

# Matrici

Informatica@DSS 2025/2026

Massimo Lauria <[massimo.lauria@uniroma1.it](mailto:massimo.lauria@uniroma1.it)>  
<https://massimolauria.net/informatica2025/>

# Rappresentazione di matrici

Strumento essenziale in algebra: in **questo corso** rappresentiamo la matrice

1	2	3
4	5	6
7	8	9
10	11	12

```
[[1,2,3], [4,5,6], [7,8,9], [10,11,12]]
```

# Lista di liste di dimensione uguale

Nella rappresentazione di una matrice  $R \times C$

- ▶ ogni riga è una lista di  $C$  elementi
- ▶ la matrice è una lista di  $R$  righe

In questo modo per una matrice  $M$

- ▶  $M[r]$  è la  $r$ -esima riga
- ▶  $M[r][c]$  è elemento in riga  $r$  e colonna  $c$

```
M = [[1,2,3], [4,5,6], [7,8,9], [10,11,12]]  
print(M[2][1])
```

1

2

# Matrice ben formata

Nei nostri esercizi consideriamo **ben formata** una rappresentazione di matrice  $R \times C$  che è

- ▶ una lista di liste contenente almeno una riga:  $R \geq 1$
- ▶ tutte le liste interne hanno la stessa lunghezza  $C$
- ▶ esiste almeno una colonna:  $C \geq 1$

Esempi di matrici **mal formate**

[ [ 1,2,3], [1,2] ]	# righe di dimensione diversa	1
[ ]	# nessuna riga	2
[ [], [] ]	# nessuna colonna	3

# Ricavare le dimensioni di una matrice

La nostra rappresentazione di una matrice  $M$  permette di accedere facilmente alle **dimensioni nella matrice**

- ▶  $\text{len}(M)$  è il numero di righe  $R$
- ▶  $\text{len}(M[0])$  è il numero di elementi della prima riga, cioè il numero di colonne  $C$

```
M = [[1,2,3], [4,5,6], [7,8,9], [10,11,12]]  
print("R =", len(M))  
print("C =", len(M[0]))
```

1

2

3

```
R = 4  
C = 3
```

# Una cella $(r, c)$ della matrice:

- ▶ un indice di riga  $0 \leq r < R$     range(R)
- ▶ un indice di colonna  $0 \leq c < C$     range(C)

```
M = [[1,2,3], [4,5,6], [7,8,9], [10,11,12]]  
R, C = len(M), len(M[0])  
for r in range(R):  
    for c in range(C):  
        print((r,c),end=" ")  
    print()
```

1  
2  
3  
4  
5  
6

```
(0, 0) (0, 1) (0, 2)  
(1, 0) (1, 1) (1, 2)  
(2, 0) (2, 1) (2, 2)  
(3, 0) (3, 1) (3, 2)
```

# Stampiamo una matrice

```
def matrixprint(M):          1
    R,C = len(M), len(M[0])  2
    for r in range(R):       3
        for c in range(C):   4
            print(M[r][c],end=' ')
        print()               5
    print()                  6
    print()                  7
M = [[1.2, 5.2, -1, -2, 1], 8
      [0.7, -9.1, 11, -2, 0.4], 9
      [0,0,0,1.5,3.5]]        10
matrixprint(M)              11
```

```
1.2 5.2 -1 -2 1
0.7 -9.1 11 -2 0.4
0 0 0 1.5 3.5
```

**Esercizio:** formattate meglio la stampa

# Creiamo una matrice tutta a zero

```
def matrixcreate(R,C,fillvalue=0):          1
    M=[]                                     2
    for i in range(R):                      3
        M.append([fillvalue]*C)              4
    return M                                 5
                                         6
A = matrixcreate(4,10)                     7
matrixprint(A)                           8
print()                                  9
B = matrixcreate(4,12,9)                  10
matrixprint(B)                           11
```

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

9 9 9 9 9 9 9 9 9 9 9 9
9 9 9 9 9 9 9 9 9 9 9 9
9 9 9 9 9 9 9 9 9 9 9 9
9 9 9 9 9 9 9 9 9 9 9 9
```

# Una nuova lista per ogni riga

Ricordate che una lista è un **riferimento** ad un contenitore di valori.

```
L = [0,0,0,0]          1  
M = [L, L, L]      # sbagliato!    2  
  
M[0][2] = 1          3  
matrixprint(M)      4  
                      5
```

```
0 0 1 0  
0 0 1 0  
0 0 1 0
```

Le tre righe in questo caso sono riferimenti alla stessa lista.

# Esempi elaborazioni di matrici

# Matrice casuale

Qui riutilizziamo `matrixcreate` per creare la matrice da riempire, e la riempiamo con valori casuali.

