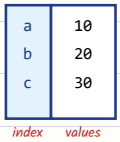
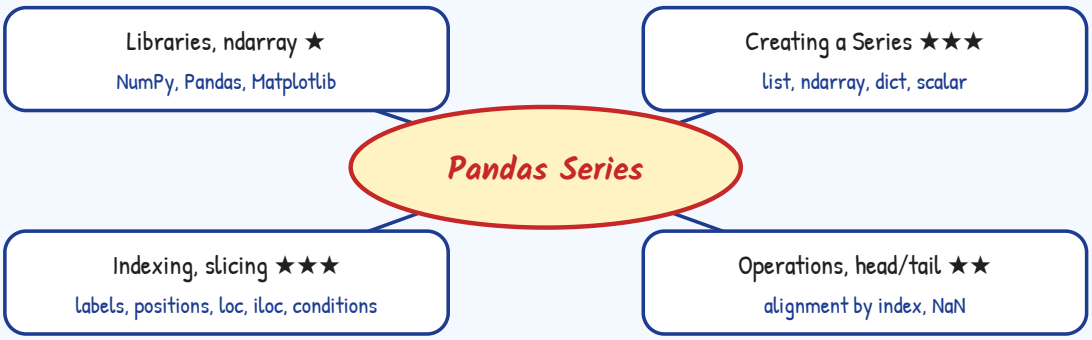
FREE SAMPLE | Buy the complete pack at notesbazar.in | page 1 of 12

Chapter 1
Pandas Series



Python libraries · NumPy ndarray · creating a Series (list, ndarray, dictionary, scalar) · attributes · indexing, slicing · head/tail · operations

Chapter map (★ = asked most often in the board exam)



Syllabus and weightage

Unit 1 **Data Handling using Pandas and Data Visualization** carries **25 marks**. Series: introduction to Python libraries (Pandas, Matplotlib); data structures in Pandas; **creation of Series** from ndarray, dictionary, scalar value; **mathematical operations**; **head() and tail()**; **selection, indexing and slicing**.

Board paper	Items from this chapter
2024	Q10, Q12, Q20, Q22
2025	Q6, Q9, Q13, Q15, Q22(b), Q28(a), Q30(b)
2026	Q1, Q14, Q15, Q22(a), Q28, Q30(a)

★ **Exam tip:** In output questions, always print the **index and value** in two columns and end with the **dtype** line (`dtype: int64`, `float64` if any NaN appears, `object` for text). Adding two Series matches **labels**, not positions; unmatched labels give **NaN**.

In one line: a **Series** is a **one-dimensional labelled array** of values of one type; it is created with `pd.Series(data, index=…)` from a list, ndarray, dictionary (keys become the index) or scalar (repeated for every label); elements are reached by label (`s['b']`, `s.loc`) or position (`s.iloc`), sliced, filtered with conditions and combined with arithmetic that aligns on the index.

1.1 Python libraries for data ★

Library	Use
NumPy (Numerical Python)	fast ndarray (n-dimensional array) of one data type; mathematical operations on whole arrays
Pandas (PANel DAta)	data analysis on tabular data: Series (1-D) and DataFrame (2-D); reading/writing CSV, cleaning, grouping
Matplotlib	plotting graphs: line, bar, histogram (Chapter 4)

Install with `pip install pandas` (NumPy comes with it). Import with the usual aliases:

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

Characteristics of Pandas (CBSE 2026): works with tabular data as Series and DataFrames; quick reading/writing (CSV, Excel); powerful filtering, sorting, grouping; handles missing data (NaN); fast for large data; labelled axes.

Why Pandas when NumPy exists? A NumPy array needs **homogeneous** data and uses only integer positions; Pandas allows labels (index, column names), heterogeneous columns and easy handling of missing values and tabular files.

> **Definition** ✎ A **Series** is a one-dimensional array containing a sequence of values of any one data type (int, float, string, ...) together with an associated array of **labels** called its **index**. If no index is given, the default index is **0, 1, 2, ...** (numeric).

