Importance ■■■■■ most asked ■■■□□ often ■□□□□ rarely

Creating a Series

A **Series** is a one-dimensional labelled array: values plus an **index**. Default index: 0, 1, 2, …

```
import pandas as pd
s = pd.Series([10, 20, 30], index=['a', 'b', 'c'])
print(s)
```

```
a    10
b    20
c    30
dtype: int64
```

FROM	CODE	INDEX
List	`pd.Series([10, 20, 30])`	0, 1, 2
Dictionary	`pd.Series({'Maths': 90, 'CS': 95})`	the keys
Scalar	`pd.Series(5, index=[1, 2, 3])`	5 repeated
NumPy array	`pd.Series(np.array([2, 4, 6]))`	0, 1, 2

Attributes of s: `s.index`, `s.values` → `array([10, 20, 30])`, `s.size` → 3, `s.shape` → `(3,)`, `s.dtype` → int64.

! Shape of a Series is a tuple with a comma: **(3,)**. Writing 3 or (3) loses the mark.

Series indexing, head and tail

CODE, ON S ABOVE	GIVES
`s['b']`	20
`s.iloc[0]`	10 (by position)
`s.loc['a':'b']`	a 10, b 20 (label slice **includes** the end)
`s.iloc[0:2]`	a 10, b 20 (position slice **excludes** the end)
`s[s > 15]`	b 20, c 30
`s.head(2)` · `s.tail(1)`	first 2 rows · last 1 row

`head()` and `tail()` give **5** rows when no number is given. They work the same on a DataFrame.

! With name labels, write **s.iloc[0]** for "the first value". Old papers write `s[0]`; pandas 2 warns it is being removed.

Creating a DataFrame

A **DataFrame** is a two-dimensional table: a row index, column names, and a column per Series.

```
df = pd.DataFrame({'Name': ['Aman', 'Riya', 'Dev'],
                   'Marks': [91, 78, 62],
                   'Class': ['12A', '12B', '12A']})
print(df)
```

```
   Name  Marks Class
0  Aman     91   12A
1  Riya     78   12B
2   Dev     62   12A
```

FROM	KEYS BECOME
Dictionary of lists or Series	column names
List of dictionaries	column names; a missing key gives NaN

```
pd.DataFrame([{'x': 1, 'y': 2}, {'x': 3, 'z': 4}])
```

```
   x    y    z
0  1  2.0  NaN
1  3  NaN  4.0
```

Add, delete, rename

DO	CODE
Add a column	`df['Grade'] = ['A', 'B', 'C']`
Add a row	`df.loc[3] = ['Sara', 85, '12B', 'A']`
Delete a column	`df.drop('Grade', axis=1)` · `del df['Grade']`
Delete a row	`df.drop(1, axis=0)` (by index label)
Rename	`df.rename(columns={'Marks': 'Score'})` · `index={0: 'r1'}`

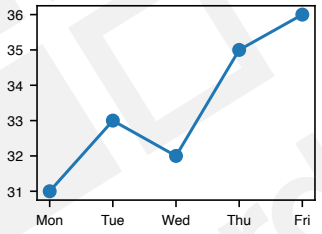
axis=0 means rows, **axis=1** means columns. `drop()` and `rename()` return a **new** DataFrame; add `inplace=True` to change `df` itself.

! "`df.drop('Grade', axis=1)` deletes the column permanently" is **false**: without `inplace=True`, df is unchanged.

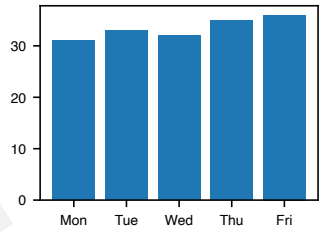
Plotting with Matplotlib

Each chart was drawn by running this data and the line under the chart, then `plt.show()`. Shown small to fit the page.

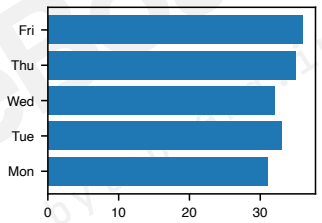
```
import matplotlib.pyplot as plt
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
temp = [31, 33, 32, 35, 36]
marks = [45, 52, 58, 61, 63, 67, 68, 70,
         72, 74, 75, 78, 81, 84, 88, 93]
```



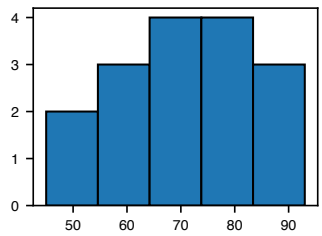
Line chart: change over time
`plt.plot(days, temp, marker='o')`



Bar graph: compare categories
`plt.bar(days, temp)`



Horizontal bar graph: same, bars sideways
`plt.barh(days, temp)`



Histogram: how many values fall in each range
`plt.hist(marks, bins=5, edgecolor='black')`