```
import random  
1  
  
def matrixrandomfill(A):  
2    R,C = len(A),len(A[0])  
3    for r in range(R):  
4        for c in range(C):  
5            x = random.random()      # numero casuale in [0,1)  
6            A[r][c] = round(x,2)   # due cifre decimali  
7  
8    M = matrixcreate(5,12)  
9    matrixrandomfill(M)  
10   matrixprint(M)  
11  
12
```

```
0.64 0.6 0.6 0.16 0.67 0.85 0.19 0.26 0.8 0.33 0.32 0.99  
0.98 0.94 0.55 0.77 0.31 0.51 0.74 0.66 0.04 0.26 0.62 0.72  
0.51 0.06 0.14 0.84 0.42 0.3 0.85 0.44 0.71 0.8 0.52 0.19  
0.6 0.01 0.6 0.66 0.83 0.09 0.93 0.36 0.44 0.55 0.85 0.26  
0.32 0.79 0.68 0.85 0.21 0.22 0.56 0.04 0.76 0.89 0.29 0.2
```

# Matrice identità

Matrice quadrata con 1 nella diagonale e 0 altrove

```
def matrixI(n):
    I = matrixcreate(n,n)
    for i in range(n):
        I[i][i] = 1
    return I

I = matrixI(6)
matrixprint(I)
```

1  
2  
3  
4  
5  
6  
7  
8

1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0

## Sommiamo i valori in una matrice

Vogliamo la somma di tutti i valori nella matrice, ad esempio per la matrice

11	-2.1	6
-4	4.5	0.5
17	-6	0
0.12	-21	1.2

la somma di tutti i valori al suo interno è 7.22

# Sommiamo i valori in una matrice (2)

```
M = [[11,-2.1,6],[-4,4.5,0.5],  
     [17,-6,0],[0.12,-21,1.2]]  
  
def matrixsum(M):  
    "Somma tutti i valori contenuti in una matrice"  
    R,C = len(M), len(M[0])  
    acc = 0  
    for r in range(R):  
        for c in range(C):  
            acc += M[r][c]  
    return acc  
  
print(matrixsum(M))
```

7.22

# Trasposta di una matrice

Data una matrice  $A$  di dimensioni  $R \times C$ ,

la trasposta  $A^T$  è una matrice di dimensioni  $C \times R$

tale che ogni elemento  $A_{i,j}^T$  sia uguale all'elemento  $A_{j,i}$ .

$$A = \begin{bmatrix} 11 & -2.1 & 6 \\ -4 & 4.5 & 0.5 \\ 17 & -6 & 0 \\ 0.12 & -21 & 1.2 \end{bmatrix} \quad A^T = \begin{bmatrix} 11 & -4 & 17 & 0.12 \\ -2.1 & 4.5 & -6 & -21 \\ 6 & 0.5 & 0 & 1.2 \end{bmatrix}$$

# Trasposta di una matrice (2)

```
def matrixtranspose(A):  
    "Produce la matrice trasposta di A"  
    R,C = len(A), len(A[0])  
    AT = matrixcreate(C,R)  
    for r in range(R):  
        for c in range(C):  
            AT[c][r] = A[r][c]  
    return AT  
  
M = [[11,-2.1,6],[-4,4.5,0.5],  
      [17,-6,0],[0.12,-21,1.2]]  
MT = matrixtranspose(M)  
matrixprint(MT)
```

```
11 -4 17 0.12  
-2.1 4.5 -6 -21  
6 0.5 0 1.2
```

## Posizione della riga di somma massima

Vogliamo sapere quale riga contiene gli elementi la cui somma è la massima possibile. Nella matrice

11	-2.1	6
-4	4.5	0.5
17	-6	0
0.12	-21	1.2

la riga 0 ha somma 14.9, la riga 1 ha somma 1, la riga 2 ha somma 11, la riga 3 ha somma -19.68.

Quindi la riga di somma massima è la riga 0.

# Posizione della riga di somma massima (2)

```
def matrix_maxrow(A):          1
    "Posizione della riga di somma massima" 2
    R = len(A)           3
    rowmax=0            4
    rowval=sum(A[0])    5
    for r in range(1,R): 6
        tmp = sum(A[r]) 7
        if tmp>rowval:   8
            rowmax=r     9
            rowval=tmp    10
    return rowmax         11
                                12
M = [[11,-2.1,6],[-4,4.5,0.5], 13
      [17,-6,0],[0.12,-21,1.2]] 14
r = matrix_maxrow(M)           15
print(r)                      16
```

0