Laboratorio di Python

Esercizi sulle liste

25 marzo 2015

Sommario

Esercizi a casa

- Scrivere un unico programma, che attraverso un menù di selezione restituisca i risultati delle funzioni che seguono, senza uscire dal programma se non selezionando il numero 4.
  - la somma dei primi *n* numeri pari, dove *n* è richiesto in input (soluzione iterativa con for)
  - la somma dei primi *n* numeri dispari, dove *n* è richiesto in input (soluzione iterativa con while)
  - la serie geometrica di *n* numeri, dove *n* è richiesto in input (soluzione ricorsiva)
- Inviate gli esercizi svolti a: **labinfo.mat.unibo@gmail.com**

Correzione

```
def SOMMA_PARI(n):
    if type(n)==int or type(n)==float:
        somma=0
        if n>0:
            for i in range(0,2*(n+1), 2):
                somma = somma+i
        return somma
def SOMMA_DISPARI(n):
    if type(n)==int or type(n)==float:
        somma=0
        i=0
        if n>0:
            while (i<n):
                somma = somma +(i*2)+1
                i=i+1
        return somma
```

Correzione

```
def SERIE_GEOMETRICA(x,n):
    if n==0:
        return 1
    else:
        return x**n+seriegeometrica(x,n-1)
```

Correzione

```
def MULTI ():
 print('Digita 1 per ottenere la somma dei primi n numeri pari')
 print('Digita 2 per ottenere la somma dei primi n numeri dispari')
 print('Digita 3 per ottenere la serie geometrica di x fino ad n')
 print('Digita 4 per uscire dal programma')
 w=int(input("Digita la tua scelta: "))
 while w<>4:
   if w==1:
      a=int(input('Inserisci un numero: '))
      print(SOMMA_PARI(a))
      w=int(input("Digita la tua scelta: "))
   elif w==2: print(SOMMA_DISPARI(a))
   elif w==3:
      x=int(input('Inserisci il numero: '))
      n=int(input("Inserisci l'indice: "))
      print(SERIE_GEOMETRICA(x,n))
      w=int(input("Digita la tua scelta: "))
   else:
       w=int(input("Scelta non valida, fai una nuova scelta: "))
```

Esempi

Risolvere i seguenti problemi:

1. Data una lista dire se essa può rappresentare una matrice.
2. Date due liste che rappresentano due matrici dire se sia possibile moltiplicarle.
3. Date due liste che rappresentano due vettori, calcolare il prodotto scalare dei due vettori nel caso si possa applicare.
4. Data una matrice restituire la matrice trasposta.
5. Date due matrici calcolarne il prodotto.

Esempi

Come possiamo rappresentare un matrice in python?

$$A^{r \times k}$$

$$A^{3 \times 2}$$

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

Come una lista di liste

$$M = [[0, 1], [3, 2], [5, 6]]$$

$$3 = len(M),\ 2 = len(M[0])$$

$$r = len(M),\ k = len(M[0])$$

Esercizio 1

```
def se_numero(t):
# restituisce True se t è una sequenza di numeri
      for i in t:
            if type(i)<>int and type(i)<> float:
                  return False
      return True
def se_matrice(t):
#Restituisce True se è t è una matrice, False altrimenti
      if type(t)<>list:
            return False
      for i in t:
            if not( type(i)==list and se_numero(i)):
                  return False
            elif len(i)<>len(t[0]):
                  return False
      return True
```

Esercizio 2

```
def righe_colonne(t):
# restituisce il numero di righe e colonne di una matrice,
# se t non è una matrice restituisce False,False
    if se_matrice(t):
        return len(t),len(t[0])
    else:
        return False, False

def se_molt(t,v):
# restituisce True se posso moltiplicare t,v False altrimenti
    a,b= righe_colonne(t)
    c,d=righe_colonne(v)
    return b==c
```

Esercizio 3

```
def se_molt_vet(x,y):
# Restituisce True se posso moltiplicare i vettori False altrimenti
    return len(x)==len(y)

def molt_vet(x,y):
# Restituisce il prodotto scalare se è possibile calcolarlo
#altrimenti restituisce False
    if se_molt_vet:
        ris=0
        for i in range(0,len(x)):
            ris=ris+x[i]*y[i]
        return ris
    else:
        return False
```

Calcolo della matrice trasposta - Liste

```
def trasposta(v):
 if se_matrice(v):
   a=[] #sequenza in cui memorizzo la mia trasposta
   for i in range (0, len(v[0])): # ciclo sulla colonna
      c=[]
      for j in range(0, len(v)): # ciclo su tutte le righe
         c.append(v[j][i])#elem. in posizione (i,j) ha posizione (j,i)
      a.append(c) #inserisco la riga alla mia matrice trasposta a
 else:
      a='inserire una matrice'
 return a
```

TxV

```
def molt_mat(t,v):
    if se_molt(t,v): # se posso eseguire la molt
        c=[] #sequenza di output
        r=trasposta(v) #trasposta di r
        for i in range(0,len(t)):
          rig=[]
          for j in range(0, len(r)):#ciclo sulle righe della trasposta
                k=molt_vet(t[i],r[j]) #prodotto scalare
                rig.append(k) #inseriesco elemento nella riga
          c.append(rig) #inserisco la riga in c
        return(c) #restituisco il ris
    else:
        return(False)

def se_molt(t,v):
# funzione che mi calcola restituisce True se posso moltiplicare
#le matrici, False altrimenti
    a,b= righe_colonne(t)
    c,d=righe_colonne(v)
    return b==c
```

Esercizi per casa

Scrivere e documentare le funzioni che risolvano i seguenti problemi:

1. Definire una funzione che presa una sequenza, come parametro restituisca il numero dei valori valle; un elemento appartenente a una determinata sequenza si definisce valle se s[i] <s[i-1] e s[i]<s[i+1]
2. Definire una funzione che presa una sequenza, come parametro restituisca il numero dei valori apice; un elemento appartenente a una determinata sequenza si definisce apice se s[i] >s[i-1] e s[i]>s[i+1] Inviare gli esercizi a: **labinfo.mat.unibo@gmail.com**

Cosa abbiamo fatto?