LABORATORIO DI PYTHON

LISTE ANNIDATE: LE MATRICI IN PYTHON

16 Aprile 2019

Correzione Esercizi sulle Liste

ESERCIZIO 1

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
```

ESERCIZIO 2

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.

ESERCIZIO 2 PER CASA I

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

ESERCIZIO 2 PER CASA II

```
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
```

ESERCIZIO 4

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

ESERCIZIO 4

Scrivere una funzione analoga 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

MATRICI COME LISTE DI LISTE

Come possiamo rappresentare un 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]`?
- `len(A)` = 3, `len(A[0])` = 2

MATRICI COME "VETTORI"

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.

SUGGERIMENTI

- 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`

ESERCIZIO

Scrivere una funzione che stampa una matrice presa come parametro Esempio se `A = [[10000,12],[3,2],[5,6]]` stampa esattamente:

```
10000    12
3        2
5        6
```

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

ESERCIZIO

```python
def stampaMatrice(M):
    for riga in M:
        for e in riga:
            print(e, end='\t')
        print() #a capo dopo ogni riga
```

O con indici espliciti:

```python
def stampaMatrice(M):
    for i in range(len(M)):
        for j in range(len(M[i]):
            print(M[i][j], end='\t')
        print() #a capo dopo ogni riga
```

ESERCIZIO

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`.

ESERCIZIO

```
1  def molt_liste(U,V):
2      if len(U)==len(V):
3          pr_sc = 0
4          for i in range(len(V)):
5              pr_sc = pr_sc + (U[i]*V[i])
6          return pr_sc
7      else:
8          return None
```

UN'ALTRA POSSIBILE SOLUZIONE IN STILE PYTHON

```
1  def molt_liste_2(U,V):
2      if len(U)==len(V):
3          pr_sc = 0
4          for eu, ev in zip(U, V): #lo posso fare perche' hanno la
       stessa len
5              pr_sc = pr_sc + eu*ev
6          return pr_sc
7      else:
8          return None
```

ESERCIZI DA UNA RIGA

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

UN'ALTRA POSSIBILE SOLUZIONE IN STILE PYTHON

```
1  def righe(M):
2      return len(M)
3
4  def colonne(M):
5      return len(M[0])
6
7  def sono_matr_molt(M,N):
8      return colonne(M) == righe(N)
```

ESERCIZIO

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

SOLUZIONE

```
1  def trasposta(M):
2      MT = []
3      r = righe(M)
4      if r == 0: #vuota
5          return MT
6      else:
7          c = colonne(M)
8          for i in range(c): #ciclo sulle colonne
9              col = [] #i-ma colonna
10             for j in range(r): #ciclo sulle righe
11                 col.append(M[j][i])
12             #ora in col ho la i-ma colonna, che diventa l'i-ma
       riga
13             MT.append(col)
14         return MT
```

ESERCIZIO

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.

SOLUZIONE

```
1  def molt_mat(A,B):
2      if sono_matr_molt(A,B): # se sono molt.
3          AB = [] #prodotto
4          BT = trasposta(B) #trasposta di B
5          for i in range(righe(A)): #rig. di A (e AB)
6            riga = [] #i-ma riga di AB
7            #j scorre righe di BT (= col di B e AB)
8            for j in range(righe(BT)):
9                  ij = molt_liste(A[i],BT[j])
10                 riga.append(ij) #elemento in riga
11           AB.append(riga) #riga in risultato
12         return AB
13     else:
14         print("Matrici non moltiplicabili")
15         return None
```

Esercizi per casa

ESERCIZI PER CASA

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

1. Scrivere una funzione che prende come parametri una matrice e un numero e restituisce **una nuova matrice** che costituisce il prodotto della matrice per quel numero.
2. Scrivere una funzione che prende come parametri una matrice e un numero e **modifica** la matrice che costituisce il prodotto della matrice per quel numero. 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 `((math.inf),math.inf))`. Usare il ciclo **while** ma non il ciclo **for**.