles
2  Unknown  35.0      Unknown
```

This fills in missing data with the word `'Unknown'`.

Grouping and Aggregating Data

1. Grouping Data

You can group data by one or more columns to perform aggregation.

Example:

```python
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'Alice'],
        'Age': [25, 30, 35, 25],
        'City': ['New York', 'Los Angeles', 'Chicago', 'New York']}

df = pd.DataFrame(data)
grouped = df.groupby('City').mean()
print(grouped)
```

Output:

```
             Age
City
Chicago     35.0
Los Angeles 30.0
New York    25.0
```

This groups the data by `City` and calculates the average age for each city.

Why Use Pandas?

- Ease of Use: Pandas makes it easy to manipulate and analyze large datasets.
- Versatility: It supports various data formats (CSV, Excel, SQL, etc.).
- Powerful Tools: Pandas provides many built-in functions for data cleaning, analysis, and visualization.

Key Points to Remember

- Series: A one-dimensional array-like structure.
- DataFrame: A two-dimensional, table-like structure.
- Use Pandas to read, write, and manipulate data easily.
- Handle missing data with functions like `isnull()` and `fillna()`.
- Group and aggregate data for summary statistics.

Next Steps

- Practice creating and manipulating DataFrames.
- Explore more advanced operations like merging, joining, and pivoting data.
- Combine Pandas with other libraries like NumPy and Matplotlib for powerful data analysis.