

CP020001 Basic Data Engineering

Basic Pandas

Lecture by: Teerapong Panboonyuen

https://github.com/kaopanboonyuen/CP020001_ComputerProgramming_2026s1
Contact: teerapong.pa@chula.ac.th

Outline



- Introduction to Data Engineering
- Creating DataFrames
- Indexing & Selecting Data (iloc & loc)
- Basic DataFrame Operations
- Handling Missing Data (fillna)
- String Operations in Pandas
- Filtering & Sorting Data
- GroupBy & Aggregation
- Feature Engineering (Creating New Features)
- Saving & Loading Data (CSV, Excel)
- Basic Data Visualization in Pandas
- Joining & Merging DataFrames

Reference



1. **Pandas Library**
 - Official Documentation: <https://pandas.pydata.org/pandas-docs/stable/>
 - Tutorial: <https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python>
2. **DataFrame Concept**
 - Understanding DataFrame: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>
 - Practical Guide to DataFrame: <https://realpython.com/pandas-dataframe/>
3. **Data Engineering Basics**
 - Introduction to Data Engineering: <https://www.dataversity.net/what-is-data-engineering/>
 - Data Engineering in Python: <https://realpython.com/data-engineering-python/>
4. **Thai Politician Dataset (Mockup)**
5. **Pokemon Dataset (Mockup)**

 This lab uses a *mockup dataset created solely for academic purposes*. The data does not represent real Thai politicians.

 Any resemblance to actual individuals or political figures is purely coincidental. This dataset is meant for learning and practice with Python and Pandas only, and should not be used for any real-world or official representations. ☞



What is Pandas?

- Pandas is a **Python library** for working with data.
- It helps us **analyze, clean, and organize data**.
- Very useful in **Data Science and Machine Learning**.
- Official website: Pandas



Why Do We Use Pandas?



- Handle large data easily
- Read data from:
 - CSV files
 - Excel files
 - Databases
- Clean missing or wrong data
- Perform calculations quickly



Saves time and makes data analysis simple!



Main Data Structures in Pandas

1 Series

- One-dimensional data
- Like a single column in Excel

Example:

