---

1. Introduction

- Pandas is a Python library for data manipulation and analysis.
- Data structures: Series (1D), DataFrame (2D).
- Install: `pip install pandas`

```
import pandas as pd
import numpy as np
```

---

2. Data Structures

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

# DataFrame from dictionary
data = {'Name':['Alice','Bob'],'Age':[25,30]}
df = pd.DataFrame(data)
```

Attributes: `.shape`, `.columns`, `.index`, `.dtypes`

---

3. Viewing and Inspecting Data

```
df.head()
df.tail(3)
df.info()
df.describe()
df.shape
df.columns
df.index
df.sample(2)
```

---

4. Selection and Indexing

```
# Columns
df['Name']
df[['Name','Age']]

# Rows
df.iloc[0]
df.iloc[0:2]
df.loc[0]
df.loc[0:1, ['Name','Age']]

# Conditional selection
df[df['Age']>25]
df[(df['Age']>25) & (df['Name']=='Bob')]

# At/ Iat
df.at[0,'Name']
df.iat[0,1]
```

---

5. Adding/Modifying/Deleting Columns

```
df['Salary'] = [50000, 60000]
df['Age_plus_5'] = df['Age']+5
df.drop('Salary', axis=1, inplace=True)
```

---

6. Handling Missing Data

```
df.isnull()
df.dropna()
df.fillna(0, inplace=True)
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

---

7. Data Cleaning

```
df.drop_duplicates(inplace=True)
df.replace('Alice','Alicia', inplace=True)
df['Name'].str.upper()
df['Name'].str.contains('Bob')
```

---

8. Sorting and Ranking

```
df.sort_values(by='Age')
df.sort_values(by='Age', ascending=False)
df.sort_index()
df['Age'].rank()
```

---

9. Aggregation and Statistics

```
df['Age'].sum()
df['Age'].mean()
df['Age'].median()
df['Age'].min()
df['Age'].max()
df['Age'].std()
df['Age'].var()
df.cumsum()
df.value_counts()
df.unique()
df.nunique()
```

---

10. Grouping Data

```
df.groupby('City')['Age'].mean()
df.groupby(['City','Name']).sum()
df.groupby('City').agg({'Age':'mean','Salary':'sum'})
```

---

11. Merging, Joining, Concatenating

```
df1 = pd.DataFrame({'ID':[1,2],'Name':['Alice','Bob']})
df2 = pd.DataFrame({'ID':[1,2],'City':['Delhi','Mumbai']})

# Merge
merged = pd.merge(df1, df2, on='ID', how='inner')

# Concatenate
df_concat = pd.concat([df1, df2], axis=0)
df_concat = pd.concat([df1, df2], axis=1)
```

12. Reshaping and Pivoting

```
df_pivot = df.pivot(index='Name', columns='City', values='Age')
df_melt = pd.melt(df, id_vars=['Name'], value_vars=['Age','Salary'])
df.stack()
df.unstack()
```

13. Working with Dates

```
df['Date'] = pd.to_datetime(['2025-01-01','2025-02-01'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
pd.date_range('2025-01-01','2025-01-10')
```

14. Input and Output

```
# CSV
pd.read_csv('file.csv')
df.to_csv('file.csv', index=False)

# Excel
pd.read_excel('file.xlsx')
```

```
df.to_excel('file.xlsx', index=False)

# SQL
pd.read_sql('SELECT * FROM table', conn)
df.to_sql('table', conn, if_exists='replace')

# JSON
pd.read_json('file.json')
df.to_json('file.json')

# HTML
df_list = pd.read_html('url')
```

---

15. Applying Functions

```
df['Age_plus_10'] = df['Age'].apply(lambda x: x+10)
df.apply(np.sqrt)
df.applymap(lambda x: x*2 if type(x)==int else x)
df['Name'].map(lambda x: x.upper())
```

---

16. Visualization

```
df['Age'].plot()
df['Age'].hist()
df.boxplot(column='Age')
```

---

17. Advanced Topics

```
# Categorical data
df['Category'] = pd.Categorical(df['City'])

# MultiIndex
df_multi = df.set_index(['City','Name'])

# Rolling and expanding
df['Age'].rolling(2).mean()
df['Age'].expanding().sum()
```

```
# Resampling
df.set_index('Date').resample('M').mean()
```

---

18. Performance Tips

- Use vectorized operations instead of loops.
- Use categorical dtype for repeated strings.
- Read large files in chunks with `chunksize`.
- Optimize memory with `.astype()`.

---

19. Practical Examples

- Exploratory Data Analysis (EDA)
- Cleaning messy datasets
- Combining multiple sources (CSV, Excel, SQL)
- Time series analysis

End of Pandas Complete Guide (Data Scientist Edition)