1.2 Creating a Series ★★★

```
s1 = pd.Series([10, 20, 30])                 # from a list
s2 = pd.Series([10, 20, 30], index=['a','b','c'])
s3 = pd.Series(np.array([1.5, 2.5, 3.5]))    # from an ndarray
s4 = pd.Series({'Jan':31, 'Feb':28, 'Mar':31})  # from a dict
s5 = pd.Series(7, index=['x','y','z'])        # from a scalar
print(s2); print(s4); print(s5)
```

Output

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

Output

```
Jan    31
Feb    28
Mar    31
dtype: int64
```

```
Output
x    7
y    7
z    7
dtype: int64
```

- From a **dictionary**, the **keys become the index** and the values become the data.
- From a **scalar**, the index must be given; the value is repeated for every label.
- The number of index labels must equal the number of values (ValueError otherwise).
- `pd.Series(np.array([10,20,30,40,50]), index=['A','B','C','D','E'])` gives a Series from an ndarray with custom labels (CBSE 2025).

Empty Series

```
MT = pd.Series(dtype=float)
print(MT.empty)      # True
```

Naming

```
s = pd.Series([3, 8, 1], index=['p','q','r'])
s.name = 'Marks'          # name of the Series
s.index.name = 'Roll'     # name of the index
s.index = ['P', 'Q', 'R'] # change all labels
```

1.3 Attributes of a Series ★★

Attribute	Gives	For s2
`s.index`	the labels	Index(['a','b','c'])
`s.values`	the data as an ndarray	[10 20 30]
`s.dtype`	data type	int64
`s.shape`	tuple of size	(3,)
`s.size`	number of elements	3
`s.ndim`	number of dimensions	1
`s.empty`	True if no elements	False
`s.name`, `s.index.name`	names	None

Size of a Series (NCERT Q4) = number of elements; it is **fixed** (size-immutable) although values can change. Size of a DataFrame = rows × columns.

1.4 Accessing elements ★★★

```
s2['b']        → 20      # by label
s2.loc['b']    → 20      # label (explicit)
s2.iloc[1]     → 20      # by position
s2[['a','c']]            # several labels → Series
```

Slicing ★★★

Slice	End point

by position `s[0:2]`, `s.iloc[0:2]`	excluded (like lists)
by label `s['a':'c']`, `s.loc['a':'c']`	included

```
d = pd.Series([10,20,30,40,50], index=list('abcde'))
print(d[0:3])          # a 10, b 20, c 30
print(d['b':'d'])      # b 20, c 30, d 40
print(d[d > 30])       # conditional: d 40, e 50
print(d[::-1])         # reversed
```

head() and tail() ★★

```
s = pd.Series(range(1, 11))
s.head()      # first 5 rows (default n = 5)
s.head(3)     # first 3
s.tail(3)     # last 3: 7→8, 8→9, 9→10
```

✗ **Don't do this:** `s.head[0:5]` is wrong: head is a method, called with round brackets (CBSE 2024 Q12).

Adding/changing values

```
rr = pd.Series({'One':1,'Two':2,'Three':3,'Seven':7})
rr['Four'] = 4        # new label added at the END
rr['One'] = 10        # existing label: value changed
```

1.5 Mathematical operations ★★★

```
s2 * 2        # a 20, b 40, c 60   (vectorised)
s2 + 5        # a 15, b 25, c 35
a = pd.Series([1,2,3], index=['a','b','c'])
b = pd.Series([10,20,30], index=['b','c','d'])
print(a + b)
print(a.add(b, fill_value=0))
```

Output

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

Output

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

Operations are done on **matching labels**; a label present in only one Series gives **NaN** (Not a Number), which turns the dtype into float64. Methods `add(), sub(), mul(), div()` accept `fill_value` to treat missing values as a number.

Series vs NumPy ndarray

A Series has labels and aligns data on labels in operations; an ndarray uses only positions and needs arrays of the same length for element-wise operations.

NCERT exercises (Ch 2, Q1–8), solved ★★

Q1. What is a Series and how is it different from a 1-D array, a list and a dictionary?

> **Ans.** A Series is a 1-D labelled array of one data type. Unlike a **1-D ndarray**, it has custom labels (and aligns on them); unlike a **list**, operations are vectorised and the values are of one type with an index; unlike a **dictionary**, the order is fixed, labels can repeat, and positional access/slicing and mathematical operations are possible.

Q2–3. DataFrame vs 2-D array; relation of DataFrames to Series. — See Chapter 2 (a DataFrame is a collection of Series sharing one index, each column being a Series).

Q4. Size of a Series / DataFrame. — Series: number of elements; DataFrame: rows × columns (section 1.3).

Q5. Create: (a) EngAlph, 26 alphabets, default index (b) Vowels, labels a, e, i, o, u, all zero; check if empty (c) Friends from a dictionary (first name: roll no.) (d) MTseries, empty; check (e) MonthDays from a numpy array, labels 1–12.