↔ Code

Age: 18, 19, 20, 21

2 DataFrame

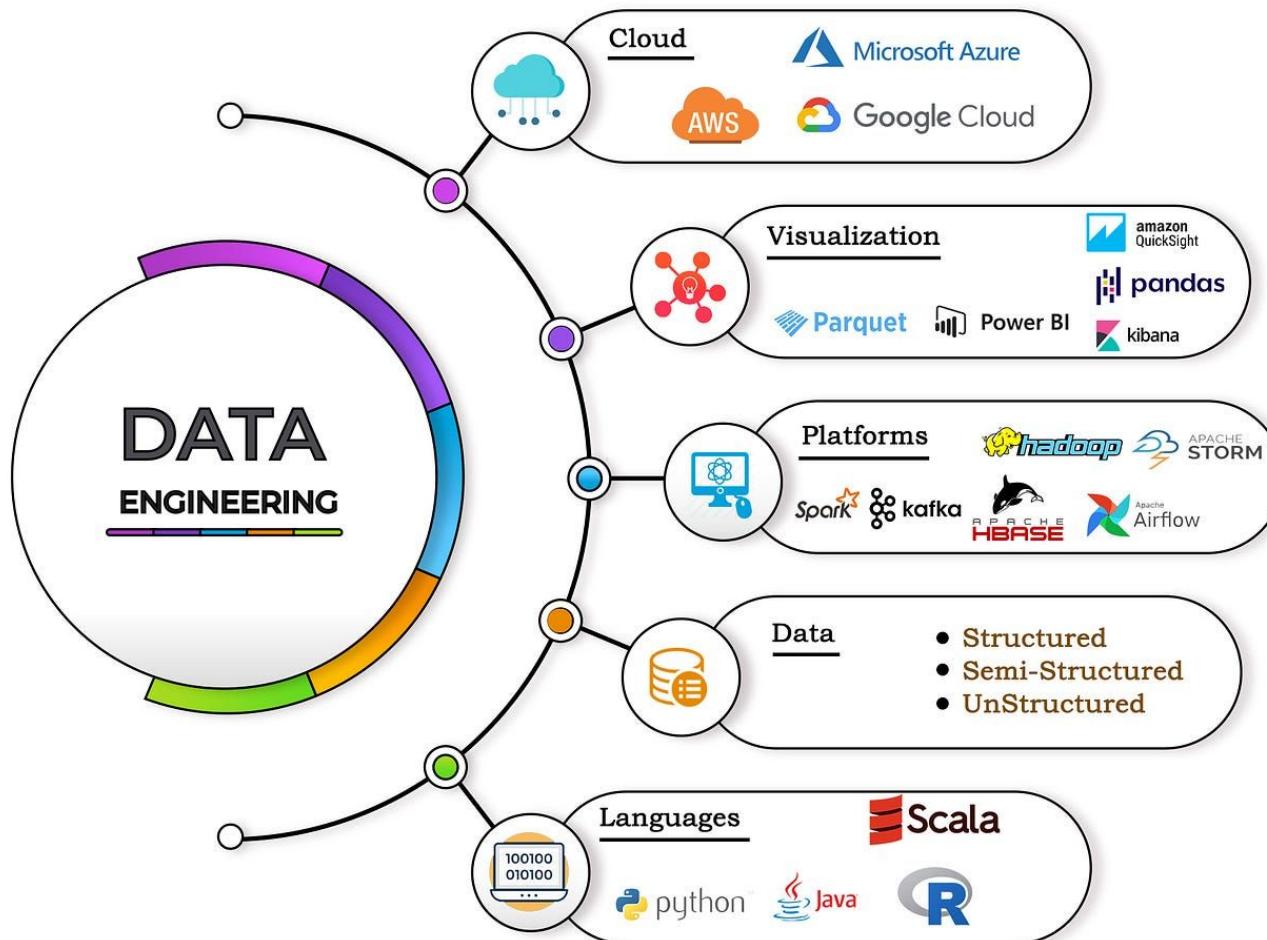
- Two-dimensional data
- Like a full Excel table (rows & columns)

Example:

Name	Age	Marks
Ali	18	85
Sara	19	90

DATA ENGINEERING







Data Engineer





Skills Needed to Become a Data Engineer



SQL



Data Warehousing Solutions



Big Data Platforms



ETL Tools



Python and Java



Data Pipeline Frameworks



Cloud Platforms

Data Scientist



uses statistics and machine learning to make predictions and answer key business questions

Skills - Math, Programming, Statistics



Tech - SQL, Python, R, Cloud

Data Engineer



build and optimize the systems that allow data scientists and analysts to perform their work

Skills - Programming, BigData & Cloud



Tech - SQL, Python, Cloud, Distributed Computing

Data Analyst



deliver value by taking data, communicating the results to help make business decisions

Skills - Communication, Business Knowledge



Tech - SQL, Excel, Tableau

Diagram illustrating the structure of a DataFrame:

