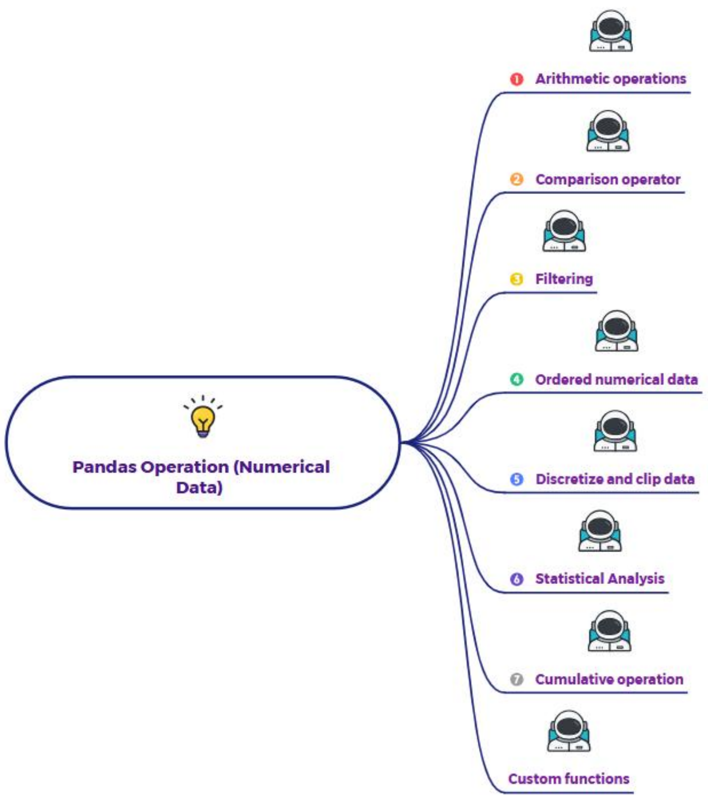
How to handle numerical data in pandas?

Handling numerical data in Pandas refers to the various operations and techniques used to process, analyze, and derive insights from columns containing numbers. These operations are fundamental to almost any data analysis workflow, as numerical data forms the basis for calculations, comparisons, and statistical modeling.



Purpose of Handling Numerical Data

The primary **purpose** of handling numerical data in Pandas is to **extract meaningful information, perform calculations, identify patterns, and prepare data for quantitative analysis or machine learning models.** It allows you to:

- Perform calculations (e.g., sum, average, difference).
- Compare values (e.g., greater than, less than).
- Group and aggregate data based on numerical properties.
- Transform numerical features for better model performance.
- Understand the distribution and characteristics of numerical variables.

How Numerical Data is Handled and Why It Is Required

Pandas provides a rich set of functionalities for numerical data, as illustrated in the image:

1. **Arithmetic Operations:**
   - **What it does:** This involves performing basic mathematical operations like addition, subtraction, multiplication, and division on numerical columns or between numerical columns. You can add a constant to a column, multiply two columns together, etc.
   - **Why it's required:** Essential for creating new derived features (e.g., calculating 'Total Price' from 'Quantity' and 'Unit Price'), scaling data, or adjusting values based on specific business rules.
2. **Comparison Operators:**
   - **What it does:** Allows you to compare numerical values using operators like > (greater than), < (less than), == (equal to), >= (greater than or equal to), <= (less than or equal to), and != (not equal to). These operations typically return boolean (True/False) Series.
   - **Why it's required:** Crucial for filtering data based on numerical conditions (e.g., selecting sales records where 'Revenue' is greater than 1000), validating data, or categorizing data based on thresholds.
3. **Filtering:**
   - **What it does:** Uses boolean Series (often generated by comparison operators) to select a subset of rows from the DataFrame where the condition is True.
   - **Why it's required:** Fundamental for isolating specific data points or observations that meet certain numerical criteria, allowing for focused analysis on relevant subsets (e.g., analyzing only high-value customers or products below a certain stock level).

4. **Ordered Numerical Data:**
   - **What it does:** Refers to operations that deal with the inherent order of numerical values. This includes sorting data (ascending or descending), finding ranks, or working with percentiles.
   - **Why it's required:** Essential for understanding data distribution, identifying top/bottom performers, creating ordered categories, or preparing data for time-series analysis where order matters.
5. **Discretize and Clip Data:**
   - **What it does:**
     - **Discretization (Binning):** Converting continuous numerical data into discrete bins or categories (e.g., grouping ages into '0-18', '19-35', '36+').
     - **Clipping (Winsorization):** Limiting values within a certain range, setting values above an upper bound to the upper bound, and values below a lower bound to the lower bound.
   - **Why it's required:**
     - **Discretization:** Useful for simplifying complex numerical data, handling non-linear relationships, or preparing data for models that prefer categorical inputs.
     - **Clipping:** Important for handling outliers by capping extreme values, which can prevent them from disproportionately influencing analyses or machine learning models.
6. **Statistical Analysis:**
   - **What it does:** Pandas provides built-in methods for calculating various descriptive statistics on numerical columns, such as mean(), median(), std() (standard deviation), var() (variance), min(), max(), sum(), count(), and describe().
   - **Why it's required:** Essential for understanding the central tendency, spread, and overall distribution of numerical data, providing quick summaries and insights into the dataset's characteristics.

7. **Cumulative Operations:**
    - **What it does:** Performs operations that accumulate results over a sequence. Examples include cumsum() (cumulative sum), cumprod() (cumulative product), cummax() (cumulative maximum), and cummin() (cumulative minimum).
    - **Why it's required:** Useful for tracking running totals, progressive maximums/minimums, or analyzing trends over time (e.g., cumulative sales over a fiscal year).
8. **Custom Functions:**
    - **What it does:** Allows users to apply their own defined Python functions to numerical data, either element-wise, row-wise, or column-wise. This is often done using methods like apply().
    - **Why it's required:** Provides ultimate flexibility to perform complex or highly specific calculations that are not covered by built-in Pandas methods, enabling tailored data transformations and analyses.

In summary, handling numerical data in Pandas involves a comprehensive suite of operations that enable everything from basic calculations and filtering to advanced statistical analysis and data transformation, all crucial for extracting meaningful insights and preparing data for further analytical steps.