

# Data analysis con pandas

Mattia Cozzi  
cozzimattia@gmail.com

a.s. 2024/2025



# Contenuti

Intro

Creare

Importare

Leggere

Manipolare

Salvare

## Cosa è pandas

Pandas è una libreria di Python per l'analisi, la manipolazione e la visualizzazione di dati.



I dati possono essere organizzati in [serie](#) (elenchi di valori) o in [dataframe](#) (tabelle di valori).

► [Guida ufficiale](#)

► [Guida di W3Schools](#)

## Installazione e importazione

Per installare pandas diamo da terminale:

```
pip install pandas
```

Per poter **visualizzare i dati** che manipoleremo, installiamo anche le estensioni “Excel Viewer” e “Rainbow CSV” in VS Code.

Per utilizzare pandas nei nostri script, iniziamo con:

```
1 import pandas as pd
2 #creo un alias per pandas, così tutte le volte
3 #posso richiamarlo usando solo due lettere
```

## Serie di dati semplici

Una serie di dati è un **elenco indicizzato di valori**:

```
1 import pandas as pd
2
3 #creazione di una lista di dati
4 serie = [177, 169, 191, 170, 155, 168]
5
6 #creazione di una serie a partire dalla lista
7 mySeries = pd.Series(serie)
8
9 #stampa della serie completa
10 print(mySeries)
11
12 #stampa di un solo elemento
13 print(mySeries[2])
```

## Esercizi

1. Crea una serie in pandas che contenga l'elenco dei tuoi voti e stampala a schermo.

## Serie di dati con etichette

Per assegnare un'etichetta ai dati, usiamo:

```
1 import pandas as pd
2 #un punto nello spazio, l'etichetta del DICT indica l'asse
3 punto = {"x": 12, "y": -3, "z": 6}
4
5 myPoint = pd.Series(punto)
6
7 #stampa della serie completa
8 print(myPoint)
9
10 #stampa di un solo elemento
11 print(myPoint["z"])
12
13 #creazione di una serie usando solo alcuni elementi
14 myPointProj = pd.Series(punto, index=["x", "y"])
15
16 print(myPointProj)
```

## Esercizi

2. Crea una serie in pandas che contenga i dati relativi ad un esame che hai dato (nome, voto, data). Stampa la serie a schermo.
3. Crea una serie identica a quella dell'esercizio precedente, ma stampa a schermo solo il nome dell'esame e il voto.



# Dataframe

I dataframe sono **tabelle di dati**:

```
1 import pandas as pd #ometteremo da qui in poi questa riga
2
3 #creazione del primo dataset: è un DICT
4 #in cui i valori delle chiavi sono delle LISTE
5 ds = {
6     "persone":
7         ↪ ["Andrea", "Beatrice", "Carlo", "Daniela", "Ernesto", "Fatima"],
8     "altezza": [177, 169, 191, 170, 155, 168]
9 }
10
11 #creazione di un dataframe
12 df = pd.DataFrame(ds)
13
14 #stampa del dataframe
15 print(df)
```

## Dataframe, esempio complesso

```
3  #un DICT di studenti
4  ds = {
5      "nomeStud": ["Andrea", "Beatrice", "Carlo"],
6      "cognomeStud": ["Rossi", "Verdi", "Neri"],
7      "matricolaStud": ["A1257", "G985", "R632"],
8      "mediaStud": [25.6, 28.2, 21.9]
9  }
10
11 #creazione dataframe
12 df = pd.DataFrame(ds)
13
14 #stampa del dataframe
15 print(df)
```

## Esercizi

4. Crea un dataframe in pandas che contenga i dati (nome, voto, data) relativi a tutti gli esami che hai dato. Stampa il dataframe a schermo.

## File CSV

I file in formato CSV sono **file di testo che contengono dati strutturati**, che possono essere mostrati come tabelle.

```
1 Username;Identifier;First name;Last name
2 booker12;9012;Rachel;Booker
3 grey07;2070;Laura;Grey
4 johnson81;4081;Craig;Johnson
5 jenkins46;9346;Mary;Jenkins
6 smith79;5079;Jamie;Smith
```

Nel corso è incluso il file **best-actor-age.csv**, con dati sui vincitori dell'Oscar come miglior attore/attrice e l'età al momento della vittoria.