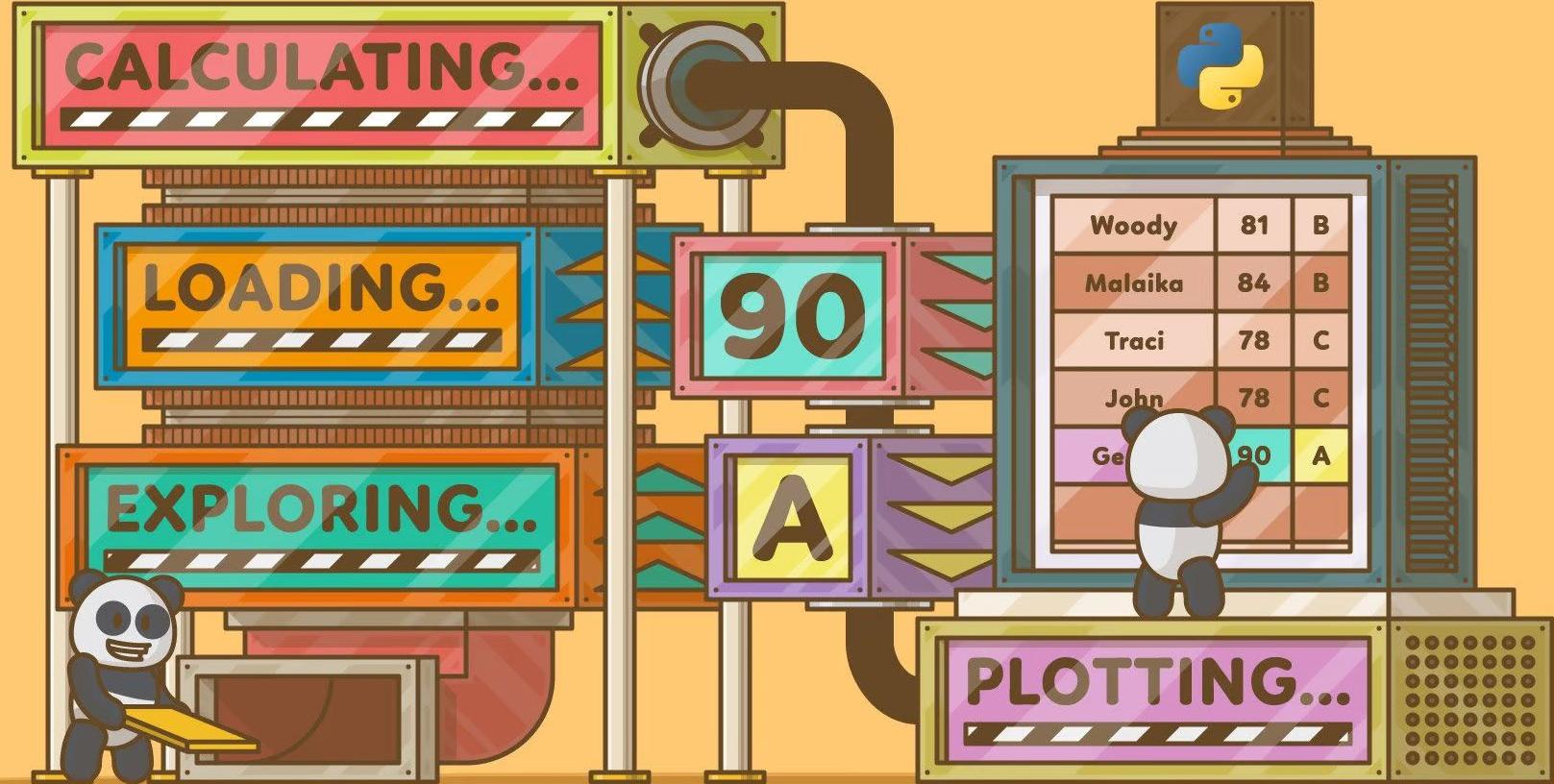
Columns (blue arrows pointing to columns): *Name*, *Team*, *Number*, *Position*, *Age*

Rows (orange arrows pointing to rows): Row 0, Row 1, Row 2, Row 3, Row 4, Row 5, Row 6

