Mindmap to Learn Pandas

* **Overview**: Pandas is a powerful open-source data analysis and manipulation library for Python, providing data structures and functions needed to work with structured data seamlessly.
* **Installation**: Install Pandas using pip: pip install pandas.
* **Data Structures**
  + **Series**: One-dimensional labeled array capable of holding any data type.
    - Example: import pandas as pd; s = pd.Series([1, 2, 3], index=['a', 'b', 'c']).
  + **DataFrame**: Two-dimensional labeled data structure with columns of potentially different types.
    - Example: df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4]}).
* **Data Input and Output**
  + **CSV**: Read from and write to CSV files.
    - Read: df = pd.read\_csv('file.csv').
    - Write: df.to\_csv('file.csv').
  + **Excel**: Read from and write to Excel files.
    - Read: df = pd.read\_excel('file.xlsx').
    - Write: df.to\_excel('file.xlsx').
  + **SQL**: Read from and write to SQL databases.
    - Read: df = pd.read\_sql('SELECT \* FROM table', con=connection).
    - Write: df.to\_sql('table', con=connection, index=False).
* **Data Inspection**
  + **Head and Tail**: View the first and last rows of a DataFrame.
    - Example: df.head(), df.tail().
  + **Info**: Get a concise summary of a DataFrame.
    - Example: df.info().
  + **Describe**: Generate descriptive statistics.
    - Example: df.describe().
* **Data Selection and Filtering**
  + **Indexing and Selecting Data**: Access specific rows and columns.
    - By label: df.loc[row\_label, column\_label].
    - By position: df.iloc[row\_position, column\_position].
  + **Boolean Indexing**: Filter data based on conditions.
    - Example: df[df['column'] > 0].
* **Data Manipulation**
  + **Adding and Dropping Columns**: Modify the structure of DataFrames.
    - Add: df['new\_column'] = value.
    - Drop: df.drop(columns=['column\_name'], inplace=True).
  + **Renaming Columns**: Rename DataFrame columns.
    - Example: df.rename(columns={'old\_name': 'new\_name'}, inplace=True).
  + **Handling Missing Data**: Detect and handle missing values.
    - Detect: df.isnull().
    - Drop: df.dropna(inplace=True).
    - Fill: df.fillna(value, inplace=True).
* **Data Transformation**
  + **Apply Functions**: Use apply to transform data.
    - Example: df['column'] = df['column'].apply(lambda x: x + 1).
  + **Group By**: Perform split-apply-combine operations.
    - Example: df.groupby('column').mean().
  + **Pivot Tables**: Create pivot tables for data summarization.
    - Example: df.pivot\_table(values='value', index='index', columns='columns').
* **Merging and Joining**
  + **Concatenation**: Concatenate DataFrames along a particular axis.
    - Example: pd.concat([df1, df2], axis=0).
  + **Merging**: Merge DataFrames based on keys.
    - Example: pd.merge(df1, df2, on='key').
  + **Joining**: Join DataFrames on indexes.
    - Example: df1.join(df2, how='inner').
* **Time Series Analysis**
  + **Datetime Conversion**: Convert data to datetime format.
    - Example: df['date'] = pd.to\_datetime(df['date']).
  + **Resampling**: Resample time series data.
    - Example: df.resample('M').mean().
  + **Rolling Windows**: Apply rolling window calculations.
    - Example: df['rolling\_mean'] = df['column'].rolling(window=3).mean().
* **Visualization**
  + **Plotting**: Use Pandas' built-in plotting capabilities.
    - Example: df.plot(kind='line').
  + **Integration with Matplotlib**: Customize plots with Matplotlib.
    - Example: import matplotlib.pyplot as plt; df.plot(); plt.show().
* **Performance Optimization**
  + **Efficient Data Loading**: Optimize data loading with chunksize.
    - Example: pd.read\_csv('file.csv', chunksize=1000).
  + **Vectorized Operations**: Use vectorized operations for performance.
    - Example: df['new\_column'] = df['column'] \* 2.
  + **Memory Usage**: Optimize memory usage by downcasting data types.
    - Example: df['column'] = pd.to\_numeric(df['column'], downcast='float').
* **Advanced Topics**
  + **MultiIndex**: Use hierarchical indexing for advanced data analysis.
    - Example: df.set\_index(['index1', 'index2']).
  + **Categorical Data**: Optimize performance with categorical data types.
    - Example: df['column'] = df['column'].astype('category').
  + **Custom Data Types**: Define and use custom data types with dtype.
    - Example: df['column'] = df['column'].astype('int32').
* **Best Practices**
  + **Code Readability**: Write clean and readable code.
    - Example: Use meaningful variable names and comments.
  + **Documentation**: Document your data analysis steps.
    - Example: Use Jupyter notebooks to document and visualize analysis.
  + **Version Control**: Use version control for code and data.
    - Example: Use Git for code versioning and DVC for data versioning.