La versione “grezza” di un file CSV è poco leggibile per gli utenti. Manipoliamo questi dati con pandas.

## Importare un file CSV

Per importare un file CSV come dataframe in Python, usiamo:

```
3  #il file CSV si trova nella stessa cartella del file PY
4  attori = pd.read_csv("best-actor-age.csv")
5
6  #se il file si trova in una sottocartella
7  # attori = pd.read_csv("./mydata/best-actor-age.csv")
8
9  #stampa le prime e le ultime righe
10 print(attori)
11
12 #stampa tutto
13 print(attori.to_string())
```

## Separatori

Il separatore dei dati di default in CSV è la virgola, ma potrebbe anche essere usata una tabulazione.

In tal caso, importiamo il file CSV con la seguente sintassi:

```
3 attori = pd.read_csv("best-actor-age.csv", sep="\t")
```

## Esercizi

5. Importa il file “best-actor-age.csv” e stampalo a schermo nella sua interezza.
6. Crea un file CSV a tua scelta, importalo in un algoritmo e stampane il contenuto. Il file CSV dovrà contenere almeno 5 record con almeno 4 attributi.

## File JSON

I file JSON sono file di testo semplice che contengono informazioni strutturate. Sono in tal senso molto simili ai file CSV, ma usano una sintassi diversa (e più leggibile).

```
1  [
2    { "Index": 1,
3      "Gender": "Female",
4      "Year": 1928,
5      "Age": 22,
6      "Name": "Janet Gaynor",
7      "Movie": "Seventh Heaven" },
8    { "Index": 2,
9      "Gender": "Female",
10     "Year": 1929,
11     "Age": 37,
12     "Name": "Mary Pickford",
13     "Movie": "Coquette" },
14  ]
```



## Importare un file JSON

Per importare un file JSON come dataframe in Python, usiamo:

```
3  #il file CSV si trova nella stessa cartella del file PY
4  attori = pd.read_json("best-actor-age.json")
5
6  #se il file si trova in una sottocartella
7  # attori = pd.read_json("./mydata/best-actor-age.json")
8
9  #stampa le prime e le ultime righe
10 print(attori)
11
12 #stampa tutto
13 print(attori.to_string())
```

## Importare un file di Excel

Per importare un file XLSX come dataframe in Python, usiamo:

```
3 attori = pd.read_excel("best-actor-age.xlsx")
4
5 #se non voglio includere i numeri di riga di Excel
6 # attori = pd.read_excel("best-actor-age.xlsx", index_col=0)
7
8 #se voglio importare solo un foglio
9 # attori = pd.read_excel("best-actor-age.xlsx", "nomeFoglio")
10
11 #stampa le prime e le ultime righe
12 print(attori)
13
14 #stampa tutto
15 print(attori.to_string())
```

## Esercizi

7. Riscrivi in formato JSON i dati del file CSV dell'esercizio precedente. Importa il file JSON creato in Python e stampare a schermo il contenuto.
8. Importa in Python il foglio "naso" del file "db-profumi.xlsx" e stampare a schermo il contenuto.
9. Importa in Python tutti i fogli del file "db-profumi.xlsx" e stampare a schermo il contenuto.

## Estrarre dati

Per estrarre dei dati da un file importato, uso:

```
5  #stampa i record da 0 (incluso) a 10 (escluso)
6  print(attori[0:10])
7
8  #stampa i primi 5 e gli ultimi 9 valori
9  print(attori.head(5))
10 print(attori.tail(9))
11
12 #stampa una sola colonna
13 print(attori["Name"])
14 #stampa alcune colonne (attenzione alle doppie parentesi!)
15 print(attori[["Name", "Movie"]])
16
17 #stampa alcuni valori di una colonna
18 print(attori["Name"][0:10])
19 print(attori["Name"].head(5))
```

## Esercizi

10. Scrivi un algoritmo che stampi a schermo i primi 10 record del file “best-actor-age.csv”.
11. Scrivi un algoritmo che stampi a schermo le colonne “Name” e “Year” degli ultimi 20 record del file “best-actor-age.csv”.