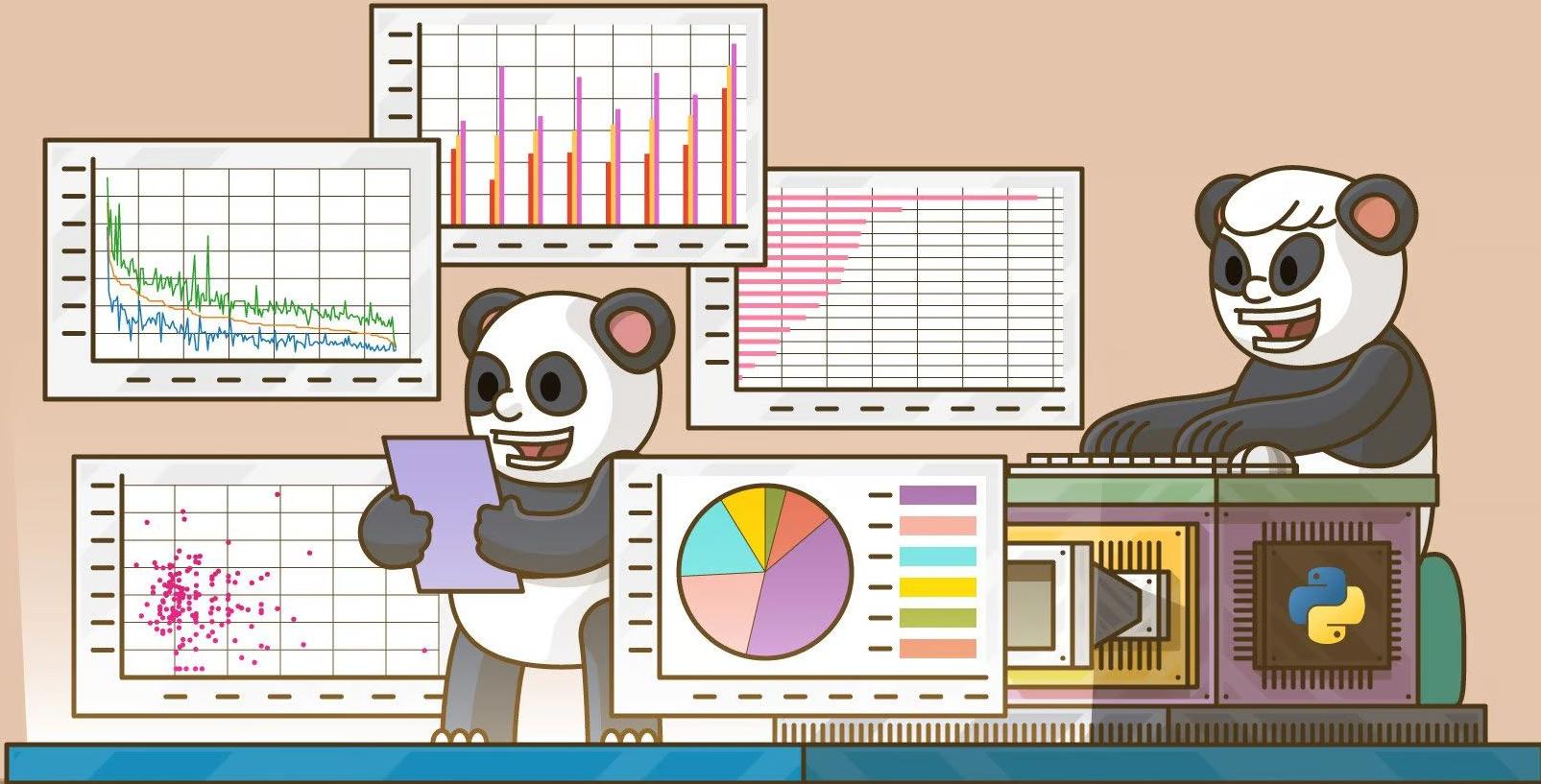
Data (pink box highlighting specific cells): *Team* for row 2, *Number* for row 2, *Position* for row 4, and *Age* for row 5.