> **Ans.**

```
import pandas as pd, numpy as np, string
EngAlph = pd.Series(list(string.ascii_lowercase))
Vowels = pd.Series(0, index=['a','e','i','o','u'])
print(Vowels.empty)                          # False
Friends = pd.Series({'Amit':12, 'Bela':5, 'Chirag':18,
                     'Disha':7, 'Esha':21})
MTseries = pd.Series(dtype=object)
print(MTseries.empty)                        # True
MonthDays = pd.Series(np.array([31,28,31,30,31,30,31,31,30,31,30,31]),
                      index=range(1, 13))
```

Q6. (a) Set all values of Vowels to 10 (b) divide them by 2 (c) Vowels1 = [2,5,6,3,8] with labels a, e, i, o, u (d) Vowels3 = Vowels + Vowels1 (e) subtract, multiply, divide (f) relabel Vowels1 as A, E, I, O, U.

> **Ans.**

```
Vowels[:] = 10;  print(Vowels)        # all 10
Vowels = Vowels / 2;  print(Vowels)   # all 5.0
Vowels1 = pd.Series([2,5,6,3,8], index=['a','e','i','o','u'])
Vowels3 = Vowels + Vowels1            # 7.0 10.0 11.0 8.0 13.0
print(Vowels - Vowels1)               # 3.0 0.0 -1.0 2.0 -3.0
print(Vowels * Vowels1)               # 10.0 25.0 30.0 15.0 40.0
print(Vowels / Vowels1)               # 2.5 1.0 0.833 1.667 0.625
Vowels1.index = ['A','E','I','O','U']
```

Q7. (a) Dimensions, size and values of the five Series (b) rename MTseries as SeriesEmpty (c) name the index of MonthDays monthno and of Friends Fname (d) 3rd and 2nd value of Friends, in that order (e) alphabets 'e' to 'p' of EngAlph (f) first 10 values (g) last 10 values (h) display MTseries.

> **Ans.**

```
for s in [EngAlph, Vowels, Friends, MTseries, MonthDays]:
    print(s.ndim, s.size, s.values)   # ndim 1; sizes 26 5 5 0 12
SeriesEmpty = MTseries                # (or MTseries.rename(...))
MonthDays.index.name = 'monthno'
Friends.index.name = 'Fname'
print(Friends.iloc[[2, 1]])           # Chirag 18, Bela 5
print(EngAlph[4:16])                  # positions 4..15 = e..p
print(EngAlph.head(10))               # a..j
print(EngAlph.tail(10))               # q..z
print(MTseries)                       # Series([], dtype: object)
```

Q8. (a) Months 3 through 7 from MonthDays (b) MonthDays in reverse order.

Ans.

```
print(MonthDays.loc[3:7])     # label slice: 3,4,5,6,7 → 31 30 31 30 31
print(MonthDays[::-1])        # 12, 11, … 1
```

(`MonthDays[2:7]` also gives months 3–7 because a plain [ ] slice with integers is positional.)

Practical-list programs

```
# elements above the 75th percentile
s = pd.Series([12, 45, 33, 78, 56, 90, 23])
print(s[s > s.quantile(0.75)])        # 78, 90
# Series from a dictionary and an ndarray
print(pd.Series({'a':1, 'b':2}))
print(pd.Series(np.arange(5, 30, 5)))  # 5 10 15 20 25
```

CBSE PYQs 2024–2026 (fully solved) ★★★

CBSE 2025 Default index type for a Pandas Series if not explicitly specified? (A) String (B) List (C) Numeric (D) Boolean

Ans. (C) Numeric: 0, 1, 2, …

CBSE 2025 Data structure for one-dimensional labelled data in Pandas? (A) Integer (B) Dictionary (C) Series (D) DataFrame

Ans. (C) Series.

CBSE 2025 Statement to select the element with index `points` from Series `ser`: (A) ser.element(points) (B) ser.select(points) (C) ser[points] (D) ser.show(points)

Ans. (C) ser[points].

CBSE 2025 Which library defines an ndarray? (A) pandas (B) numpy (C) matplotlib (D) scipy

Ans. (B) numpy.

CBSE 2026 True or False: In a Pandas Series, the positional index can be a string or an integer.

Ans. False. Positional indices are always integers 0, 1, 2…; only labels can be strings.

CBSE 2026 Creating a Series from a dictionary: (A) values become indices (B) keys become values (C) keys become the indices (D) default integer indices

Ans. (C) The keys of the dictionary become the index.

