Libraries

Libraries:

Libraries are the collection of pre written code including modules, functions, classes designed to help developers perform specific tasks without having to write code from scratch.

Data Analysis & Manipulation

- **Pandas :** for data manipulation and analysis (DataFrames, CSVs, Excel, etc.).
- **Numpy :** (numerical python) is used for multidimensional Arrays and to perform mathematical operations.

Data Visualization :

- **Matplotlib :** basic plotting (line, bar, scatter, etc.)
- **Seaborn :** statistical and beautiful visualizations (built on Matplotlib)
- **Plotly :** interactive visualizations
- **Bokeh :** interactive, web-based plots
- **Altair :** declarative and easy visualization library

Machine Learning & AI :

- **Scikit-learn** – traditional ML algorithms (SVM, regression, clustering)
- **TensorFlow** – deep learning framework from Google
- **PyTorch** – deep learning framework from Meta
- **XGBoost / LightGBM / CatBoost** – boosting algorithms for ML models
- **Keras** – high-level API for TensorFlow

Data Science & Statistics :

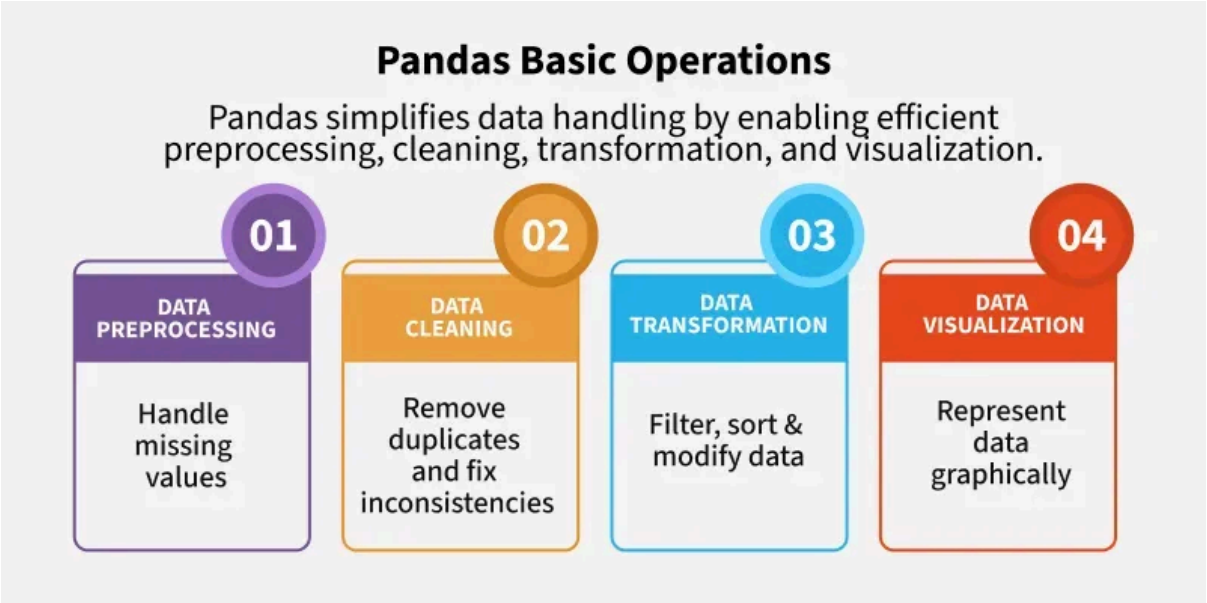
- **Statsmodels –** for statistical models and tests
- **SciPy –** for scientific and mathematical operations
- **SymPy –** symbolic mathematics (algebra, calculus, equations)

NLP (Natural Language Processing):

- **NLTK** → basic NLP tasks like tokenization and stemming
- **spaCy** → advanced NLP for named entity recognition and parsing
- **Transformers (Hugging Face)** → for LLMs like BERT, GPT, etc.

Pandas

- Is a powerful open source python library used for data manipulation, cleaning and analysis.
- Pandas stands for **python data analysis.** Built on top of NumPy.
- Pandas can only process the tabular data. It will not support the image, audio, video data.



Main data structures:

1. **1-D labeled array (**like a column in excel).
2. **Data Frame - 2D (**like a excel sheet)

Reading and Writing of the data:

To read file

- **pd.read_csv(** path **)**
- **pd.read_excel(** path **)**
- **pd.read_json(** path **)**

To write file

- **df.to_csv(** "file", index = False**)**
- **df.to_excel(** "file", index = False**)**

Exploring the Data:

- df.head() -> first 5 rows , we can increase the rows by passing parameter
- df.tails() -> last 5 rows, we can increase the rows by passing parameter.
- df.shape -> (rows, columns)
- df.info() -> column types
- df.describe -> summary statistics
- df.columns -> list of column names

Selecting Data:

- df['Age'] # select one column
- df[['Name', 'Score']] # select multiple columns
- df.iloc[0] # select first row (by index)
- df.loc[0, 'Name'] # select specific cell
- df[df['Score'] > 85] # filter rows with condition

Modifying Data:

- df['Grade'] = ['A', 'A', 'B'] # add new column
- df.drop('Age', axis=1, inplace=True) # remove column
- df.rename(columns={'Score': 'Marks'}, inplace=True)

Aggregating & Grouping:

- df['Score'].mean() # average score
- df.groupby('Grade')['Score'].mean() # avg score by grade
- df.sort_values('Score', ascending=False) # sort by column

Handling Missing Data:

- df.isnull().sum() # check missing values
- df.fillna(0, inplace=True) # replace missing with 0
- df.dropna(inplace=True) # remove missing rows

Exploring the Data:

- pd.merge(df1, df2, on='ID') # merge by column
- pd.concat([df1, df2]) # stack vertically

Useful operations:

- df['Score'].max() # max value
- df['Score'].min() # min value
- df['Score'].value_counts() # frequency of values
- df.sort_index() # sort by index