## loc

La “localizzazione” è la selezione, all'interno di una tabella, di una certa porzione di essa.

Possiamo estrarre i dati con:

- `loc`, che lavora sulle intestazioni della tabella;

```
3 #utilizzo come indice del dataframe la QUINTA colonna
4 #cioè quella che contiene il nome dell'attore/attrice
5 attori = pd.read_csv("best-actor-age.csv", index_col=4)
6
7 #stampa righe che hanno come indice (ovvero nome) Meryl Streep
8 print(attori.loc["Meryl Streep"])
9
10 #stampa solo certe colonne (attenzione alle parentesi!)
11 print(attori.loc[["Meryl Streep"], "Year"])
```

## iloc

- `iloc` (*index localization*), che lavora sugli **indici numerici** di riga e colonna;

```
3 attori = pd.read_csv("best-actor-age.csv")
4
5 #mostra le righe che hanno come indice 7
6 print(attori.iloc[7])
7
8 #mostra le righe in un certo range
9 print(attori.iloc[10:20])
10
11 #mostra solo una certa colonna
12 print(attori.iloc[7,5])
```

## Dimensione di un dataframe

Come intuibile, per avere la dimensione (cioè il numero di righe) di un dataframe, usiamo la funzione `len()`.

```
3 #leggo il dataframe iniziale
4 attori = pd.read_csv("best-actor-age.csv")
5
6 #stampo la dimensione del dataframe
7 print(len(attori))
```



## Esercizi

12. Scopri in quali anni Tom Hanks ha vinto l'Oscar come miglior attore.
13. Scopri quante volte Sean Penn ha vinto l'Oscar e con quali film.
14. Calcola la media delle età delle persone che hanno vinto l'Oscar. Non distinguere tra uomini e donne.

## Filtrare i dati

Se vogliamo porre specifiche condizioni sui dati che vengono estratti dal file di partenza, usiamo:

```
3  #leggo il dataframe iniziale
4  attori = pd.read_csv("best-actor-age.csv")
5
6  #stampo solo le righe relative alle attrici
7  print(attori[attori.Gender == "Female"])
8
9  #stampo le righe di attori con più di 60 anni
10 print(attori[attori.Age > 60])
11
12 #stampo le righe in base a più di una condizione
13 print(attori[(attori.Age > 60) & (attori.Gender=="Male")])
14 #attenzione alle parentesi!
```

## Esercizi

15. Scopri quanti attori uomini hanno vinto un Oscar a più di 60 anni. Stampa a schermo tutti i loro nomi.
16. Scopri quante attrici donne hanno vinto un Oscar a più di 60 anni. Stampa a schermo tutti i loro nomi.
17. Calcola la media delle età delle attrici che hanno vinto l'Oscar.
18. Calcola la media delle età degli attori che hanno vinto l'Oscar.

## Ordinare un dataframe

Ci sono diverse possibilità per riordinare i dati:

```
5  #salvo il dataframe ordinato in una nuova variabile
6  #l'ordine viene stabilito in base all'indice
7  attoriSorted = attori.sort_index()
8
9  #in ordine inverso
10 attoriSorted = attori.sort_index(ascending=False)
11
12 #ordina in base ad una certa colonna
13 attoriSorted = attori.sort_values(by="Age")
14
15 #ordina in base a più colonne
16 attoriSorted = attori.sort_values(by=["Age", "Year"],
17 ↪ ascending=[False, True])
18 print(attoriSorted)
```

## Esercizi

19. Mostra il dataframe relativo agli Oscar ordinando i dati per età dell'attore/attrice che ha vinto.
20. Mostra a schermo i nomi delle cinque donne più giovani ad aver vinto l'Oscar.
21. Mostra a schermo i nomi dei cinque uomini più vecchi ad aver vinto l'Oscar.

