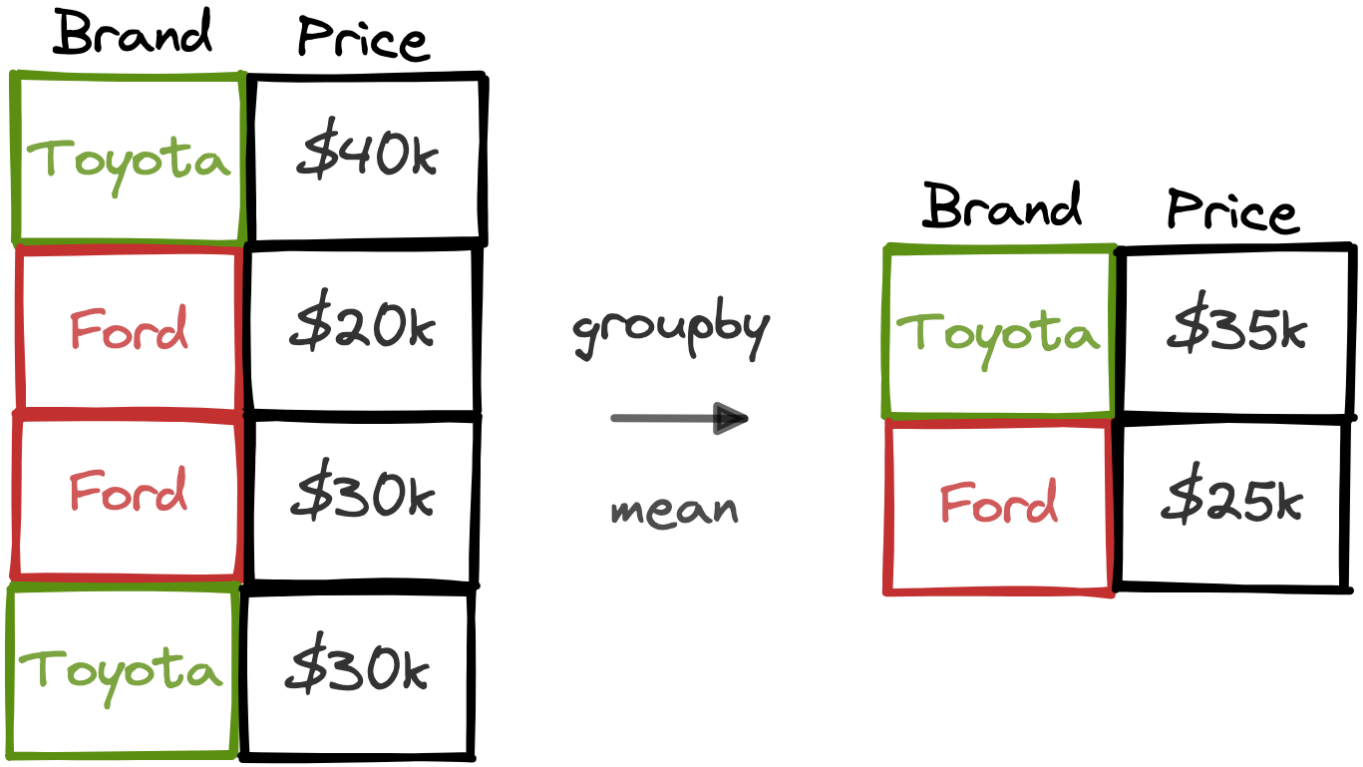
groupby

intro

- **Definition**: GroupBy is a powerful feature in Pandas that allows you to split data into groups, perform calculations on each group, and combine the results.
- **Key Concept**: It follows the **split-apply-combine** strategy:
  - **Split**: Split the data into groups.
  - **Apply**: Apply a function (e.g., mean, sum, min, etc.) to each group.
  - **Combine**: Combine the results into a new object.

intro

- In real-world scenarios, data is often **large** and **structured** in ways that require grouping to analyze trends, patterns, or summaries. GroupBy is essential for breaking data into smaller, meaningful chunks for analysis.



GroupBy Object vs DataFrame

- When you use the groupby() method, it **returns a GroupBy object, not a DataFrame**.
- The GroupBy object acts as an intermediate representation of the grouped data.
- **Code Example**:

```
# Group by 'Category'
grouped = df.groupby('Category')

print(type(grouped))

<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
```

The GroupBy object is not a DataFrame; it's a special object that contains group information.

Basic Aggregations on GroupBy Objects

group calculations like
min, max, mean, sum, first, last, idxmax, and idxmin
can be preformed on grouped data.

```
grouped['Values'].sum()
```

→ The total sum of the values

```
grouped['Values'].last()
```

→ The last row of each group

```
grouped['Values'].idxmax()
```

→ The index of the maximal value

Attributes of the GroupBy Object

- .size()
  Returns the size of each group, similar to value_counts() on a DataFrame.
- .groups
  Returns a dictionary where keys are group names, and values are lists of row indices belonging to each group.
- .indices
  Returns a dictionary where keys are group names, and values are arrays of row indices.

```
x.size()
```

```
g    12
s    11
```

```
x.indices
```

```
{'g': array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11], dtype=int64),
 's': array([12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22], dtype=int64)}
```

get_group()

The get_group() method allows you to retrieve all rows belonging to a specific group.

```
group_a = grouped.get_group('A')
print("Group A:")
print(group_a)
```

```
Group A:
  Category  Values
0        A      10
2        A      15
6        A      40
```

Summary and Best Practices

- GroupBy is not a DataFrame: It’s an intermediate object for grouping data.
- Basic Aggregations: Functions like min, max, mean, sum, etc., can be applied to groups.
- Attributes:
  - .size() → Similar to value_counts().
  - .groups → Dictionary of group names and row indices.
  - .indices → Row indices for each group.
  - get_group(): Retrieve rows for a specific group.