CBSE 2026 `ser1[start:end]` with positional indices: (A) end included (B) end excluded (C) both excluded (D) start excluded 1

Ans. (B) The element at the end position is excluded.

CBSE 2024 What will be the output of the following Python code? `import pandas as pd; dd= { 'One':1, 'Two':2, 'Three':3, 'Seven':7}; rr=pd.Series(dd); rr['Four']=4; print(rr)` 1

(i)
```
One      1
Two      2
Three    3
Seven    7
dtype: int64
```

(ii)
```
One      1
Two      2
Three    3
Four     4
Seven    7
dtype: int64
```

(iii)
```
Four     4
One      1
Two      2
Three    3
Seven    7
dtype: int64
```

(iv)
```
One      1
Two      2
Three    3
Seven    7
Four     4
dtype: int64
```

Ans. A new label is appended at the end: (iv)

Output
```
One      1
Two      2
Three    3
Seven    7
Four     4
dtype: int64
```

CBSE 2024 Which will NOT show the first five rows of Series S1? (i) S1[0:5] (ii) S1.head() (iii) S1.head(5) (iv) S1.head[0:5] 1

Ans. (iv): head is a method; `head[0:5]` raises TypeError.

CBSE 2024 Correct the errors: `import Pandas as pd` / `stud= ['Name':'Ramya','Class':11,'House':'Red']` / `s=p.Series(s)` / `print(s)` 2

Ans.

```
import pandas as pd                                  # lower-case pandas
stud = {'Name':'Ramya', 'Class':11, 'House':'Red'}   # { } for a dict
s = pd.Series(stud)                                  # pd, and stud
print(s)
```

Output
```
Name      Ramya
Class        11
House       Red
dtype: object
```

CBSE 2024 Output: `com=pd.Series([45,12,15,200], index= ['mouse','printer','webcam','keyboard'])` / `print(com[1:3])` 2

Ans. Positions 1 and 2 (3 excluded):

Output
```
printer    12
webcam     15
dtype: int64
```

CBSE 2025 (b) Explain how elements of a Series can be accessed using slicing, with an example. 2

Ans. For `data = pd.Series([10,20,30,40,50], index=['a','b','c','d','e'])`: positional slicing `data[0:3]` or `data.iloc[0:3]` → a, b, c (end excluded); label slicing `data['b':'d']` or `data.loc['b':'d']` → b, c, d (end included); conditional `data[data > 30]` → d, e.

CBSE 2025 (a) Rewrite Rohit's code correctly: `import pandas` / `data = [50, 15, 40]` / `series = pd.series(data, Index=['x','y','z'])` / `Print(series)` 2

Ans.

```
import pandas as pd                                  # alias pd
data = [50, 15, 40]
series = pd.Series(data, index=['x', 'y', 'z'])   # Series, index
print(series)                                        # print
```

CBSE 2025 (b) Create a Series from an ndarray of 10, 20, 30, 40, 50 with indices 'A'-'E'. 3

Ans.

```
import pandas as pd
import numpy as np
data = np.array([10, 20, 30, 40, 50])     # or np.arange(10, 60, 10)
s = pd.Series(data, index=['A', 'B', 'C', 'D', 'E'])
print(s)
```

CBSE 2026 (a) Two characteristics of the Pandas library. 2

Ans. Any two: tabular data structures (Series, DataFrame); easy reading/writing of files such as CSV and Excel; powerful filtering, sorting, grouping and calculations; easy handling of missing data; fast on large datasets.

CBSE 2026 (a) Complete: Series from a NumPy array of 'a', 'b', 'c' with indexes 'first', 'second', 'third'. OR (b) Output of `s1=pd.Series([1,2,3],index=['a','b','c']); s2=pd.Series([4,5,6],index=['b','c','d']); print(s1 + s2)` 2

Ans. (a) `import numpy as np; arr = np.array(['a','b','c']); s = pd.Series(arr, index=['first','second','third'])`.

(b) Labels are matched; 'a' and 'd' are in only one Series:

Output

```
a    NaN
b    6.0
c    8.0
d    NaN
dtype: float64
```

(As printed, the paper's code uses `S2` (capital); the board also accepted "NameError" as an answer.)