CUSTOMISE	CODE
Axis names · title	`xlabel()` `ylabel()` · `title()`
Legend	`label='…'` in the plot call, then `legend()`
Marks on an axis	`xticks([0, 1, 2], ['Mon', 'Tue', 'Wed'])`
Line style	`color`, `marker`, `linestyle`, `linewidth`
Save · show	`savefig('file.png')` before `show()`

	BAR GRAPH	HISTOGRAM
Shows	categories you name	counts in ranges (bins)
Bars	with gaps; you give the heights	touching; it counts the heights

! "Show month names on the x-axis" → **xticks()**, not xlabel(). In `barh()` the second argument is the bar's **length**, and `height=` is its thickness.

loc, iloc, Boolean indexing

CODE, ON DF ABOVE	GIVES
`df['Name']`	the Name column, as a Series
`df.loc[1, 'Name']`	Riya
`df.loc[0:1]`	rows 0 **and** 1 (label slice includes the end)
`df.iloc[0:1]`	row 0 only (position slice excludes the end)
`df.loc[:, ['Name', 'Marks']]`	all rows, two columns
`df[df['Marks'] > 70]`	Aman and Riya
`df[(df['Marks'] > 70) & (df['Class'] == '12A')]`	Aman

! First 4 rows of a 0, 1, 2 … index: `df.loc[0:3]`, `df.iloc[0:4]` or `df.head(4)`. `df.loc[0:4]` gives **5** rows. Join conditions with **&** and **|**, each in brackets, not `and`/`or`.

DataFrame attributes and iteration

ATTRIBUTE, ON DF ABOVE	GIVES
`df.shape` · `df.size`	`(3, 3)` · 9 (rows × columns)
`df.ndim`	2
`df.columns` · `df.index`	column names · row labels
`df.dtypes`	Name object, Marks int64, Class object
`df.T`	transpose: rows become columns
`df.empty`	True only for a DataFrame with no data

```
for i, row in df.iterrows():
    print(i, row['Name'], row['Marks'])
```

```
0 Aman 91
1 Riya 78
2 Dev 62
```

`iterrows()` gives each row's index and the row; `items()` gives each column's name and the column.

Maths on Series

Operations match values **by index label**, not by position.

```
s1 = pd.Series([1, 2, 3], index=['a', 'b', 'c'])
s2 = pd.Series([10, 20, 30], index=['b', 'c', 'd'])
print(s1 + s2)
print(s1.add(s2, fill_value=0))
```

```
a     NaN
b    12.0
c    23.0
d     NaN
dtype: float64
a     1.0
b    12.0
c    23.0
d    30.0
dtype: float64
```

! In "write the output", list **every** label from both Series in order, NaN for the unmatched ones, and end with **dtype: float64** (NaN makes it float). `s * 2` doubles each value and stays int64.

CSV files

```
df.to_csv('marks.csv', index=False)
df2 = pd.read_csv('marks.csv')
```

```
Name,Marks,Class
Aman,91,12A
Riya,78,12B
Dev,62,12A
```

to_csv() writes a DataFrame to a file; without `index=False` the row index is saved as an extra first column. **read_csv()** reads a CSV into a DataFrame; the first line becomes the column names (`header=None` if it has none). `sep=';'` for another separator.

Series vs DataFrame

	SERIES	DATAFRAME
Dimensions	one (a column)	two (rows and columns)
Data types	one type	each column its own
Size	fixed: size immutable	can grow: size mutable
Values	both value mutable	

Pandas (Panel Data) is the library for data analysis; **Matplotlib** (pyplot) draws charts. Import them as `pd` and `plt`.

What's the output?

On s and df above. Cover the right column and answer; every result was run in pandas 2.2.

PRINT(…)	OUTPUT
`df.shape`	(3, 3)
`df.loc[2, 'Class']`	12A
`df.iloc[1, 0]`	Riya
`df.columns[1]`	Marks
`len(df)`	3
`s.loc['b':'c'].size`	2
`(s * 2)['c']`	60
`s.iloc[-1]`	30
`df[df['Marks'] < 80].shape`	(2, 3)

Find the error

WRONG	RIGHT
`pd.series([1, 2])`	`pd.Series`
`pd.DataFrame({'a': [1, 2], 'b': [3]})`	lists of equal length
`df.drop('Marks')`	`axis=1` for a column
`df.loc[0, 1]`	`df.iloc[0, 1]` or `df.loc[0, 'Marks']`
`df[df['Marks'] > 70 and df['Class'] == '12A']`	`&`, each condition in `( )`
`df.rename({'Marks': 'Score'})`	`columns={…}`: else nothing changes
`plt.xlable('Month')`	`plt.xlabel`