	<i>Name</i>	<i>Team</i>	<i>Number</i>	<i>Position</i>	<i>Age</i>
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	Nan	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	Nan
6	Evan Turner	Boston Celtics	11.0	SG	27.0











How to Use Pandas

Step 1: Import Pandas

↔ Python

```
import pandas as pd
```

Step 2: Create a DataFrame

↔ Python

```
import pandas as pd\n\ndata = {\n    "Name": ["Ali", "Sara", "John"],\n    "Age": [18, 19, 20]\n}\n\ndf = pd.DataFrame(data)\nprint(df)
```



Useful Pandas Functions

Function	Purpose
head()	Show first 5 rows
tail()	Show last 5 rows
info()	Show data information
describe()	Show statistics
columns	Show column names



Data Cleaning Example

Remove missing values:

↔ Python

```
df.dropna()
```

Fill missing values:

↔ Python

```
df.fillna(0)
```



Simple Data Analysis

Average age:

</> Python

```
df["Age"].mean()
```

Maximum marks:

</> Python

```
df["Marks"].max()
```

1 Creating DataFrames in Pandas

You can create DataFrames using dictionaries, lists, and lists of dictionaries.

◆ Creating a DataFrame from a Dictionary

```
[ ]    1 import pandas as pd
      2
      3 data_dict = {
      4     'Customer_ID': [101, 102, 103, 104, 105],
      5     'Customer_Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
      6     'Product': ['Laptop', 'Phone', 'Tablet', 'Laptop', 'Phone'],
      7     'Price': [1200, 800, 600, 1300, 850],
      8     'Salary': [5000, 6000, 7000, 4000, 6500]
      9 }
     10
     11 df = pd.DataFrame(data_dict)
     12 df
```

→

	Customer_ID	Customer_Name	Product	Price	Salary
0	101	Alice	Laptop	1200	5000
1	102	Bob	Phone	800	6000
2	103	Charlie	Tablet	600	7000
3	104	David	Laptop	1300	4000
4	105	Eve	Phone	850	6500

⌄ ◆ Creating a DataFrame from a List of Lists

```
▶ 1 data_list = [
 2      [101, 'Alice', 'Laptop', 1200, 5000],
 3      [102, 'Bob', 'Phone', 800, 6000],
 4      [103, 'Charlie', 'Tablet', 600, 7000],
 5      [104, 'David', 'Laptop', 1300, 4000],
 6      [105, 'Eve', 'Phone', 850, 6500]
 7 ]
 8
 9 columns = ['Customer_ID', 'Customer_Name', 'Product', 'Price', 'Salary']
10 df_list = pd.DataFrame(data_list, columns=columns)
11 df_list
```

→

	Customer_ID	Customer_Name	Product	Price	Salary
0	101	Alice	Laptop	1200	5000
1	102	Bob	Phone	800	6000
2	103	Charlie	Tablet	600	7000
3	104	David	Laptop	1300	4000
4	105	Eve	Phone	850	6500

✓  Assignment 1:

Create a DataFrame about students with columns 'Student_ID', 'Name', 'Age', 'GPA', and 'Major'.

```
[ ] 1 # Insert your code here
```

→

	Student_ID	Name	Age	GPA	Major
0	1	John	20	3.5	Engineering
1	2	Jane	22	3.8	Marketing
2	3	Mike	21	3.2	Finance
3	4	Sara	23	3.9	Data Science
4	5	Tom	22	3.6	Computer Science

2 Selecting Data in Pandas (iloc & loc)

◆ Using .iloc[] for Index-Based Selection

```
[ ] 1 # Select first 3 rows  
2 df.iloc[:3]
```

→

	Customer_ID	Customer_Name
0	101	Alice
1	102	Bob
2	103	Charlie

```
[ ] 1 # Select second row  
2 df.iloc[1]
```

→

	1
Customer_ID	102
Customer_Name	Bob
Product	Phone
Price	800
Salary	6000

dtype: object

```
[ ] 1 # Select first 3 rows and first 2 columns  
2 df.iloc[:3, :2]
```



