

# Matrici

Informatica@DSS 2023/2024

Massimo Lauria <massimo.lauria@uniroma1.it>  
<https://massimolauria.net/informatica2023/>

# 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

8

# 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**

- ▶ `len(M)` è il numero di righe  $R$
- ▶ `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]]      1
print("R =",len(M))                             2
print("C =",len(M[0]))                          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]]      1
R, C = len(M), len(M[0])                        2
for r in range(R):                               3
    for c in range(C):                           4
        print((r,c),end=" ")                     5
    print()                                       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=' ') 5
        print() 6
    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

A = matrixcreate(4,10)                       6
matrixprint(A)                               7
print()                                      8
B = matrixcreate(4,12,9)                     9
matrixprint(B)                              10
                                             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
```



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

```
0.72 0.09 0.4 0.39 0.72 0.06 0.8 0.79 0.36 0.92 0.03 0.64
0.62 0.75 0.87 0.29 0.04 0.85 0.57 0.79 0.75 0.43 0.98 0.94
0.38 0.95 0.84 0.72 0.86 0.3 0.57 0.41 0.25 0.84 0.35 0.15
0.42 0.63 0.68 0.43 0.03 0.14 0.29 0.54 0.18 0.7 0.7 0.77
0.31 0.22 0.53 0.81 0.77 0.12 0.51 0.0 0.74 0.79 0.73 0.75
```

# Matrice identità

Matrice quadrata con 1 nella diagonale e 0 altrove

```
def matrixI(n): 1
    I = matrixcreate(n,n) 2
    for i in range(n): 3
        I[i][i] = 1 4
    return I 5
6
I = matrixI(6) 7
matrixprint(I) 8
```

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

# Sommiamo i valori in una matrice

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

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

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  $A_{j,i}$ .

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

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

## Trasposta di una matrice (2)

```
def matrixtranspose(A):           1
    "Produce la matrice trasposta di A"           2
    R,C = len(A), len(A[0])           3
    AT = matrixcreate(C,R)           4
    for r in range(R):           5
        for c in range(C):           6
            AT[c][r] = A[r][c]           7
    return AT           8
                                     9
M = [[11,-2.1,6],[-4,4.5,0.5],           10
      [17,-6,0],[0.12,-21,1.2]]           11
MT = matrixtranspose(M)           12
matrixprint(MT)           13
```



# Posizione della riga di somma massima

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

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

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