# Pandas in Python - Step-by-Step Tutorial

## Introduction to Pandas

Pandas is a Python library used for data manipulation and analysis. It provides data structures like Series and DataFrame for efficiently handling large datasets.

## Step 1: Install Pandas

To install Pandas, use the following command:

pip install pandas

## Step 2: Import Pandas

To start using Pandas, import it in your Python script:

import pandas as pd

## Step 3: Creating a DataFrame

A DataFrame is a 2D table-like structure similar to an Excel spreadsheet or SQL table.

### Example: Creating a DataFrame

import pandas as pd  
  
data = {  
 'Name': ['Alice', 'Bob', 'Charlie', 'David'],  
 'Age': [25, 30, 35, 40],  
 'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']  
}  
  
df = pd.DataFrame(data)  
print(df)

## Step 4: Displaying Data

To display the first few rows of the DataFrame, use:

df.head()

## Step 5: Reading and Writing Data

Pandas can read and write data from multiple file formats such as CSV, Excel, and JSON.

### Example: Reading a CSV File

df = pd.read\_csv('data.csv')

### Example: Writing to a CSV File

df.to\_csv('output.csv', index=False)

## Step 6: Selecting Data from a DataFrame

### Example: Selecting a Single Column

df['Name']

### Example: Selecting Multiple Columns

df[['Name', 'City']]

## Step 7: Filtering Data

To filter data based on a condition:

### Example: Filter Rows Where Age > 30

df[df['Age'] > 30]

## Step 8: Sorting Data

Sort data using the `sort\_values()` method.

### Example: Sorting by Age

df.sort\_values(by='Age', ascending=False)

## Step 9: Grouping Data

Use the `groupby()` method to group data and apply aggregation functions.

### Example: Grouping by City and Counting Entries

df.groupby('City').count()

## Step 10: Handling Missing Data

Pandas provides methods to handle missing data, such as `fillna()` and `dropna()`.

### Example: Filling Missing Values with Default

df.fillna('Unknown')