Customer_ID **Customer_Name**

0	101	Alice
1	102	Bob
2	103	Charlie

❖ Using .loc[] for Label-Based Selection

```
[ ] 1 # Select a single row by index  
2 df.loc[2]
```



2

Customer_ID	103
Customer_Name	Charlie
Product	Tablet
Price	600
Salary	7000

dtype: object

```
[ ] 1 # Select rows where Price > 1000  
2 df.loc[df['Price'] > 1000]
```



	Customer_ID	Customer_Name	Product	Price	Salary
0	101	Alice	Laptop	1200	5000
3	104	David	Laptop	1300	4000

```
[ ] 1 # Select specific columns  
2 df.loc[:, ['Customer_Name', 'Product']]
```



Customer_Name Product

0 Alice Laptop

1 Bob Phone

2 Charlie Tablet

3 David Laptop

4 Eve Phone

✓ ⚡ Assignment 2:

Select all customers who bought a laptop and show only 'Customer_Name' and 'Price'.

```
[ ] 1 # Insert your code here
```

→

	Customer_Name	Price
0	Alice	1200
3	David	1300

⌄ 3 Basic DataFrame Operations

```
[ ] 1 df.head()    # First 5 rows
```

→

	Customer_ID	Customer_Name	Product	Price	Salary
0	101	Alice	Laptop	1200	5000
1	102	Bob	Phone	800	6000
2	103	Charlie	Tablet	600	7000
3	104	David	Laptop	1300	4000
4	105	Eve	Phone	850	6500

```
[ ] 1 df.shape    # (rows, columns)
```

→ (5, 5)

```
[ ] 1 df.info() # Data type of each column
```

```
→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Customer_ID      5 non-null      int64  
 1   Customer_Name    5 non-null      object  
 2   Product          5 non-null      object  
 3   Price            5 non-null      int64  
 4   Salary           5 non-null      int64  
dtypes: int64(3), object(2)
memory usage: 332.0+ bytes
```

```
[ ] 1 df.describe() # Summary statistics
```

	Customer_ID	Price	Salary
count	5.000000	5.000000	5.000000
mean	103.000000	950.000000	5700.000000
std	1.581139	291.547595	1204.159458
min	101.000000	600.000000	4000.000000
25%	102.000000	800.000000	5000.000000
50%	103.000000	850.000000	6000.000000
75%	104.000000	1200.000000	6500.000000
max	105.000000	1300.000000	7000.000000



4

Handling Missing Data (fillna)

```
[ ]    1 df.fillna(0) # Replace NaN with 0
      2 df.fillna(df.mean()) # Replace NaN with column mean
      3 df.dropna() # Drop missing rows
```

▼ 5 String Operations in Pandas

```
[ ]    1 df['Customer_Name'].str.lower()    # Convert to lowercase
```



Customer_Name

0	alice
1	bob
2	charlie
3	david
4	eve

dtype: object

```
[ ] 1 df['Customer_Name'].str.upper() # Convert to uppercase
```



Customer_Name

0	ALICE
1	BOB
2	CHARLIE
3	DAVID
4	EVE

dtype: object

```
[ ] 1 df['Customer_Name'].str.contains('Alice') # Check if contains
```



Customer_Name

0	True
1	False
2	False
3	False
4	False

dtype: bool



6

Filtering & Sorting Data

```
[ ]     1 df[df['Price'] > 1000] # Filter rows
```



	Customer_ID	Customer_Name	Product	Price	Salary
0	101	Alice	Laptop	1200	5000
3	104	David	Laptop	1300	4000

```
[ ] 1 df.sort_values('Price', ascending=False) # Sort by price
```



	Customer_ID	Customer_Name	Product	Price	Salary
--	-------------	---------------	---------	-------	--------

3	104	David	Laptop	1300	4000
0	101	Alice	Laptop	1200	5000
4	105	Eve	Phone	850	6500
1	102	Bob	Phone	800	6000
2	103	Charlie	Tablet	600	7000

