pandas

January 28, 2026

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# creating series
# creating series from list
s1 = pd.Series([10, 20, 30, 40])
print(s1)

# creating series with custom index
s2 = pd.Series([100, 200, 300], index=["a", "b", "c"])
print(s2)

# creating series from numpy array
arr = np.array([1, 2, 3])
s3 = pd.Series(arr)
print(s3)

# creating series from dictionary
data = {"Math": 85, "Science": 90, "English": 88}
s4 = pd.Series(data)
print(s4)
```

```
0    10
1    20
2    30
3    40
dtype: int64
a    100
b    200
c    300
dtype: int64
0    1
1    2
2    3
dtype: int32
Math       85
Science    90
English    88
```

```
dtype: int64
```

[3]:
```python
# series attributes
s = pd.Series([10, 20, 30])

print(s.index)
print(s.values)
print(s.dtype)
print(s.size)
print(s.ndim)
```

```
RangeIndex(start=0, stop=3, step=1)
[10 20 30]
int64
3
1
```

[ ]:
```python
# Accessing series elements
s = pd.Series([100, 200, 300], index=["x", "y", "z"])

print(s[0])  # by position
print(s["y"])  # by label
print(s[:2])  # slicing
print(s[["x", "z"]])
print(s[s > 150])  # conditional filtering


# updating values
s = pd.Series([10, 20, 30])

s[1] = 200
print(s)

s[s > 20] = 999
print(s)
```

[6]:
```python
# Arithmetic operations
s1 = pd.Series([10, 20, 30], index=["a", "b", "c"])
s2 = pd.Series([1, 2, 3], index=["a", "b", "d"])

print(s1 + s2)  # alignment by index

# Aggregation functions
s = pd.Series([10, 20, 30, 40])

print(s.sum())
print(s.mean())
print(s.median())
```

```
print(s.min())
print(s.max())
print(s.std())
print(s.var())
print(s.count())
```

```
a    11.0
b    22.0
c     NaN
d     NaN
dtype: float64
100
25.0
25.0
10
40
12.909944487358056
166.66666666666666
4
```

[5]:
```
# value counts
s = pd.Series(["apple", "banana", "apple", "orange", "banana"])

print(s.value_counts())


# unique & nunique
s = pd.Series([1, 2, 2, 3, 3, 3])

print(s.unique())
print(s.nunique())
```

```
apple     2
banana    2
orange    1
Name: count, dtype: int64
[1 2 3]
3
```

[7]:
```
# Handling missing data
s = pd.Series([1, 2, np.nan, 4])

print(s.isna())
print(s.notna())

print(s.fillna(0))
print(s.dropna())
```

```
# combining series
s1 = pd.Series([1, 2, 3])
s2 = pd.Series([4, 5, 6])

print(pd.concat([s1, s2]))
```

```
0    False
1    False
2     True
3    False
dtype: bool
0     True
1     True
2    False
3     True
dtype: bool
0    1.0
1    2.0
2    0.0
3    4.0
dtype: float64
0    1.0
1    2.0
3    4.0
dtype: float64
0    1
1    2
2    3
0    4
1    5
2    6
dtype: int64
```

0.0.1 DATAFRAME

[8]:
```
# creating dataframe from dictionary
data = {"Name": ["A", "B", "C"], "Age": [20, 21, 22], "Marks": [80, 85, 90]}

df = pd.DataFrame(data)
print(df)
```

```
  Name  Age  Marks
0    A   20     80
1    B   21     85
2    C   22     90
```

[9]:
```
# dataframe from a list of dictionaries
# dataframe from list of dicts
```

```
data = [{"Name": "A", "Age": 20}, {"Name": "B", "Age": 21}, {"Name": "C", "Age":
 ↪ 22}]

df = pd.DataFrame(data)
print(df)
```

```
  Name  Age
0    A   20
1    B   21
2    C   22
```

[10]:
```
# Dataframe attributes
print(df.shape)  # rows, columns
print(df.columns)
print(df.index)
print(df.dtypes)
print(df.ndim)
print(df.size)

# head& tail
print(df.head())
print(df.tail(2))
```

```
(3, 2)
Index(['Name', 'Age'], dtype='object')
RangeIndex(start=0, stop=3, step=1)
Name    object
Age      int64
dtype: object
2
6
  Name  Age
0    A   20
1    B   21
2    C   22
  Name  Age
1    B   21
2    C   22
```

[13]:
```
# Selecting columns
# selecting single column
print(df["Name"])

# selecting multiple columns
print(df[["Name", "Age"]])

# selecting rows(loc & iloc)
# loc -> label based
```

```
print(df.loc[0])
print(df.loc[0, "Name"])

