**pandas as pd:**

* Pandas is a powerful Python library used for data manipulation and analysis.
* It provides data structures and functions designed to make working with structured data easy and intuitive.
* The primary data structures in Pandas are the Series and DataFrame.

**Series:**

* A one-dimensional array-like object containing an array of data and an associated array of labels, called the index.
* consists of index and values
* it is vector

**DataFrame:**

* A two-dimensional labeled data structure with columns of potentially different types.
* It is similar to a spreadsheet or SQL table, or a dictionary of Series objects.
* consists of index and columns
* it is matrix

**Key Features:**

* **Data Alignment:** Automatically aligns data based on label, simplifying data manipulation.
* **Handling Missing Data:** Provides methods for detecting and handling missing data, making data cleaning more manageable.
* **Operations:** Supports various operations like groupby, merge, join, reshape, and pivot tables, making data manipulation and analysis more flexible.
* **Input/Output:** Can read from and write to various data formats like CSV, Excel, SQL databases, JSON, and HDF5.
* **Time Series:** Offers powerful tools for working with time-series data, including date range generation and frequency conversion.

**Functionality:**

* **Data Cleaning:** Provides methods for removing or filling missing data, handling duplicates, and performing transformations.
* **Data Analysis:** Allows for data aggregation, filtering, grouping, and statistical operations.
* **Data Visualization**: Integrates well with libraries like Matplotlib and Seaborn for data visualization and plotting.
* **Time Series Analysis:** Offers specialized methods for time series data manipulation, including resampling, shifting, and rolling window calculations.
* **Integration with Other Libraries:**Pandas integrates well with other Python libraries such as NumPy, Matplotlib, SciPy, and scikit-learn, enabling comprehensive data analysis and machine learning workflows.

Pandas is widely used in various fields, including data science, finance, economics, social sciences, and more. Its intuitive interface and powerful functionality make it a go-to tool for data analysis and manipulation tasks in Python.

**loc and iloc**

* loc stands for location
* loc and iloc are both methods used in Pandas to access and manipulate data within a DataFrame.
* They are essential for selecting rows and columns based on labels (with loc) or integer-based indices (with iloc).
* \* loc => labels , front end

\* iloc => back end , integer index , row index

**head and tails:**

* head shows first 5 rows as default
* tail shows 5 last rows as default

**.apply(---):**

* we can apply any function on df with .apply()
* we have to pass variable in the brackets

**axis:**

* x-axis is represented as 0 in pandas
* y -axis is represented by 1 in pandas
* we can use it for movements

**SIMD:**

**Single Instruction, Multiple Data.**

* It's a parallel computing technique used to perform the same operation on multiple data points simultaneously.
* SIMD architectures are common in modern CPUs and GPUs and are used to accelerate data processing tasks by exploiting data-level parallelism.
* In SIMD, a single instruction is applied to multiple data elements in a single processor cycle.
* This contrasts with traditional scalar processors, where each instruction operates on only one data element at a time.
* SIMD instructions typically operate on vectors or arrays of data elements.
* For example, if you have two arrays A and B of the same length and you want to add corresponding elements together to create a third array C, a SIMD instruction could perform this operation much faster than a scalar instruction by adding multiple pairs of elements simultaneously.

Some common SIMD instruction sets include:

**SSE (Streaming SIMD Extensions):**

* Introduced by Intel, SSE provides SIMD instructions for floating-point and integer operations on 128-bit vectors.
* SSE has been updated several times with SSE2, SSE3, SSE4, and SSE4.2 offering additional functionality.

**AVX (Advanced Vector Extensions):**

* Also introduced by Intel, AVX extends the SIMD capabilities of SSE by providing wider registers (256 bits and 512 bits), additional instructions, and improved performance.

**NEON:**

* NEON is an SIMD architecture extension for ARM processors commonly found in mobile devices and embedded systems.
* It provides SIMD instructions for 64-bit and 128-bit vectors.

**SIMD extensions for GPUs:**

* Modern GPUs (Graphics Processing Units) also utilize SIMD architectures to perform parallel processing on large datasets, which is essential for graphics rendering, scientific simulations, and machine learning algorithms.
* SIMD is widely used in various fields such as multimedia processing, scientific computing, signal processing, and cryptography.
* Programmers can utilize SIMD instructions through compiler intrinsics or specialized libraries to optimize performance-critical code sections for tasks that exhibit data-level parallelism.

**Functions in pandas:**

Pandas offers a wide range of functions for data manipulation, analysis, and exploration. Here are some of the key functions and capabilities of Pandas:

1. **Data Structures**:
   * **DataFrame**: A two-dimensional labeled data structure with columns of potentially different types. You can create a DataFrame using various methods, load data from different file formats, or convert other data structures like lists or dictionaries into DataFrames.
   * **Series**: A one-dimensional labeled array capable of holding any data type.
2. **Data Input/Output**:
   * **read\_csv()**, **read\_excel()**, **read\_json()**: Functions to read data from CSV files, Excel files, JSON files, etc.
   * **to\_csv()**, **to\_excel()**, **to\_json()**: Functions to write data to CSV files, Excel files, JSON files, etc.
3. **Data Exploration and Manipulation**:
   * **head()**, **tail()**: View the first or last few rows of a DataFrame.
   * **info()**: Get a concise summary of the DataFrame, including the data types and number of non-null values.
   * **describe()**: Generate descriptive statistics for numerical columns in the DataFrame.
   * **shape**: Get the dimensions of the DataFrame (number of rows and columns).

**drop()**: Remove rows or columns from the DataFrame.

* + **fillna()**, **dropna()**: Handling missing values by filling them with a specified value or dropping rows/columns with missing values.
  + **groupby()**: Group data in a DataFrame based on one or more columns.
  + **merge()**, **concat()**, **join()**: Combining multiple DataFrames based on indexes or columns.

1. **Data Selection and Filtering**:
   * Conditional selection: Use boolean indexing to filter rows or columns based on conditions.
   * **isin()**: Filter data based on whether it is contained in a list-like object.
   * **query()**: Select data using an expression.
2. **Data selection and indexing:**
   * **loc[]:** Access a group of rows and columns by label(s).
   * **iloc[]:** Access a group of rows and columns by integer position(s).
   * **at[]:** Access a single value for a row/column label pair.
   * **iat[]:** Access a single value for a row/column pair by integer position.
3. **Data Aggregation and Transformation**:
   * **groupby()**: Group data and perform aggregation operations like sum, mean, count, etc.
   * **pivot\_table()**: Create a spreadsheet-style pivot table as a DataFrame.
   * **apply()**: Apply a function along an axis of the DataFrame.
   * **map()**, **applymap()**: Apply functions element-wise to Series or DataFrames.
4. **Data Visualization**:
   * Integration with Matplotlib and Seaborn for data visualization directly from Pandas DataFrames.
   * **plot()**: Generate various types of plots (line, bar, histogram, scatter, etc.) from DataFrame dat