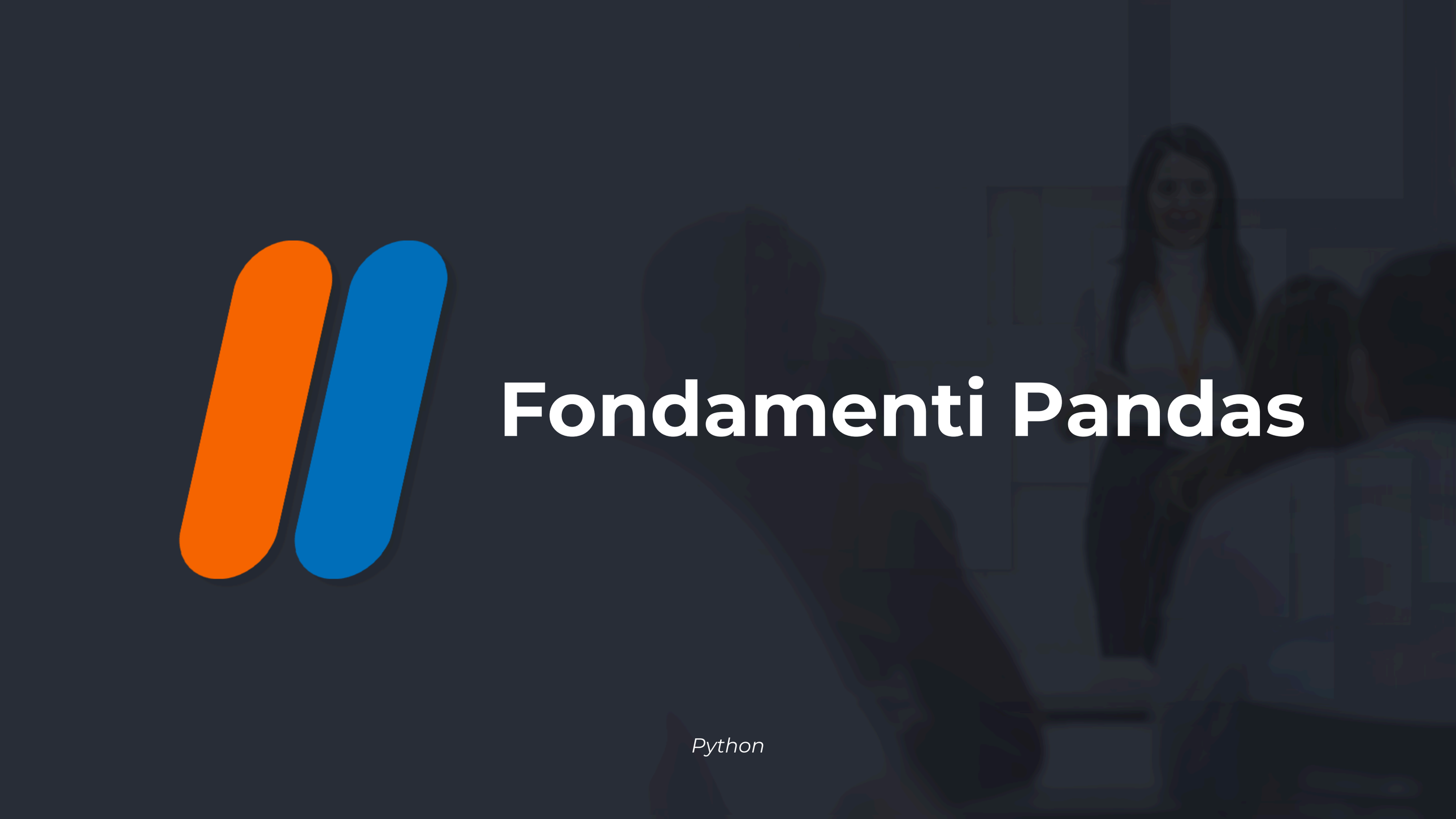

Fondamenti Pandas

Python

Introduzione a Pandas

Pandas è costruito sopra NumPy, un'altra libreria Python che fornisce supporto per array multidimensionali, e integra bene con altre librerie come Matplotlib per la visualizzazione dei dati e scikit-learn per il m.l.

Pandas introduce principalmente due nuove strutture di dati:

Series e DataFrame.