# iloc -> index based
print(df.iloc[1])
print(df.iloc[0,1])
```

```
0    A
1    B
2    C
Name: Name, dtype: object
  Name  Age
0    A   20
1    B   21
2    C   22
Name     A
Age     20
Name: 0, dtype: object
A
Name     B
Age     21
Name: 1, dtype: object
20
```

[ ]:
```
# slicing rows
print(df[0:2])
print(df.iloc[:2])


# adding new column
df["Passed"] = df["Marks"] >= 85
print(df)

# update values
df.loc[0, "Marks"] = 95
df["Age"] = df["Age"] + 1
print(df)

# delete column
df.drop(columns=["Passed"], inplace=True)

# delete row
df.drop(index=2, inplace=True)
print(df)

# filtering rows
print(df[df["Marks"] > 80])
```

[ ]:
```
# sorting & searching
# sorting
print(df.sort_values("Marks"))
print(df.sort_values("Marks", ascending=False))

print(df.sort_index())

# searching
print(df["Marks"].idxmax())
print(df["Marks"].idxmin())

# aggregation functions
print(df["Marks"].sum())
print(df["Marks"].mean())
print(df["Marks"].max())
print(df["Marks"].min())
print(df["Marks"].count())
print(df["Marks"].std())
print(df["Marks"].var())
```

[17]:
```
# describe() --> statistical summary
print(df.describe())
```

```
        Age
count   3.0
mean   21.0
std     1.0
min    20.0
25%    20.5
50%    21.0
75%    21.5
max    22.0
```

[18]:
```
# value counts
print(df["Age"].value_counts())
```

```
Age
20    1
21    1
22    1
Name: count, dtype: int64
```

[19]:
```
# handling missing data
df2 = pd.DataFrame({"A": [1, 2, None], "B": [4, None, 6]})

print(df2.isna())
print(df2.fillna(0))
print(df2.dropna())
```

```
       A      B
0  False  False
1  False   True
2   True  False
     A    B
0  1.0  4.0
1  2.0  0.0
2  0.0  6.0
     A    B
0  1.0  4.0
```

[ ]:
```
# apply function
df["Grade"] = df["Marks"].apply(lambda x: "A" if x >= 90 else "B")
print(df)
```

[21]:
```
# rename columns
df.rename(columns={"Marks": "Score"}, inplace=True)

# rename index
df.rename(index={0: "Row1", 1: "Row2"}, inplace=True)
print(df)

# String operations
df["Name"] = df["Name"].str.lower()
print(df)
```

```
     Name  Age
Row1    A   20
Row2    B   21
2       C   22
     Name  Age
Row1    a   20
Row2    b   21
2       c   22
```

[22]:
```
# Group by (IMP)
data = {"Dept": ["IT", "IT", "HR", "HR"], "Salary": [50000, 60000, 40000,␣
 ↪45000]}

df = pd.DataFrame(data)

print(df.groupby("Dept").mean())
print(df.groupby("Dept").sum())
```

```
       Salary
Dept
HR    42500.0
IT    55000.0
      Salary
```

```
Dept
HR      85000
IT     110000
```

[23]:
```
# Merge/Join
df1 = pd.DataFrame({"id": [1, 2, 3], "Name": ["A", "B", "C"]})

df2 = pd.DataFrame({"id": [1, 2, 4], "Salary": [50000, 60000, 70000]})

print(pd.merge(df1, df2, on="id", how="inner"))
print(pd.merge(df1, df2, on="id", how="left"))
```

```
   id Name  Salary
0   1    A   50000
1   2    B   60000
   id Name   Salary
0   1    A  50000.0
1   2    B  60000.0
2   3    C      NaN
```

[24]:
```
# concatenation
df1 = pd.DataFrame({"A": [1, 2]})
df2 = pd.DataFrame({"A": [3, 4]})

print(pd.concat([df1, df2]))
```

```
   A
0  1
1  2
0  3
1  4
```

[25]:
```
# reading & writing files

# reading csv
# df = pd.read_csv("data.csv")

# writing csv
df.to_csv("output.csv", index=False)
```

[26]:
```
# example
# student dataframe
students = pd.DataFrame({"Name": ["A", "B", "C", "D"], "Marks": [78, 85, 62,␣
 ↪90]})

print(students[students["Marks"] >= 75])
print("Topper:", students.loc[students["Marks"].idxmax()])
```

```
  Name  Marks
```

```
0    A     78
1    B     85
3    D     90
Topper: Name      D
Marks    90
Name: 3, dtype: object
```