For e strutture dati complesse - Esercizi

Mirko Cesarini - Dario Pescini
nome.cognome@unimib.it

Università di Milano Bicocca

Numerosità caratteri in una struttura dati complessa

Scrivete uno script python che conti la numerosità dei caratteri contenuti nelle stringhe di una struttura dati complessa

```
1  ds=[ # (nome, #cognome, #composizione_letteraria)
2         ('mario','rossi', 'i computer fanno esattamente quello
3                                                 che gli viene detto'),
4         ('giorgio', 'verdi', 'sono convinto che gli anelli di
5           saturno siano composti da tutte le valigie perse
6           negli aeroporti')                    ]
7  chr2num={}
8  for tu in ds:
9      for st in tu:
10         for ch in st:
11             if ch in chr2num:
12                 chr2num[ch]=chr2num[ch]+1
13             else:
14                 chr2num[ch]=1
15 print('Numerosita')
16 for ch in chr2num:
17     num=chr2num[ch]
18     print(ch, num)
```

```
Numerosita
('a', 10)
(' ', 23)
('c', 5)
('e', 19)
('d', 4)
('g', 6)
('f', 1)
('i', 18)
('h', 2)
('m', 4)
...
```

Fattoriale rivisitato

- Scrivere un programma che chieda all'utente un numero e ne calcoli il fattoriale utilizzando il ciclo for
- Soluzione

```
1 n=int(input('Inserisci un numero '))
2 prod=1
3 for el in range(1,n+1):
4     prod=prod*el
5 print("Il fattoriale di %d e' %d" % (n,prod))
```

- Domanda, cosa cambierebbe se a riga 3 scrivessimo `for el in range(1,n):` ? Risp.: calcoleremmo il fattoriale di n-1, poiché `range(1,n+1)`=[1,2, ..., n−1, n]
- E se a riga 3 scrivessimo `range(n)` ? Risp.: otterremmo 0 poiché `range(n)`=[0,1,2, ..., n−1]

Triangolo di Floyd rivisitato

Scrivete uno script python che visualizza a video le prime n righe del triangolo di Floyd.

- Soluzione

```
1 nr=5
2 i=1
3 for riga in range(nr):
4     s='' # creo una stringa vuota
5     for k in range(riga+1):
6         s=s+'%3d  ' % (i)
7         i=i+1
8     print(s)
```

```
  1
  2   3
  4   5   6
  7   8   9  10
 11  12  13  14  15
```

Individuazione outlier

Supponete di avere una lista li contenente dei valori numerici. Dovete individuare gli outlier (i valori esterni all'intervallo *media* $\pm$ 3 *scarto quadratico medio*)

```
1  import random
2  li=[]
3  # Inserisco 500 numeri
4  for i in range(500):
5      li.append(
6          random.randint(1,10))
7
8  # Inserisco 5 outlier
9  for i in range(5):
10     li.append(
11         random.randint(1,100))
12
13 # calcolo la media
14 numeratore=0
15 for el in li:
16     numeratore+=el
17 media=float(numeratore)/len(li)
```

```
18 # calcolo lo scarto
19 sommaq=0
20 for el in li:
21     sommaq+=(el-media)**2
22 sqm=(sommaq/len(li))**0.5
23
24 # identifico gli outlier
25 outlier=[]
26 for el in li:
27     if el < media-3*sqm or
28             el > media+3*sqm:
29         outlier.append(el)
30
31 print(outlier)
```

Tokenizzazione di una frase

Data una stringa, composta da diverse parole separate da spazi, scrivete un algoritmo in grado di individuare le parole (separate) e inserirle in una lista. Per es., `'questa frase va divisa'` dovrebbe produrre `['Questa', 'frase', 'va', 'divisa']`.

```
1  testo="Questa frase va divisa"
2  lifinale=[]
3  temp=[]
4  for ch in testo:
5      if ch!=' ':
6          temp.append(ch)
7      else:
8          parola=''
9          for sc in temp:
10             parola+=sc
11         lifinale.append(parola)
12         temp=[]
13 print(lifinale)
```

- Secondo voi viene individuato il risultato corretto?
- Esempio di risultato

  `['Questa', 'frase', 'va']`

- Cosa c'è che non va?

Tokenizzazione - versione migliorata

```
1  testo="Questa frase va divisa"
2  lifinale=[]
3  temp=[]
4  for ch in testo:
5      if ch!=' ':
6          temp.append(ch)
7      else:
8          parola=''
9          for sc in temp:
10             parola+=sc
11         lifinale.append(parola)
12         temp=[]
13 if len(temp)>=0:
14     parola=''
15     for sc in temp:
16         parola+=sc
17     lifinale.append(parola)
18 print(lifinale)
```

```
['Questa', 'frase', 'va', 'divisa']
```

- Vedremo più avanti che esistono delle funzioni di libreria che rendono più semplice svolgere questa operazione

Incrementi casuali

- Scrivete uno script python che visualizzi a video il numero 1,
  - incrementate il numero stampato in precedenza di un valore casuale (compreso tra 1 e 10) e stampate a video il nuovo numero
  - ripete l'operazione precedente diverse volte, fino a che il numero incrementato si mantiene inferiore a 100
- Provate ad ipotizzare una soluzione
- Esempio

```
1 import random
2 i=1
3 while i<100:
4     print(i)
5     i=i+random.randint(1,10)
```

- Se aveste voluto utilizzare *for* in questo esercizio?
- Sarebbe stato molto difficile. Il for mal si adatta a quelli scenari in cui la numerosità delle iterazioni non è determinabile a priori (es., conoscendo il numero di elementi di un insieme)

Somma di numeri in strut. dati complessa

- Create una lista di liste
- La lista esterna deve contenere un numero di elementi pari a k
- Ogni elemento deve essere a sua volta una lista di 5 interi generati casualmente, con queste caratteristiche
  - il primo numero deve essere estratto casualmente nell'intervallo [0, 4]
  - i successivi 4 numeri devono essere estratti casualmente nell'intervallo [0,100]
  - es., con k=2

```
1 [
2   [4, 31, 93, 70, 21],
3   [1, 98, 43, 90, 39],
4  ]
```

- Scrivete uno script python che:
  - estragga da ogni lista interna il valore il cui indice è individuato dal primo elemento della lista interna (es., in [2, 29, 71, 43, 92] dovra' essere estratto 71, cioé l'elemento di indice 2)
  - sommi tutti i valori estratti e stampi a video il risultato

Soluzione

```
1  import random
2  k=5
3  lext=[]
4  for i in range(k):
5      li=[random.randint(0,4)]
6      for j in range(4):
7          ncas=random.randint(0,100)
8          li.append(ncas)
9      lext.append(li)
10
11 sum=0
12 for i in range(len(lext)):
13     pos=lext[i][0]
14     sum=sum+lext[i][pos]
15 print(sum)
```