Esercitazione di Laboratorio 07

Temi trattati

1. Liste di elementi
2. Usare il ciclo `for` per scorrere una lista
3. Funzioni che fanno uso di liste

Discussione

A. Cos'è una lista?
B. Quali sono le differenze tra un elemento di una lista, e il suo indice?
C. Quale informazione serve su una lista per scorrerla senza incorrere in errori?

Esercizi

Parte 1 – Elaborazione di liste

***Consegna**: per ciascuno degli esercizi seguenti, scrivere un programma in Python che risponda alle richieste indicate. Completare almeno due esercizi durante l'esercitazione, e i rimanenti a casa.*

07.1.1 Somma a segni alterni. Scrivere un programma che riceva in input una sequenza di numeri interi (terminata da una riga vuota), e che calcoli la somma alternata dei suoi elementi. Ad esempio, se il programma legge i dati `1 4 9 16 9 7 4 9 11`, deve calcolare e visualizzare `1 - 4 + 9 - 16 + 9 - 7 + 4 - 9 + 11 = -2`. [P6.8]

07.1.2 Lista di numeri casuali. Scrivere un programma che inizializzi una lista con dieci numeri interi casuali tra `1` e `100` e poi visualizzi, su quattro righe successive:

I. Tutti gli elementi di **indice** pari;
II. Tutti gli elementi di **valore** pari;
III. Tutti gli elementi in ordine **inverso**;
IV. Il primo e l'ultimo elemento. [P6.1]

07.1.3 Rimuovere il valore minimo. Scrivere una funzione `remove_min(v)` che rimuova il valore minimo da una lista `v` senza usare la funzione `min()` né il metodo `remove()`. [P6.7]

07.1.4 Massimi locali. Leggere una sequenza di numeri interi conclusa da una riga vuota. Stampare la posizione dei massimi locali (numeri maggiori sia del valore precedente che di quello successivo) se ce ne sono, altrimenti stampare il messaggio `'Non ci sono massimi locali'`.
Estensione: se sono presenti più coppie di massimi locali, individuare i due massimi locali più vicini fra loro e stampare la loro posizione.

07.1.5 Gli stessi elementi. Scrivere la funzione `same_set(a, b)` che verifichi se due liste contengono gli stessi elementi, indipendentemente dall'ordine e ignorando la presenza di duplicati. Ad esempio, le due liste `1 4 9 16 9 7 4 9 11` e `11 11 7 9 16 4 1` devono essere considerate uguali. La funzione non deve modificare le liste che sono state passate come parametri. [P6.12]

07.1.6 Lista ordinata. Scrivere un programma che generi una sequenza di `20` valori interi casuali compresi tra `0` e `99`, poi visualizzi la sequenza generata, la ordini e la visualizzi di nuovo, ordinata. Usate il metodo `sort()`. [P6.17]

07.1.7 Somma senza il minimo. Scrivere la funzione `sum_without_smallest(v)` che calcoli la somma di tutti i valori di una lista `v`, escludendo il valore minimo. [P6.6]

Parte 2 – Algoritmi che fanno uso di liste

***Consegna**: per ciascuno degli esercizi seguenti, scrivere un programma in Python che risponda alle richieste indicate. Completare almeno un esercizio durante l'esercitazione, e i rimanenti a casa.*

07.2.1 Rumore di misura. Spesso i dati raccolti durante un esperimento vanno elaborati per rimuovere parte del rumore di misura. Un approccio semplice a questo problema prevede di sostituire, in una lista di valori, ciascun valore con la media tra il valore stesso e i due valori adiacenti (o dell'unico valore adiacente se il valore in esame si trova a una delle due estremità della lista). Scrivere un programma che svolga tale operazione, senza creare una seconda lista. [P6.36]

07.2.2 Distanziamenti. Le persone che parcheggiano la propria automobile in una fila di parcheggi di solito preferiscono massimizzare la distanza tra il posto che occupano e i posti che sono già occupati da altri veicoli. Tendono quindi ad occupare il posto centrale della fila più lunga di posti liberi a disposizione.

Ad esempio, si consideri la situazione in cui dieci posti sono liberi:

```
_ _ _ _ _ _ _ _ _ _
```

La prima persona che arriva occuperà, col proprio veicolo, un posto nella parte centrale della fila:

```
_ _ _ _ _ X _ _ _ _
```

La persona successiva lo posizionerà a metà della fila lasciata libera più lunga (cioè quella di sinistra):

```
_ _ X _ _ X _ _ _ _
```

Scrivere un programma che riceva in input il numero di posti auto di cui si compone la fila di parcheggi e che, ogni volta che un nuovo posto viene occupato secondo la regola indicata, visualizzi la fila nel formato indicato sopra. *Suggerimento*: utilizzare un elenco di valori booleani per indicare se un posto auto è occupato o meno. [P6.19]

07.2.3 Solitario bulgaro. Il gioco del Solitario bulgaro inizia con `45` carte. Le carte vengono divise a caso in un dato numero di pile con dimensioni casuali. Ad esempio, si può iniziare con `5` pile di dimensioni `20`, `5`, `1`, `9` e `10`. Ad ogni turno, si sottrae una carta da ciascuna pila, e si crea una nuova pila composta dalle carte sottratte dalle pile di partenza. Ad esempio, la configurazione di partenza verrebbe trasformata in pile di dimensioni `19`, `4`, `8`, `9` e `5`. Il gioco finisce quando le pile hanno dimensioni `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8` e `9` (si può dimostrare che si ottiene sempre una configurazione di questo tipo). Scrivere un programma che generi una configurazione iniziale casuale e la visualizzi, e che poi continui con i passi del solitario, visualizzando la configurazione dopo ciascun passo, e fermandosi quando si raggiunge la configurazione finale del solitario. [P6.20]