CBSE 2026 (a) Create and display a Series of days in July, Aug, Sept, Oct (31, 31, 30, 31) with month names as index. 3

Ans.

```
import pandas as pd
s = pd.Series([31, 31, 30, 31], index=['July', 'Aug', 'Sept', 'Oct'])
# or pd.Series({'July':31, 'Aug':31, 'Sept':30, 'Oct':31})
print(s)
```

Practice set (answers at the end of the chapter)

A. Fill in the blanks

1. A Series is a ____-dimensional labelled array.
2. The default index of a Series starts at ____.
3. From a dictionary, the ____ become the index.
4. ____ returns the number of elements of a Series.
5. Label slicing ____ the end label.
6. Missing values are shown as ____.
7. The default number of rows shown by head() is ____.
8. NumPy's array object is called ____.

B. True or False

1. A Series can have duplicate index labels.
2. s.iloc uses labels.
3. A Series created from a scalar needs an index.
4. Adding two Series adds by position.
5. s.shape for 4 elements is (4,).
6. s.empty is True for pd.Series(dtype=int).
7. The size of a Series can be changed by assigning a new label.
8. dtype of a Series with any NaN in integers becomes float64.

C. Very short answer

1. Write a statement to import pandas with alias pd.
2. Create a Series of 3 zeros labelled p, q, r.
3. Output: `pd.Series([5,6,7])[1]`
4. Output: `pd.Series([1,2,3,4]).tail(2).tolist()`
5. How do you name the index of s as "Roll"?
6. Statement to show the last 4 elements of s.
7. Expand NaN.
8. Give two attributes of a Series.
9. Output: `pd.Series([2,4],index=['a','b'])*3`
10. Which method adds two Series treating missing values as 0?

D. Short answer I (2 marks)

1. Create a Series of marks of 4 students with their names as index.
2. Differentiate loc and iloc for a Series.

3. Write the output: `s=pd.Series([10,20,30,40],index=['w','x','y','z']); print(s['x':'z'])`
4. Write the output: `print(s[s>15]*2)` for the same s.
5. Find and correct errors: `import pandas as pd; s=pd.series([1,2],index=['a','b','c'])`
6. Create a Series from an ndarray of the first five even numbers.
7. How is a Series different from a list?
8. What happens when two Series with different labels are added?

E. Short answer II (3 marks)

1. Create a Series Temp of 7 days' temperatures (Mon–Sun); display days with temperature above 30 and the average.
2. Create Series S1 = [5,10,15] labels a–c and S2 = [1,2,3] labels b–d; show S1+S2, S1−S2 and S1.add(S2, fill_value=0).
3. Write the output of `s=pd.Series(range(10,60,10)); print(s.head(2)); print(s.tail(2)); print(s[::-2])`

F. Long answer (5 marks)

1. Explain four ways of creating a Series with code and output.
2. Explain indexing and slicing of a Series using labels, positions, loc, iloc and conditions with examples.

G. HOTS

1. Why does `pd.Series([1,2,3]) + pd.Series([1,2])` contain NaN?
2. For a Series with integer labels 1..5, why is s[1] different from s.iloc[1]?

Case study. A shop stores stock as `Stock = pd.Series({'Pen':120,'Pencil':80,'Eraser':45,'Sharpener':30})`. (i) Display items with stock below 50. (ii) Increase all stock by 10. (iii) Add a new item 'Scale' with 60. (iv) Display the first two items.

Multiple choice: 20 MCQs

1. Pandas Series is:
   (a) three-dimensional (b) two-dimensional (c) one-dimensional (d) zero-dimensional
2. Default index starts at:
   (a) 0 (b) 1 (c) −1 (d) 'a'
3. Keys of a dict passed to Series become:
   (a) values (b) dtype (c) name (d) index
4. Number of elements:
   (a) s.shape (b) s.size (c) s.len (d) s.ndim
5. Label slice s['a':'c'] includes:
   (a) a and b (b) a, b and c (c) b and c (d) only a
6. Positional access:
   (a) s.loc (b) s.label (c) s.at (d) s.iloc
7. s.head() shows by default:
   (a) 5 rows (b) 10 rows (c) 3 rows (d) all rows
8. Unmatched labels in s1+s2 give:
   (a) 0 (b) error (c) NaN (d) None
9. s.ndim for a Series:
   (a) depends (b) 2 (c) 0 (d) 1

10. ndarray belongs to:
    (a) pandas (b) numpy (c) matplotlib (d) math

11. pd.Series(5, index=[1,2,3]) has values:
    (a) 5 only once (b) error (c) 5, 5, 5 (d) 1, 2, 3