✗ 7 GroupBy & Aggregation

```
[ ] 1 df.groupby('Product')['Price'].mean() # Average price per product
```



Price

Product

Laptop 1250.0

Phone 825.0

Tablet 600.0

dtype: float64



```
1 df.groupby('Product').agg({'Price': ['mean', 'max', 'min']}) # Multiple aggregations
```



Price



mean max min



Product

Product	mean	max	min
Laptop	1250.0	1300	1200
Phone	825.0	850	800
Tablet	600.0	600	600

✓ 8 Feature Engineering (Creating New Features)

```
[ ] 1 df['Discounted_Price'] = df['Price'] * 0.9 # 10% discount
```

```
[ ] 1 df.head()
```

→

	Customer_ID	Customer_Name	Product	Price	Salary	Discounted_Price	
0	101	Alice	Laptop	1200	5000	1080.0	
1	102	Bob	Phone	800	6000	720.0	
2	103	Charlie	Tablet	600	7000	540.0	
3	104	David	Laptop	1300	4000	1170.0	
4	105	Eve	Phone	850	6500	765.0	

Next steps:

[Generate code with df](#)

[!\[\]\(48cebcbbac721c1b7f9a5efb8d8cb622_img.jpg\) View recommended plots](#)

[New interactive sheet](#)

```
[ ] 1 df['Salary_to_Price'] = df['Salary'] / df['Price'] # Ratio
```



```
1 df.head()
```



	Customer_ID	Customer_Name	Product	Price	Salary	Discounted_Price	Salary_to_Price
0	101	Alice	Laptop	1200	5000	1080.0	4.166667
1	102	Bob	Phone	800	6000	720.0	7.500000
2	103	Charlie	Tablet	600	7000	540.0	11.666667
3	104	David	Laptop	1300	4000	1170.0	3.076923
4	105	Eve	Phone	850	6500	765.0	7.647059

Next steps:

[Generate code with df](#)

[View recommended plots](#)

[New interactive sheet](#)

❖ ♦ Using apply() for Custom Functions

```
[ ] 1 def categorize_salary(salary):  
2     return 'High' if salary > 6000 else 'Low'  
3  
4 df['Salary_Category'] = df['Salary'].apply(categorize_salary)
```

```
[ ] 1 df.head()
```

	Customer_ID	Customer_Name	Product	Price	Salary	Discounted_Price	Salary_to_Price	Salary_Category
0	101	Alice	Laptop	1200	5000	1080.0	4.166667	Low
1	102	Bob	Phone	800	6000	720.0	7.500000	Low
2	103	Charlie	Tablet	600	7000	540.0	11.666667	High
3	104	David	Laptop	1300	4000	1170.0	3.076923	Low
4	105	Eve	Phone	850	6500	765.0	7.647059	High

Next steps:

[Generate code with df](#)

 [View recommended plots](#)

[New interactive sheet](#)



