Pandas is a powerful Python library for data manipulation, analysis, and visualization.

Table of Contents

1. Installing and Importing Pandas

```
pip install pandas
```

```
import pandas as pd
```

---

2. Creating DataFrames and Series

Create a DataFrame from a Dictionary

```
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['NY', 'LA', 'SF']
}
df = pd.DataFrame(data)
```

Create a Series

```
s = pd.Series([1, 2, 3, 4])
```

---

3. Reading and Writing Data

Read CSV File

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

Write to CSV

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

Read Excel File

```
df = pd.read_excel('data.xlsx')
```

Read JSON

```
df = pd.read_json('data.json')
```

---

4. DataFrame Overview

```
df.head()        # First 5 rows
df.tail()        # Last 5 rows
df.info()        # Info about DataFrame
df.describe()    # Summary statistics
df.shape         # Shape (rows, columns)
df.columns       # Column names
df.index         # Row indices
```

---

5. Selecting Data

Select Columns

```
df['Name']              # Single column
df[['Name', 'Age']]     # Multiple columns
```

Select Rows by Index

```
df.loc[0]               # Select row by label
df.iloc[0]              # Select row by index
df.loc[0:2]             # Slice rows by label
df.iloc[0:2]            # Slice rows by position
```

---

6. Filtering Data

```
df[df['Age'] > 30]                        # Filter rows
df[(df['Age'] > 25) & (df['City'] == 'NY')]  # Multiple conditions
df.query('Age > 25')                      # Query method
```

---

7. Adding and Modifying Data

Add New Column

```
df['Salary'] = [70000, 80000, 90000]
```

Modify Values

```
df['Age'] = df['Age'] + 1
```

Apply Functions

```
df['Age'] = df['Age'].apply(lambda x: x + 5)
```

---

8. Dropping Data

```
df.drop('Salary', axis=1, inplace=True)  # Drop column
df.drop(1, axis=0, inplace=True)         # Drop row
```

---

9. Sorting Data

```
df.sort_values(by='Age', ascending=False)
```

---

10. Handling Missing Data

Check for Missing Values

```
df.isnull().sum()
```

Drop Missing Values

```
df.dropna()
```

Fill Missing Values

```
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

---

11. Aggregation and Grouping

```
df.groupby('City')['Age'].mean()      # Group by and aggregate
df.groupby('City').agg({'Age': 'max', 'Salary': 'mean'})  # Multiple
aggregations
```

---

12. Merging and Joining DataFrames

Concatenate DataFrames

```
pd.concat([df1, df2], axis=0)  # Vertical (rows)
pd.concat([df1, df2], axis=1)  # Horizontal (columns)
```

Merge DataFrames (SQL-like joins)

```
pd.merge(df1, df2, on='ID')  # Inner join by default
pd.merge(df1, df2, on='ID', how='left')  # Left join
```

---

13. Pivot Tables

```
df.pivot_table(index='City', values='Salary', aggfunc='mean')
```

---

14. Working with Dates

```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
```

---

15. Exporting Data

```
df.to_csv('output.csv')
df.to_excel('output.xlsx')
df.to_json('output.json')
```

---

16. Common DataFrame Operations

Operation	Command
Head / Tail	df.head() / df.tail()
Shape (Rows, Columns)	df.shape
Column Names	df.columns
Row and Column Access	df.loc[row, col] / df.iloc[row, col]
Sorting by Column	df.sort_values(by=’col’)
Filtering	df[df[‘col’] > x]
Drop Columns	df.drop(‘col’, axis=1)
Fill Missing Data	df.fillna(value)
Group By	df.groupby(‘col’)

Operation	Command
Reset Index	df.reset_index(drop=True)

---

17. Visualization with Pandas

```
df['Age'].plot(kind='hist')     # Histogram
df.plot(kind='line')            # Line plot
df.plot(kind='bar')             # Bar plot
```

Tips for Learning Pandas

- Practice with Real Data – Use datasets from Kaggle or CSV files.
- Understand DataFrame Operations – Master filtering, grouping, and aggregation.
- Explore Pandas Documentation – It has extensive resources and examples.
- Combine with Matplotlib/Seaborn – Enhance data visualization.

Pandas is essential for data analysis