SVILUPPO APPLICAZIONI E ANALISI DATI CON PYTHON

Pandas

Strutture Dati e Operazioni Base

Docente: Marino Speciale

Indice della Lezione

Cos'è Pandas?

Pandas è una libreria Python per l'**analisi e manipolazione dati**. Fornisce strutture dati efficienti e flessibili.

```
import pandas as pd

# Series (1D) - colonna singola
s = pd.Series([10, 20, 30])

# DataFrame (2D) - tabella completa
df = pd.DataFrame({'Nome': ['Alice', 'Bob'],
                   'Età': [25, 30]})
```

Pandas è la base per l'analisi dati e il machine learning in Python

Series: Array etichettato

```
import pandas as pd

# Serie semplice (indice numerico di default)
s1 = pd.Series([10, 20, 30, 40])

# Serie con indice personalizzato
s2 = pd.Series([100, 200, 300],
               index=['a', 'b', 'c'])

# Accesso ai valori
print(s2['a'])      # 100
print(s2[0])        # 100 (accesso numerico)
print(s2.values)    # [100 200 300]
print(s2.index)     # ['a', 'b', 'c']
```

DataFrame: Tabella 2D

```
# Creazione da dizionario (metodo più comune)
df = pd.DataFrame({
    'Nome': ['Mario', 'Luigi', 'Peach'],
    'Età': [25, 30, 28],
    'Città': ['Roma', 'Milano', 'Napoli']
})

# Visualizzazione
print(df.head())    # Prime 5 righe
print(df.info())    # Info struttura (colonne, tipi)
print(df.shape)     # (3, 3) righe e colonne

# Accesso colonne
print(df['Nome'])   # Intera colonna
print(df.Nome)      # Stesso risultato
```

Accesso ai Dati: loc vs iloc

```
# .loc[] - accesso per ETICHETTA (label-based)
print(df.loc[0, 'Nome'])      # Mario (riga 0, col Nome)
print(df.loc[0:1, ['Nome']])  # Prime 2 righe

# .iloc[] - accesso per POSIZIONE (position-based)
print(df.iloc[0, 0])          # Mario (riga 0, col 0)
print(df.iloc[0:2, [0]])      # Prime 2 righe, col 0

# Filtri booleani
print(df[df['Età'] > 26])     # Solo chi ha più di 26 anni
```

Operazioni Base su DataFrame

```
# Aggiungere colonna
df['Stipendio'] = [2000, 2500, 2200]

# Modificare valori
df.loc[0, 'Città'] = 'Firenze'

# Rimuovere colonna
df = df.drop('Stipendio', axis=1)

# Ordinare
df_sorted = df.sort_values(by='Età', ascending=False)

# Statistiche
print(df.describe())     # media, min, max, quartili
print(df['Età'].mean()) # Media della colonna
```

Lettura File: CSV

```
import pandas as pd

# Lettura CSV
df = pd.read_csv('dati.csv')

# Con opzioni personalizzate
df = pd.read_csv('dati.csv',
                 sep=';',           # Separatore
                 header=0,          # Riga intestazione
                 encoding='utf-8')  # Encoding

# Scrittura CSV
df.to_csv('output.csv', index=False)

# Visualizzazione veloce
print(df.head())
print(df.tail())
```

Valori mancanti (NaN)

```
import pandas as pd
import numpy as np

# DataFrame con valori mancanti
df = pd.DataFrame({
    'A': [1, 2, np.nan, 4],
    'B': [5, np.nan, 7, 8]
})

# Identificare NaN
print(df.isna())        # Booleano per ogni cella
print(df.isnull())      # Sinonimo di isna()

# Rimuovere righe con NaN
df_clean = df.dropna()

# Riempire con valore default
df_filled = df.fillna(0)  # Sostituisce NaN con 0
```

Groupby: Raggruppare dati

```
# DataFrame vendite
vendite = pd.DataFrame({
    'Prodotto': ['A', 'B', 'A', 'B', 'A'],
    'Quantità': [10, 15, 20, 5, 30],
    'Prezzo': [100, 200, 100, 200, 100]
})

# Raggruppare per Prodotto e sommare Quantità
totali = vendite.groupby('Prodotto')['Quantità'].sum()
# Output: A: 60, B: 20

# Molteplici aggregazioni
risultato = vendite.groupby('Prodotto').agg({
    'Quantità': 'sum',      # somma
    'Prezzo': 'mean'        # media
})
```