## Modificare i dati

Per modificare il valore di una cella, possiamo usare `loc`:

```
3 #leggo il dataframe iniziale
4 attori = pd.read_csv("best-actor-age.csv")
5
6 #imposto una condizione, scelgo una colonna e cambio il valore
7 attori.loc[attori["Movie"] == "Annie Hall", "Movie"] = "Io e
  ↳ Annie"
8
9 attori.loc[attori["Name"] == "Roberto Benigni", "Movie"] = "La
  ↳ vita è bella"
```

## Esercizi

22. Traduci il nome di almeno 3 film citati nel file “best-actor-age.csv” in italiano. Stampa a schermo i record modificati.
23. Traduci in italiano il nome di tutti i film con cui Marlon Brando ha vinto l'Oscar.

## Aggiungere dati (1)

Per aggiungere nuovi dati, **concateno** insieme due dataframe:

```
3 #leggo il dataframe iniziale
4 attori = pd.read_csv("best-actor-age.csv")
5
6 #creo un dataframe con i nuovi valori da una LISTA DI DICT
7 nuovo = pd.DataFrame([
8     {"Gender": "Male", "Year": 2017, "Age": 41, "Name": "Casey
9     ↪ Affleck", "Movie": "Manchester by the Sea"},
10    {"Gender": "Male", "Year": 2018, "Age": 60, "Name": "Gary
11    ↪ Oldman", "Movie": "Darkest Hour"}
12 ])
13
14 #concateno i due dataframe
15 final = pd.concat([attori,nuovo])
16
17 #stampo con una condizione
18 print(final[final["Year"] >= 2017])
```



## Aggiungere dati (2)

Una sintassi alternativa a quella precedente:

```
5  #creo un dict
6  nuoviValori = {
7      "Gender": "Male",
8      "Year": [2017, 2018],
9      "Age": [41, 60],
10     "Name": ["Casey Affleck", "Gary Oldman"],
11     "Movie": ["Manchester by the Sea", "Darkest Hour"]
12 }
13
14 #trasformo il dict in un dataframe
15 newDf = pd.DataFrame(nuoviValori)
16
17 #concateno i due dataframe
18 final = pd.concat([attori,newDf])
19
20 #stampo con una condizione
21 print(final[final["Year"] >= 2017])
```

## Esercizi

24. Scopri quante volte è successo che un attore e un'attrice abbiano vinto l'Oscar con lo stesso film. Stampa a schermo poi il nome del film, il nome dell'attore e il nome dell'attrice.
25. Completa i dati contenuti in "best-actor-age.csv" con le informazioni relative ai vincitori dell'Oscar tra il 2017 e il 2025 (sia uomini sia donne).

## Salvare un dataframe

Dopo aver manipolato un dataframe, possiamo salvarlo in un nuovo file:

```
3  #leggo il dataframe iniziale
4  attori = pd.read_csv("best-actor-age.csv")
5
6  #salvo il dataframe ordinato in una nuova variabile
7  attoriSorted = attori.sort_values(by="Age")
8
9  #salvo in CSV o XLSX specificando il nuovo nome
10 attoriSorted.to_csv("best-actor-sorted.csv")
11 attoriSorted.to_excel("best-actor-sorted.xlsx")
```

## Esercizi

26. Aggiorna il file dei vincitori dell'Oscar come miglior attore come indicato nell'esercizio precedente e salva il tutto in un nuovo file chiamato "best-actor-age-updated.csv".
27. Come sopra, ma salva in formato XLSX.
28. Salva in formato XLSX i dati relativi alle vittorie agli Oscar, per uomini e donne, precedenti alla fine della Seconda Guerra Mondiale.

## Esercizi

29. Scrivi un algoritmo che chieda all'utente se vuole convertire il file "best-actor-age-updated.csv" in formato XLSX o lasciarlo in CSV. Chiedi poi se vuole salvare solo i dati relativi alle donne, agli uomini o a entrambi. Salva poi il file così come l'utente ha richiesto.
30. Scrivi un algoritmo che chieda all'utente una data iniziale e una data finale. Salva in un file CSV i dati relativi alle vittorie agli Oscar negli anni compresi tra i due che l'utente ha inserito. Mostra a schermo il contenuto di tale file.