
📘 Pandas Series – Fully Functional Notes

🔹 1. What is a Series?

A **Series** is a **1-dimensional labeled array** in Pandas capable of holding any data type: int, float, string, objects, etc.

```
import pandas as pd

import numpy as np
```

🔹 2. Creating Series

✅ From List

```
data = [10, 20, 30]

s = pd.Series(data)
```

✅ From List with Custom Index

```
s = pd.Series([10, 20, 30], index=["a", "b", "c"])
```

✅ From Dictionary

```
data = {"a": 100, "b": 200, "c": 300}

s = pd.Series(data)
```

✅ From NumPy Array

```
arr = np.array([1, 2, 3])

s = pd.Series(arr)
```

✅ From CSV (Single Column)

```
s = pd.read_csv("file.csv", squeeze=True)
```

---

3. Important Attributes

Attribute	Description
s.size	Number of elements
s.dtype	Data type of elements
s.index	Returns index (labels)
s.values	Returns underlying NumPy array
s.name	Name of the Series
s.is_unique	Returns True if all values are unique

---

4. Useful Methods

Basic Methods

```
s.head(n)        # First n elements
s.tail(n)        # Last n elements
s.sample(n)      # Random n elements
s.unique()       # Unique values
s.nunique()      # Number of unique values
s.value_counts() # Frequency of each unique value
```

Sorting

```
s.sort_values()            # Sort by values (ascending)
s.sort_values(ascending=False)
s.sort_index()             # Sort by index
```

---

5. Mathematical & Statistical Methods

Aggregation

```
s.count()    # Non-null values
s.sum()      # Sum of values
s.prod()     # Product of values
```

```
s.mean()      # Mean
s.median()    # Median
s.mode()      # Most frequent value(s)
s.min()       # Minimum
s.max()       # Maximum
```

🔸 Summary

```
s.describe()
# count, mean, std, min, 25%, 50%, 75%, max
```

🔹 6. Indexing & Selection

✅ Access by Label

```
s['a']
```

✅ Access by Position

```
s[0]
```

✅ Slicing

```
s[1:4]      # By position
s['a':'c']  # By label
```

✅ Using .loc and .iloc

```
s.loc['b']    # Label-based
s.iloc[2]     # Integer position
```

🔹 7. Modifying Series

🔸 Change Value

```
s['a'] = 500
```

🔸 Add New Value

```
s['new'] = 1000
```

🔸 Delete Value

```
s.drop('b')         # Returns a new series
```

```
s.drop('b', inplace=True)  # Modifies in place
```

🔷 8. Python Functional Compatibility

🔶 Works with:

```
len(s)
max(s)
min(s)
sum(s)
'a' in s        # Checks if 'a' is an index
```

🔶 Looping

```
for val in s:
    print(val)
```

🔷 9. Boolean Indexing (Filtering)

```
s[s > 50]              # Values greater than 50
s[(s > 20) & (s < 100)]   # Multiple conditions
```

🔷 10. Handling Missing Values (Extra)

🔶 Detect Missing

```
s.isna()        # Returns Boolean Series
s.notna()
```

🔶 Fill Missing

```
s.fillna(0)
s.fillna(method='ffill')
```

🔶 Drop Missing

```
s.dropna()
```

🔹 11. Series with Custom Functions

🔸 Apply Custom Logic

```
def convert(val):
    return val * 2


s.apply(convert)
# or
s.apply(lambda x: x ** 2)
```

🔹 12. Plotting Series

```
import matplotlib.pyplot as plt
s.plot()                    # Line plot
s.value_counts().plot(kind='bar') # Bar chart
plt.show()
```

🔹 13. Converting Series

🔸 To List / Dict / NumPy

```
s.to_list()
s.to_dict()
s.to_numpy()
```

🔹 14. Comparison and Logic Ops

```
s == 100
s > 50
s.equals(another_series)
```

◆ 15. String Operations (for string series)

```
s.str.upper()

s.str.contains("abc")

s.str.replace("old", "new")
```

◆ 16. DateTime Support (Extra)

```
s = pd.Series(pd.date_range("2024-01-01", periods=5))

s.dt.day

s.dt.month
```