- Series: Una Serie è un array unidimensionale etichettato capace di contenere dati di qualsiasi tipo (interi, stringhe, numeri a virgola mobile, oggetti Python, etc.). L'asse delle etichette di una Serie viene chiamato indice.
- DataFrame: Un DataFrame è una struttura dati bidimensionale, con dati allineati in un formato tabellare. È simile a un foglio di calcolo o a una tabella SQL. Un DataFrame supporta colonne di diversi tipi e può essere visto come un dizionario di Serie, dove ogni colonna è una Serie.

Esempio di utilizzo:

```
import pandas as pd

# Creare una Series da una lista
s = pd.Series([10, 20, 30, 40, 50])
print(s)

# Creare una Series con un indice personalizzato
s_custom = pd.Series([100, 200, 300], index=['a', 'b', 'c'])
print(s_custom)
```

```
import pandas as pd

# Creare un DataFrame da dizionario di liste
data = {
    'Nome': ['Alice', 'Bob', 'Charlie'],
    'Età': [25, 30, 35],
    'Città': ['Roma', 'Milano', 'Napoli']
}

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

# Creare un DataFrame da dei dizionari
data_list = [
    {'Nome': 'Daniele', 'Età': 28, 'Città': 'Torino'},
    {'Nome': 'Elisa', 'Età': 22, 'Città': 'Bologna'}
]

df2 = pd.DataFrame(data_list)
print(df2)
```

Funzioni Principali di Pandas

Gestione dei dati: Pandas fornisce funzionalità avanzate per il caricamento, la pulizia, l'esplorazione e la manipolazione dei dati.

Indicizzazione e selezione: La libreria offre strumenti potenti per l'indicizzazione e la selezione dei dati, permettendo di accedere a sottinsiemi di dati in modo efficiente.

Esempio di utilizzo:

Accedere ai dati in un DataFrame:

```
# Accedere a una colonna specifica
print(df['Nome'])

# Accedere a una riga specifica tramite l'indice
print(df.loc[0])  # Prima riga
```

Aggiungere una nuova colonna a un DataFrame:

```
df['Salario'] = [50000, 60000, 70000]
print(df)
```

Operazioni di base su Series e DataFrame:

```
# Operazioni su Series
s_somma = s + 10
print(s_somma)

# Operazioni su DataFrame (somma, media, ecc.)
df['Età Media'] = df['Età'].mean()
print(df)
```

Pandas offre una vasta gamma di metodi che lo rendono uno strumento potente per l'analisi dei dati. Ecco un'analisi di alcuni dei metodi più utilizzati in Pandas, focalizzandosi su come vengono applicati nelle operazioni comuni di manipolazione dei dati.

Metodi di Lettura e Scrittura

- pd.read_csv(): Carica dati da un file CSV in un DataFrame. È uno dei metodi di input più comuni, grazie alla popolarità del formato CSV per lo scambio di dati.

- pd.to_csv(): Esporta un DataFrame in un file CSV, permettendo di salvare i risultati delle analisi in un formato facilmente condivisibile e leggibile.

Esplorazione e Pulizia dei Dati

- df.head(): Visualizza le prime N righe di un DataFrame (5 per default), utile per un primo esame dei dati.
- df.describe(): Fornisce un riepilogo statistico delle colonne numeriche, utile per avere una visione generale delle distribuzioni e identificare eventuali valori anomali.
- df.dropna(): Rimuove le righe con valori mancanti, essenziale nella fase di pulizia dei dati.
- df.fillna(): Sostituisce i valori mancanti con un valore specificato, permettendo di mantenere dati completi anche in presenza di lacune.

Manipolazione dei Dati

- df.loc[]: Seleziona dati in base alle etichette delle righe e delle colonne. È molto flessibile, permettendo selezioni condizionali e accesso a sottogruppi di dati.

- df.iloc[]: Seleziona dati in base alla posizione numerica delle righe e delle colonne. È utile quando l'ordine dei dati è importante o quando si lavora con indici numerici.

- df.groupby(): Raggruppa i dati in base a una o più colonne e consente di applicare funzioni di aggregazione (come somma, media, etc.) sui gruppi, essenziale per l'analisi aggregata dei dati.

- df.merge(): Combina due DataFrame in base a una o più chiavi comuni, simile all'operazione di JOIN in SQL. È cruciale per unire dati da fonti diverse.

Per dimostrare le capacità avanzate di Pandas, creiamo un esempio che attraversa diverse fasi dell'analisi dei dati: il caricamento, la pulizia, la manipolazione e l'aggregazione, fino alla visualizzazione.

Immagina di avere un dataset di vendite di un negozio, contenente informazioni su prodotti, quantità vendute, prezzi e date di vendita.

