Python

Pandas

Alish Bista - 11/11/2025

Pandas

Pandas

Pandas To The Rescue

- Created by Wes McKinney in 2008 while working at a hedge fund
- Name comes from “Panel Data” - an econometrics term
- Now the most popular data analysis tool in Python

Series

Binging into series

- A single column of data with labels
- Like one column in an Excel spreadsheet
- Has an index (row labels) and values (the actual data)
- All values are typically the same type (all numbers, all text, etc.)
- Example: A list of movie titles, a column of temperatures, prices

```
import pandas as pd


# Create a simple Series
fruits = pd.Series(['Apple',
'Banana', 'Cherry', 'Date'])
print(fruits)


# Create a Series with custom index
prices = pd.Series([1.99, 0.59,
3.49, 2.99], index=['Apple',
'Banana', 'Cherry', 'Date'])
print(prices)
```

DataFrame

Data In A Frame

- A table of data with rows and columns
- Like a complete Excel spreadsheet
- Multiple columns (each column is a Series)
- Each column can have different data types
- Has row labels (index) and column labels (names)

```
import pandas as pd

# Create from a dictionary
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['NYC', 'LA', 'Chicago']
}

df = pd.DataFrame(data)
print(df)
```

Loading Data

Loading Data

Loading or Creating DF

- ***DataFrame from scratch:***
  - Use dictionaries to define your data
  - Pandas turns it into a neat table
- ***Loading existing data:***
  - Read from CSV files: pd.read_csv('filename.csv')
  - Read from Excel: pd.read_excel('filename.xlsx')
  - Other formats: JSON, SQL databases, and more

First Look

Exploring the Data

- *See the first few rows:*
  - df.head() - Shows first 5 rows (or specify a number)
  - df.tail() - Shows last 5 rows
- *Get basic information:*
  - df.shape - How many rows and columns? (rows, columns)
  - df.info() - Column names, data types, missing values
  - df.columns - Just the column names

Column Selection

Selecting One or More Columns

- Selecting one column:
  - df['column_name'] - Returns a Series
  - Like highlighting one column in Excel
- Selecting multiple columns:
  - df[['col1', 'col2', 'col3']] - Returns a DataFrame
  - Need to use the double brackets!

Filtering

Filtering Rows Where ....

- **Basic filtering:**
  - df[df['age'] > 25] - All rows where age is greater than 25
  - df[df['city'] == 'NYC'] - All rows where city is NYC
- **Common operators:**
  - >, <, >=, <= for numbers
  - == for exact matches (note: double equals!)
  - != for “not equal to”

Sorting & Multiple Conditions

Sorting and Combining Conditions

- **Sorting data:**
  - df.sort_values('column_name') - Sort by one column
  - Add ascending=False for descending order
  - df.sort_values(['col1', 'col2']) - Sort by multiple columns
- **Combining conditions:**
  - AND: df[(df['age'] > 25) & (df['city'] == 'NYC')]
  - OR: df[(df['age'] > 25) | (df['age'] < 18)]

Visualization

Visualization

Viz Basics

- Basic plot types:
  - df['column'].plot(kind='bar') - Bar chart
  - df['column'].plot(kind='line') - Line chart
  - df['column'].plot(kind='pie') - Pie chart
  - df['column'].plot(kind='hist') - Histogram
- When to use each:
  - Bar: Compare categories
  - Line: Show trends over time
  - Pie: Show proportions
  - Histogram: Show distribution of values

Q&A