12. Check if a Series is empty:
    (a) s.empty (b) s.isempty() (c) len(s)==None (d) s.none

13. Output of pd.Series([2,4,6])[1]:
    (a) 2 (b) 4 (c) 1 (d) 6

14. dtype of pd.Series(['a','b']):
    (a) char (b) str (c) object (d) int64

15. Last 3 elements:
    (a) s.end(3) (b) s[-3] (c) s.last(3) (d) s.tail(3)

16. s.shape for 6 elements:
    (a) (6,) (b) (1,6) (c) 6 (d) (6,1)

17. Elements greater than 50:
    (a) s[s>50] (b) s.filter(50) (c) s.where(50) (d) s(>50)

18. Correct import:
    (a) import Pandas as pd (b) include pandas (c) using pandas (d) import pandas as pd

19. s * 2 multiplies:
    (a) the index (b) every value by 2 (c) only the first value (d) the size

20. Labels of a Series are its:
    (a) dtype (b) values (c) index (d) name

1 (c) · 2 (a) · 3 (d) · 4 (b) · 5 (b) · 6 (d) · 7 (a) · 8 (c) · 9 (d) · 10 (b) · 11 (c) · 12 (a) · 13 (b) · 14 (c) · 15 (d) · 16 (a) · 17 (a) · 18 (d) · 19 (b) · 20 (c)

Assertion–Reason (6)

(a) Both A and R true, R explains A. (b) Both true, R does not explain A. (c) A true, R false. (d) A false, R true.

1. A: s['a':'c'] includes the value at 'c'. R: Label-based slicing includes the end label.
2. A: pd.Series([1,2]) + pd.Series([1,2,3]) has a NaN. R: Series arithmetic aligns on index labels.
3. A: A Series can store values of many types in one element. R: A Series is two-dimensional.
4. A: Creating a Series from a scalar needs an index. R: The scalar is repeated for each index label.
5. A: s.head[0:3] shows 3 rows. R: head() is a method.
6. A: The size of a Series is the number of elements. R: s.ndim is 1 for a Series.

Key: 1 (a) · 2 (a) · 3 (d) · 4 (a) · 5 (d) · 6 (b)

Answers to the practice set

A. 1. one 2. 0 3. keys 4. size 5. includes 6. NaN 7. 5 8. ndarray

B. 1. T 2. F 3. T 4. F 5. T 6. T 7. T (a new label adds an element) 8. T

C. 1. `import pandas as pd` 2. `pd.Series(0, index=['p','q','r'])` 3. 6 4. [3, 4] 5. `s.index.name = 'Roll'` 6. `s.tail(4)` 7. Not a Number 8. size, index (also values, dtype, shape) 9. a 6, b 12 10. `add(..., fill_value=0)`

D2. loc uses labels (end included in slices); iloc uses integer positions (end excluded). **D3.** x 20, y 30, z 40. **D4.** x 40, y 60, z 80. **D5.** `pd.Series([1,2], index=['a','b'])` (Series with capital S; 2 values need 2 labels). **D6.** `pd.Series(np.arange(2, 11, 2))`. **D8.** Matching labels are added; others give NaN.

E1.

```
Temp = pd.Series([29,31,33,28,35,30,32],
       index=['Mon','Tue','Wed','Thu','Fri','Sat','Sun'])
print(Temp[Temp > 30]); print(Temp.mean())   # 31.14
```

E2. S1+S2: a NaN, b 11.0, c 17.0, d NaN; S1–S2: a NaN, b 9.0, c 13.0, d NaN; add(fill_value=0): a 5.0, b 11.0, c 17.0, d 3.0.

E3. head(2): 0 10, 1 20; tail(2): 3 40, 4 50; s[::-2]: 4 50, 2 30, 0 10.

G1. Label 2 exists only in the first Series. **G2.** s[1] uses the label 1 (first element); s.iloc[1] uses position 1 (second element).

Case. (i) `Stock[Stock < 50]` → Eraser 45, Sharpener 30 (ii) `Stock = Stock + 10` (iii) `Stock['Scale'] = 60` (iv) `Stock.head(2)`

- **Last-night summary:** import pandas as pd; Series = 1-D labelled array; default index 0,1,2 (numeric). Create from list, ndarray (np.array), dict (keys → index), scalar (needs index).
- Attributes: index, values, dtype, shape (n,), size, ndim 1, empty, name. Access s['b'], s.loc['b'], s.iloc[1]; slicing by position (end excluded) or label (end included); s[s>30]; head()/tail() default 5.
- Arithmetic aligns on labels → NaN for unmatched (dtype float64); add/sub/mul/div with fill_value. New label assignment appends at the end.