Questo esempio assumerà che hai un file CSV vendite.csv con le seguenti colonne: Data, Prodotto, Quantità, PrezzoUnitario.

```
import pandas as pd
import matplotlib.pyplot as plt

# Caricamento dei dati
df = pd.read_csv('vendite.csv', parse_dates=['Data'])

# Pulizia dei dati
# Rimuoviamo eventuali righe con valori mancanti
df.dropna(inplace=True)

# Assicuriamoci che Quantità e PrezzoUnitario siano positivi
df = df[(df['Quantità'] > 0) & (df['PrezzoUnitario'] > 0)]

# Aggiungiamo una colonna per il totale vendite
df['TotaleVendite'] = df['Quantità'] * df['PrezzoUnitario']

# Manipolazione dei dati
# Aggiungiamo colonne per anno e mese per analisi temporale
df['Anno'] = df['Data'].dt.year
df['Mese'] = df['Data'].dt.month

# Aggregazione dei dati
# Calcoliamo le vendite totali per prodotto
vendite_per_prodotto = df.groupby('Prodotto')['TotaleVendite'].sum().sort_values(ascending=False)

# Calcoliamo le vendite mensili totali
vendite_mensili = df.groupby(['Anno', 'Mese'])['TotaleVendite'].sum()

# Visualizzazione
# Grafico delle vendite totali per prodotto
vendite_per_prodotto.head(10).plot(kind='bar', title='Top 10 Prodotti per Vendite Totali')
plt.xlabel('Prodotto')
plt.ylabel('Totale Vendite')
plt.show()

# Grafico delle vendite mensili
vendite_mensili.plot(kind='line', title='Vendite Mensili')
plt.xlabel('Mese')
plt.ylabel('Totale Vendite')
plt.show()
```

Spiegazione del Codice

1. Caricamento dei dati: Carichiamo il dataset da un file CSV, specificando che la colonna Data deve essere interpretata come una data.
2. Pulizia dei dati: Rimuoviamo le righe con valori mancanti e filtriamo i dati per assicurarci che le quantità e i prezzi siano positivi, rimuovendo eventuali dati errati o non validi.
3. Manipolazione dei dati: Aggiungiamo una colonna TotaleVendite per rappresentare il totale delle vendite per ogni transazione. Introduciamo anche colonne per l'Anno e il Mese estratti dalla data di vendita, per facilitare analisi temporali.
4. Aggregazione dei dati: Raggruppiamo i dati per prodotto e per coppia anno-mese, calcolando il totale delle vendite. Questo ci permette di analizzare quali prodotti generano maggiori entrate e come le vendite variano nel tempo.
5. Visualizzazione: Creiamo due grafici: uno a barre per le vendite totali dei top 10 prodotti e uno lineare per le vendite mensili nel tempo. Questi grafici offrono una visione immediata delle performance di vendita.

cenni su Tecniche Avanzate con Pandas

Analisi delle Serie Temporali

Pandas offre strumenti potenti per lavorare con serie temporali, rendendolo ideale per analizzare dati temporali complessi. Le funzionalità includono:

- Resampling e Windowing: Permette di aggregare o trasformare dati temporali su intervalli regolari o applicare funzioni su finestre di tempo, come medie mobili o esponenziali.
- Time Zone Handling: Gestisce conversioni tra fusi orari e operazioni consapevoli del fuso orario.
- Periodi e Frequenze: Supporta periodi di tempo specifici e la loro conversione, facilitando l'analisi su basi temporali fisse come mensile o annuale.

cenni su Tecniche Avanzate con Pandas

Operazioni Avanzate di Manipolazione dei Dati

Oltre alle funzionalità di base, Pandas include metodi avanzati per manipolazioni complesse dei dati, come:

- Categorical Data: Supporta dati categorici, che possono aumentare le performance e ridurre l'uso di memoria, specialmente su grandi dataset.
- Query Method: Permette di interrogare i DataFrame con espressioni booleane, semplificando la selezione di sottogruppi di dati.
- MultiIndex: Gestisce DataFrame con più livelli di indici (indice gerarchico), utile per rappresentare dati ad alta dimensionalità in forma tabulare.

In conclusione, la padronanza di Pandas apre la porta a un'ampia gamma di analisi dati e applicazioni, dalle operazioni più semplici a tecniche di analisi avanzate.

La sua integrazione con altre librerie di Python per la visualizzazione e l'analisi statistica ne fa un pilastro fondamentale nell'ecosistema dei dati di Python.


Buon MasterD a tutti