# Understanding Pandas Series vs DataFrame

## When You Get a Series

When You Get a Series:

1. Selecting a Single Column: df['column_name'] - Accessing a single column returns a Series.

2. Row Selection Using .loc or .iloc with a Single Label/Index: df.loc['row_label'] or df.iloc[0] - Selecting a single row returns a Series.

3. Aggregation on a Single Column: df['column_name'].sum() - Aggregating a single column returns a Series.

4. Dot Notation for Column Access: df.column_name - Accessing a column with dot notation returns a Series.

5. Applying a Unary Operation on a DataFrame: (df > 0).any() - Unary operations return a Series.

6. Slicing a Single Row from DataFrame: df.iloc[1] - Slicing a single row returns a Series.

7. Single Condition Filtering that Reduces to a Single Column: df[df['column_name'] > 0]['column_name'] - Filtering with a condition for a single column returns a Series.

## When You Get a DataFrame

**Pandas Series vs DataFrame**

When You Get a DataFrame:

1. Selecting Multiple Columns: df[['column_name1', 'column_name2']] - Selecting multiple columns returns a DataFrame.

2. Conditional Selection on Rows: df[df['column_name'] > 0] - Conditional row filtering returns a DataFrame.

3. Slicing Multiple Rows: df[1:4] - Slicing rows returns a DataFrame.

4. Using .loc or .iloc for Slicing or Multiple Indices/Labels: df.loc['row_label1':'row_label3'] or df.iloc[0:3] - Selecting multiple elements returns a DataFrame.

5. Applying Aggregations with groupby: df.groupby('column_name').mean() - Groupby aggregations return a DataFrame.

6. Transposing (T): df.T - Transposing the DataFrame returns a DataFrame.

7. Adding a Dimension with to_frame(): df['column_name'].to_frame() - Converting a Series to a DataFrame.

8. Using .assign() to Add Columns: df.assign(new_col=df['column_name']*2) - Adding a column with .assign() returns a DataFrame.

9. Filtering with Multiple Conditions: df[(df['column1'] > 0) & (df['column2'] < 0)] - Multiple conditions filtering returns a DataFrame.

10. Using .drop() to Remove Columns or Rows: df.drop(columns=['column_name']) - Dropping columns or rows returns a DataFrame.