Statistiche descrittive

```
# DataFrame voti
voti = pd.DataFrame({
    'Matematica': [7, 8, 6, 9],
    'Italiano': [6, 7, 8, 9],
    'Inglese': [8, 9, 7, 8]
})

# Statistiche complete
print(voti.describe())

# Funzioni individuali
print(voti.mean())        # Media per colonna
print(voti.median())      # Mediana
print(voti.std())         # Deviazione standard
print(voti.corr())        # Correlazione tra colonne
```

Unire DataFrame: Merge

```
# Due DataFrame da combinare
studenti = pd.DataFrame({
    'ID': [1, 2, 3],
    'Nome': ['Alice', 'Bob', 'Charlie']
})

voti = pd.DataFrame({
    'ID': [1, 2, 3],
    'Matematica': [8, 7, 9]
})

# Merge su colonna comune (ID)
risultato = pd.merge(studenti, voti, on='ID')
# Risultato: ID, Nome, Matematica

# Inner join (default), left, right, outer
```

Lettura/Scrittura JSON

```
import pandas as pd

# Leggere JSON
df = pd.read_json('dati.json')

# Leggere da stringa JSON
json_str = '[{"nome":"Mario", "età":25}]'
df = pd.read_json(json_str)

# Scrivere su JSON
df.to_json('output.json', orient='records')

# Orientamenti disponibili
# 'records': lista di dizionari
# 'split': {index, columns, data}
# 'index': {indice: riga}
```

Esempio Pratico: Analisi vendite (Part 1)

```
import pandas as pd

# Caricamento dati
vendite = pd.read_csv('vendite.csv')

# Esplorazione iniziale
print(vendite.head())
print(vendite.info())
print(vendite.describe())

# Filtraggio: vendite > 1000€
vendite_grandi = vendite[vendite['Importo'] > 1000]

# Nuova colonna: commissione (10%)
vendite['Commissione'] = vendite['Importo'] * 0.10
```

Esempio Pratico: Analisi vendite (Part 2)

```
# Raggruppamento per venditore
per_venditore = vendite.groupby('Venditore').agg({
    'Importo': 'sum',              # Totale
    'Commissione': 'mean',         # Media commissione
    'ID': 'count'                  # Numero transazioni
})

per_venditore.columns = ['Totale', 'Comm_Media', 'Transazioni']

# Ordinare per totale decrescente
per_venditore = per_venditore.sort_values('Totale', ascending=False)

# Salvare risultati
per_venditore.to_csv('report_venditori.csv')
```

Best Practices

- Usa loc e iloc per accesso efficiente
- Evita loop: usa operazioni vettorizzate
- Gestisci NaN prima delle analisi
- Usa groupby invece di loop for
- Copia esplicita con .copy () se necessario
- Verifica i tipi di dato con df.dtypes

Strumenti Consigliati

Jupyter Notebook

Interattivo, perfetto per l'analisi dati esplorativa

→ Data Science

PyCharm

IDE completo con debugging avanzato

→ Progetti grandi

Installazione

```
# Via pip
pip install pandas numpy matplotlib jupyter

# Via Anaconda (consigliato)
# Scarica Anaconda Distribution da anaconda.com
# Tutto è già incluso

# Verifica installazione
python -c "import pandas; print(pandas.__version__)"
```

Pandas + NumPy

```
import pandas as pd
import numpy as np

# Creare DataFrame da array NumPy
arr = np.random.randn(3, 3)
df = pd.DataFrame(arr, columns=['A', 'B', 'C'])

# Operazioni NumPy su DataFrame
risultato = np.sqrt(df)  # Radice quadrata

# Accesso a dati NumPy sottostanti
valori = df.values       # Ritorna array NumPy
```

Visualizzazione Integrata

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

df = pd.DataFrame({
    'Mese': ['Jan', 'Feb', 'Mar'],
    'Vendite': [100, 150, 200]
})

# Grafico direttamente da DataFrame
df.plot(x='Mese', y='Vendite', kind='bar')
plt.show()

# Tipi disponibili: line, bar, hist, scatter, box
```

Conversione Tipi di Dati

```
df = pd.DataFrame({
    'Numero': ['1', '2', '3'],
    'Data': ['2025-01-01', '2025-01-02', '2025-01-03']
})

# Convertire stringhe a numeri
df['Numero'] = df['Numero'].astype(int)

# Convertire a datetime
df['Data'] = pd.to_datetime(df['Data'])

# Verificare i tipi
print(df.dtypes)
```

Riepilogo Lezione

✓ Series e DataFrame: strutture 1D e 2D

✓ Accesso dati: loc, iloc, filtri booleani

✓ Lettura/Scrittura: CSV, JSON, Excel

✓ Aggregazioni: groupby, statistiche, merge

Domande?

Risorse utili:

→ pandas.pydata.org/docs

→ Real Python Pandas Guide

→ Kaggle Datasets (pratica)