

# LABORATORIO DI PYTHON

## LISTE ANNIDATE: LE MATRICI IN PYTHON

---

16 Aprile 2019

**CORREZIONE ESERCIZI SULLE LISTE**

Scrivere una funzione che verifica se una lista è ordinata in modo crescente ( $\leq$ ) (supponiamo che la lista contenga oggetti ordinabili)

## ESERCIZIO 1

Scrivere una funzione che verifica se una lista è ordinata in modo crescente ( $\leq$ ) (supponiamo che la lista contenga oggetti ordinabili)

```
1 def ordinataCrescente(L):  
2     if len(L)==0:  
3         return True  
4     for i in range(1,len(L)):  
5         if L[i]<L[i-1]:  
6             return False  
7     return True
```

Scrivere una funzione che presi come parametri due liste **ordinate in modo crescente** (verificarlo usando la funzione dell'esercizio precedente) ne crei una terza, anch'essa ordinata, dall'unione delle due.

```
1 import Es1L08 as ol
2
3 def unisciOrdinate(L1, L2):
4     if (not ol.ordinataCrescente(L1)) or (not ol.
5         ordinataCrescente(L2)):
6         print("Le liste devono essere ordinate in modo crescente")
7         return None
8     LR = []
9     i = 0
10    j = 0
11    while i < len(L1) and j < len(L2):
12        if L1[i] <= L2[j]:
13            LR.append(L1[i])
14            i += 1
15        else:
```

```
15         LR.append(L2[j])
16         j+=1
17
18     if i<len(L1):
19         LR = LR + L1[i:]
20     else:
21         LR = LR + L2[j:]
22
23     return LR
```

## ESERCIZIO 3

Scrivere una funzione `ripetis(s)` che data una lista di numeri naturali `s`, restituisce una nuova lista in cui compaiono gli stessi elementi di `s` e nello stesso ordine, ma ciascun `s[i]` è ripetuto `s[i]` volte. (Esempio: da `[3,0,2]` si deve ottenere `[3,3,3,2,2]`.)



## ESERCIZIO 3

Scrivere una funzione `ripetis(s)` che data una lista di numeri naturali `s`, restituisce una nuova lista in cui compaiono gli stessi elementi di `s` e nello stesso ordine, ma ciascun `s[i]` è ripetuto `s[i]` volte. (Esempio: da `[3,0,2]` si deve ottenere `[3,3,3,2,2]`.)

```
1 def ripetis(s):  
2     res = []  
3     for e in s:  
4         res += [e]*e  
5     return res
```

Scrivere una funzione analoga analoga alla precedente, ma che **modifica** la lista `s`.

## ESERCIZIO 4

Scrivere una funzione analoga alla precedente, ma che **modifica** la lista **s**.

```
1 def ripetisposto(s):  
2     i = 0  
3     while i < len(s):  
4         e = s[i]  
5         del s[i]  
6         s[i:i] = [e]*e #inserimento sottolista  
7         i = i+e
```

# MATRICI

Come possiamo rappresentare una matrice  $A^{r \times c}$  in Python? Per esempio, la matrice  $A^{3 \times 2}$ :

$$A = \begin{pmatrix} 0 & 1 \\ 3 & 2 \\ 5 & 6 \end{pmatrix}$$

può essere rappresentata come una lista di liste.

```
A = [[0,1],[3,2],[5,6]]
```

Notiamo che:

- la notazione è coerente: quanto vale  $A[1][1]$ ?
- $\text{len}(A) = 3$ ,  $\text{len}(A[0]) = 2$

I vettori (matrici unidimensionali) sono vettori riga e vengono rappresentati come una lista che ha come primo elemento una lista. Dunque:

- $[1, 2, 3]$  è una lista di numeri, **ma non una matrice** (e dunque neanche un vettore riga)
- $[[1, 2, 3]]$  è un vettore riga, cioè una matrice
- $[[1], [2], [3]]$  è un vettore colonna, trasposto del precedente.

- In tutti gli esercizi seguenti – se non diversamente specificato – supporremo per brevità che l'utente inserisca matrici e vettori “ben formati” (dunque con la sintassi espressa in precedenza, tutte le righe della stessa lunghezza, elementi tutti numerici...)
- Nel codice allegato alla lezione verranno fornite anche le funzioni necessarie a controllare che l'input rappresenti matrici e vettori “ben formati”
- Gli esercizi seguenti sono tutte funzioni ausiliarie necessarie per calcolare il prodotto tra due matrici
- Consiglio: svolgere tutti gli esercizi nello stesso file chiamato `matrici.py`

Scrivere una funzione che stampa una matrice presa come parametro Esempio se  $A = \begin{bmatrix} 10000 & 12 \\ 3 & 2 \\ 5 & 6 \end{bmatrix}$  stampa esattamente:

10000	12
3	2
5	6

Suggerimento: usare `print(e, end='\t')`



Scrivere una funzione che, presi come parametri due **liste** (non vettori riga) moltiplicabili (verificare che abbiano la stessa lunghezza), calcola e restituisce il prodotto scalare, oppure **None**.

Scrivere tre funzioni:

- Una funzione che prende come parametro una matrice e restituisce il numero delle sue righe
- Una funzione che prende come parametro una matrice e restituisce il numero delle sue colonne
- Una funzione che verifica se due matrici prese come parametri possono essere moltiplicate tra loro

Scrivere una funzione che calcola e restituisce la trasposta di una matrice presa come parametro.

Scrivere una funzione che restituisce la matrice prodotto tra due matrici prese come parametri (se non sono moltiplicabili, stampa un messaggio di errore e restituisce **None**).

Suggerimento: usare la matrice trasposta per fare il prodotto scalare tra le righe di A e le righe della trasposta di B.

## ESERCIZI PER CASA

Anche **usando le funzioni viste a lezione**:

1. Scrivere una funzione che prende come parametri una matrice e un numero (verificare che lo sia) e restituisce **una nuova matrice** che costituisce il prodotto della matrice per quel numero.
2. Scrivere una funzione che prende come parametro una matrice, verifica che sia costituita solo da numeri interi  $\geq 0$  (se non è così, stampa errore e termina), e **modifica** la matrice (senza usare quindi altre liste) NON restituisce nulla.
3. Scrivere una funzione che prende come parametri:
  - una matrice non vuota di interi
  - un intero  $n$

e restituire la coppia  $(i, j)$  che identifica la posizione del primo elemento (leggendo la matrice per righe) che, sommato a tutti i suoi precedenti, dia come risultato un valore  $> n$ . Se la somma di tutti gli elementi è minore di  $n$ , restituire la coppia  $(\text{math.inf}, \text{math.inf})$ . Usare il ciclo **while** ma non il ciclo **for**.