9

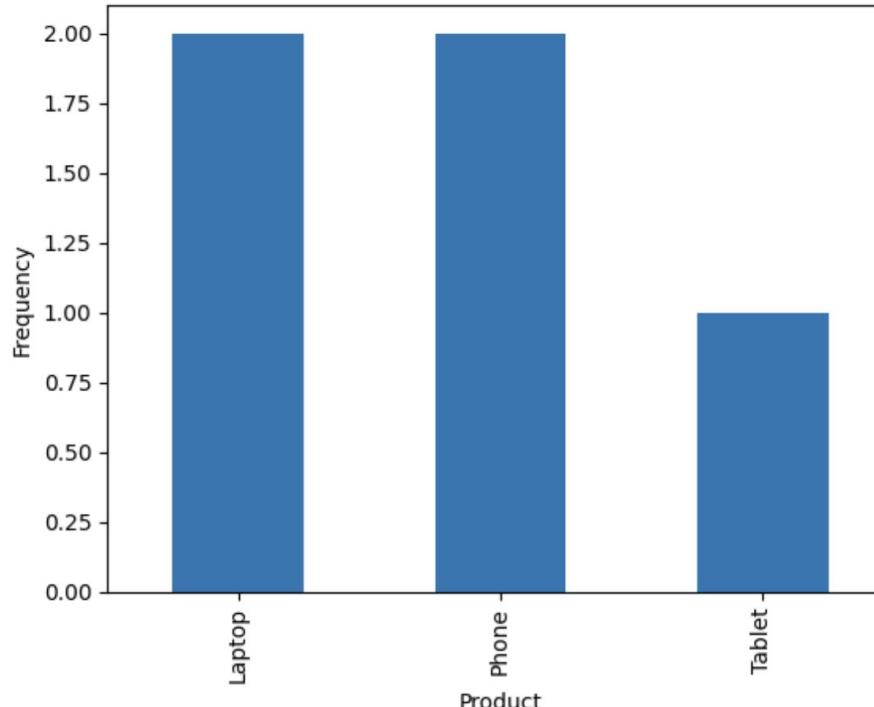
Saving & Loading Data

```
[ ]    1 df.to_csv('data.csv', index=False) # Save  
      2 df = pd.read_csv('data.csv') # Load
```

10 Basic Data Visualization

```
[ ]    1 import matplotlib.pyplot as plt  
2  
3 df['Price'].plot(kind='hist') # Histogram  
4 df['Product'].value_counts().plot(kind='bar') # Bar chart
```

→ <Axes: xlabel='Product', ylabel='Frequency'>



1 1 Joining & Merging DataFrames

```
[ ] 1 df1 = pd.DataFrame({'Customer_ID': [101, 102, 103], 'Region': ['North', 'South', 'East']})  
2 df2 = pd.merge(df, df1, on='Customer_ID', how='left')  
3 df2
```

Customer ID Customer Name Product Price Salary Discounted Price Salary to Price Salary Category Region

	Customer_ID	Customer_Name	Product	Price	Salary	Discounted_Price	Salary_to_Price	Salary_Category	Region	grid icon
0	101	Alice	Laptop	1200	5000	1080.0	4.166667	Low	North	grid icon
1	102	Bob	Phone	800	6000	720.0	7.500000	Low	South	bar chart icon
2	103	Charlie	Tablet	600	7000	540.0	11.666667	High	East	edit icon
3	104	David	Laptop	1300	4000	1170.0	3.076923	Low	NaN	
4	105	Eve	Phone	850	6500	765.0	7.647059	High	NaN	

Next steps:

[Generate code with df2](#)

[View recommended plots](#)

[New interactive sheet](#)

✗ 🔒 Final Assignment:

- Create a new feature Discounted_Price (Price with a 20% discount).
- Filter customers who have a salary above 6000 and sort them by price (descending order).

```
[ ] 1 # Insert your code here
```

→

	Customer_ID	Customer_Name	Product	Price	Salary	Discounted_Price	Salary_to_Price	Salary_Category	
4	105	Eve	Phone	850	6500	680.0	7.647059	High	
2	103	Charlie	Tablet	600	7000	480.0	11.666667	High	 

Next steps:

[Generate code with df_filtered](#)

[View recommended plots](#)

[New interactive sheet](#)

Lab: Thai Politicians Assignment

NIKKEI Asia

Log In Subscribe

World ▾ Trending ▾ Business ▾ Markets ▾ Tech ▾ Politics ▾ Economy ▾ Features ▾ Opinion ▾ Life & Arts ▾ Podcast ▾

Thailand goes to the polls on May 14, with a cast of leadership hopefuls aiming to unseat Prime Minister Prayuth Chan-ocha, right. © Nikkei montage/Source photos by Reuters and Getty Images

Homework: Pokemon

