DATI STRUTTURATI: LE LISTE

Le **liste** sono oggetti mutabili il cui valore è rappresentato da sequenze di dati che possono essere modicati.
Gli elementi che le compongono sono oggetti arbitrari.
Sono definite da una espressione posta tra parentesi quadre con gli elementi separati da virgole. Come le stringhe, vengono indiciate in modo da poter accedere in modo semplice al loro contenuto.
Apri un terminale (per esempio konsole) ed entra in Python in modalità interattiva (basta scrivere **python).**
Esegui questo esempio:

```
>>> lista = ['Cagliari', 'Inter', 'Lazio', 12]
>>> lista[0]         # nota come la numerazione parta da zero
'Cagliari'
>>> lista[1:3]       # elenca gli elementi dal posto 1 incluso al posto 3 escluso
['Inter', 'Lazio']
>>> lista +['Juventus']
['Cagliari', 'Inter', 'Lazio', 12, 'Juventus']
>>> len(lista)
4
>>>
```

L'esempio precedente mostra come si possa intervenire sulla lista e vediamo l'uso della funzione `len()` che restituisce la “lunghezza” della lista.
Esistono alcune funzioni specifiche (si chiamano **metodi**) per agire su una lista:

- **insert(i,x)** Inserisce l'elemento x nella posizione i della lista.
- **append(x)** Inserisce l'elemento x in coda alla lista.
- **index(x)** Restituisce l'indice relativo alla posizione occupata dal primo elemento x della lista.
- **remove(x)** Elimina dalla lista il primo elemento x trovato.
- **sort( )** Ordina la lista in maniera crescente.
- **reverse( )** Ordina la lista in maniera decrescente.
- **count(x)** Restituisce il numero delle volte che l'elemento x compare nella lista

Impariamo ad utilizzare le funzioni che agiscono sulle liste.

```
>>> num = [3,5,1,4,2]
>>> num
[3, 5, 1, 4, 2]
>>> num.sort()       # osserva la notazione “punto”
>>> num
[1, 2, 3, 4, 5]
>>> num.append(7)
>>> num
[1, 2, 3, 4, 5, 7]
>>> num.reverse()
>>> num
[7, 5, 4, 3, 2, 1]
>>>
```

La funzione **del** consente di eliminare elementi di una lista:

```
>>> del num[2]
>>> num
[7, 5, 3, 2, 1]
>>> del num[2:5]
>>> num
[7, 5]
>>>
```

L'esempio che segue utilizza le liste e le funzioni che agiscono su di esse. Copialo attentamente ed eseguilo.

Esercizio 1 sulle liste

```
demolist = [’life’,42, ’the universe’, 6,’and’,7]
print ’demolist = ’,demolist
demolist.append(’everything’)
print "after ’everything’ was appended demolist is now:"
print demolist
print ’len(demolist) =’, len(demolist)
print ’demolist.index(42) =’,demolist.index(42)
print ’demolist[1] =’, demolist[1]
# Il prossimo ciclo analizza la lista.
c = 0
while c < len(demolist):
            print ’demolist[’,c,’]=’,demolist[c]
            c = c + 1
del demolist[2]
print "After ’the universe’ was removed demolist is now:"
print demolist
if ’life’ in demolist:
                      print "’life’ was found in demolist"
else:
     print "’life’ was not found in demolist"
if ’amoeba’ in demolist:
            print "’amoeba’ was found in demolist"
if ’amoeba’ not in demolist:
            print "’amoeba’ was not found in demolist"
demolist.sort()
print ’The sorted demolist is ’,demolist
```

L'output dovrebbe essere questo:

```
demolist = [’life’, 42, ’the universe’, 6, ’and’, 7]
after ’everything’ was appended demolist is now:
[’life’, 42, ’the universe’, 6, ’and’, 7, ’everything’]
len(demolist) = 7
demolist.index(42) = 1
demolist[1] = 42
demolist[ 0 ]= life
demolist[ 1 ]= 42
demolist[ 2 ]= the universe
demolist[ 3 ]= 6
demolist[ 4 ]= and
demolist[ 5 ]= 7
demolist[ 6 ]= everything
After ’the universe’ was removed demolist is now:
[’life’, 42, 6, ’and’, 7, ’everything’]
’life’ was found in demolist
’amoeba’ was not found in demolist
The sorted demolist is [6, 7, 42, ’and’, ’everything’, ’life’]
```

Questo esempio utilizza diverse funzioni. Ad esempio guardate come potete stampare (`print`) un’intera lista per poi aggiungere nuovi elementi con la funzione `append`. La funzione `len` serve a contare quanti elementi sono presenti nella lista. Le entrate (`index`) valide di una lista (cioè quelle richiamabili grazie a `nome_lista [numerodell’elemento]`) vanno da 0 a `len-1`. La funzione `index` serve a sapere la posizione di un elemento in una lista. Nel caso esistano elementi con lo stesso nome ritornerà la prima posizione dell’elemento. Osservate come l’istruzione `demolist.index(42)` ritorni il valore 1 e l’istruzione `demolist[1]` ritorni il valore 42. La linea `# Il prossimo ciclo analizza la lista` è un semplice commento usato per introdurre le prossime righe di codice:

```
c = 0
while c < len(demolist):
            print ’demolist[’,c,’]=’,demolist[c]
            c = c + 1
```

Il codice precedente crea una variabile `c` inizialmente di valore 0 che viene incrementata finché non raggiunge l’ultimo elemento della lista. Nel frattempo l’istruzione `print` stampa ogni elemento della lista.
L’istruzione `del` può essere usata per rimuovere un elemento dalla lista. Le linee successive utilizzano l’operatore `in` per sapere se un elemento è presente o meno nella lista.
La funzione `sort` ordina la lista ed è utile se avete bisogno di una lista ordinata dall’elemento più piccolo a quello più grande, oppure in ordine alfabetico. Notate che questo comporta di elaborare nuovamente la lista.