Esercitazione di Laboratorio 11

Temi trattati

1. Insiemi
2. Dizionari
3. Ricerche in tabelle

Discussione

A. Discutere le somiglianze e le differenze tra:
    a. una lista;
    b. un insieme;
    c. un dizionario.
B. Discutere se un dizionario può avere:
    a. due chiavi con lo stesso valore;
    b. due valori con la stessa chiave.
C. Se Python non fornisse il contenitore `set`, ma ce ne fosse bisogno in un programma, quale tipo di contenitore si potrebbe usare al suo posto?

Esercizi

Parte 1 – Strutture dati complesse

***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.*

11.1.1 Contare le parole. Scrivere un programma che conti le occorrenze di ciascuna parola presente in un file di testo, il cui nome è inserito in input. Si assuma che il file contenga solo caratteri alfabetici o di spaziatura (come ad esempio il file `input.txt`). Il programma deve visualizzare tutte le parole presenti, con a fianco il numero di occorrenze di ciascuna. [P8.2]

11.1.2 Parole più frequenti. Estendere il programma dell'esercizio 11.1.1 in modo che visualizzi solamente le `5` parole più frequenti nel file, senza considerare nel conteggio articoli, preposizioni e congiunzioni. [P8.3]

11.1.3 Due stringhe. Scrivere un programma che chieda all'utente di fornire due stringhe, per poi visualizzare (evitando ripetizioni di caratteri nella stampa):

I. i caratteri che compaiono in entrambe le stringhe;
II. i caratteri che compaiono in una stringa ma non nell'altra;
III. le lettere (alfabetiche) che non compaiono in nessuna delle due stringhe.

Suggerimento: utilizzare la funzione `set()` per trasformare una stringa in un insieme di caratteri. [P8.9]

11.1.4 Censore. Scrivere un programma 'censore', che legga un file (`bad_words.txt`) contenente una lista di 'parolacce' (come `'sesso'`, `'droga'`, `'C++'` e così via), una per riga, inserendole in un insieme. Leggere poi un altro file di testo (`raw_text.txt`), che contenga occorrenze di alcune delle 'parolacce' in questione. Il programma deve riscrivere il secondo file, generandone un terzo (`censored_text.txt`) nel quale il contenuto sia lo stesso, ma con la differenza che tutte le

occorrenze di parole e sotto-parole corrispondenti a 'parolacce' sono sostituite da un numero di asterischi (`'*'`) pari alla loro lunghezza. [P8.14]

11.1.5 Vettori sparsi. Un vettore sparso è una sequenza di numeri, la maggior parte dei quali è `0`. Un modo efficiente per memorizzare un vettore sparso è un dizionario, nel quale le chiavi sono le posizioni in cui sono presenti i soli valori diversi da zero, e i valori sono i corrispondenti valori nella sequenza. Per esempio, la sequenza `0 0 0 0 0 4 0 0 0 2 9` sarebbe rappresentata dal dizionario `{5:4, 9:2, 10:9}`. Scrivere una funzione, `sparse_array_sum(a, b)`, i cui argomenti sono due dizionari di questo tipo, `a` e `b`. La funzione, senza modificare i dizionari passati come argomenti, deve restituire il loro vettore somma come un vettore sparso, dove un valore nella posizione `i` è la somma dei valori di `a` e `b` nelle rispettive posizioni `i`. [P8.22]

11.1.6 Crivello di Eratostene. Il Crivello di Eratostene è un algoritmo iterativo, noto anche nell'Antica Grecia, che calcola tutti i numeri primi minori di un numero intero n . Ad ogni iterazione, l'algoritmo cancella tutti i multipli del valore minimo presente nell'insieme, partendo dal valore minimo 2 e arrivando a considerare come valore minimo $\sqrt{n}$. Alla fine dell'ultima iterazione, i numeri rimasti nell'insieme sono quelli richiesti. Definire una funzione che implementa il Crivello di Eratostene. Per prima cosa, definire un insieme ed inserirvi tutti i numeri interi da `2` a `n`. Poi, eliminare dall'insieme tutti i multipli di `2`, tranne `2` (cioè `4, 6, 8, 10, 12,` e così via, fino a `n`). Come secondo passo, eliminare tutti i multipli di `3`, tranne `3` (cioè `6, 9, 12, 15,` e così via, fino a `n`). Proseguire così, cancellando ogni volta i multipli del valore minimo presente nell'insieme, fino a che i numeri rimasti nell'insieme sono quelli richiesti. [P8.4]

Parte 2 – Operazioni complesse

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

11.2.1 Redditi pro capite. Scrivere un programma che legga i dati dal file di testo `rawdata_2004.txt` e li inserisca in un dizionario le cui chiavi sono nomi di nazioni e i cui valori sono redditi annui pro capite. Si noti che nel file i campi sono separati da un carattere di tabulazione `'\t'`. Poi, il programma deve chiedere all'utente di fornire in input nomi di nazioni, per visualizzare i valori corrispondenti di reddito annuo pro capite. Il programma deve terminare quando l'utente inserisce in input la stringa `'quit'`. È possibile leggere dati analoghi, aggiornati al 2021, in formato `.csv`, dal file `rawdata_2021.csv.` Provare a risolvere lo stesso esercizio lavorando su questo file `.csv`. [P8.17]

11.2.2 Codice genetico. Nelle cellule viventi, l'acido desossiribonucleico (DNA) contiene le informazioni fondamentali perché l'organismo svolga tutte le funzioni utili alla vita. Per supportare le funzioni della cellula, le istruzioni contenute nel DNA vengono prima trascritte in molecole di acido ribonucleico (RNA). Poi, queste molecole di RNA vengono tradotte in proteine. La traduzione da RNA messaggero (mRNA) a proteine si basa sul codice genetico, cioè l'insieme delle regole con le quali una sequenza di nucleotidi (`'A'`, `'C'`, `'U'` e `'G'`) viene tradotta in una sequenza di amminoacidi per la sintesi proteica. Ciascun amminoacido corrisponde a una o più sequenze di tre nucleotidi, che vengono chiamate *codoni*. La tabella seguente[1] riporta i codoni che in un mRNA codificano i `20` amminoacidi ordinari.

1 Codice genetico (Wikipedia): https://it.wikipedia.org/wiki/Codice_genetico

Amminoacido	Codoni	Amminoacido	Codoni
Ala	GCU, GCC, GCA, GCG	Leu	UUA, UUG, CUU, CUC, CUA, CUG
Arg	CGU, CGC, CGA, CGG, AGA, AGG	Lys	AAA, AAG
Asn	AAU, AAC	Met	AUG
Asp	GAU, GAC	Phe	UUU, UUC
Cys	UGU, UGC	Pro	CCU, CCC, CCA, CCG
Gln	CAA, CAG	Ser	UCU, UCC, UCA, UCG, AGU, AGC
Glu	GAA, GAG	Thr	ACU, ACC, ACA, ACG
Gly	GGU, GGC, GGA, GGG	Trp	UGG
His	CAU, CAC	Tyr	UAU, UAC
Ile	AUU, AUC, AUA	Val	GUU, GUC, GUA, GUG

I seguenti codoni indicano le istruzioni di inizio e di fine del processo di traduzione. Si noti che il codone di start codifica anche la Metionina (`'Met'`).

Istruzione	Codoni
start	AUG, GUG
stop	UAG, UGA, UAA

Scrivere un programma che elabori una sequenza di mRNA inserita in input dall'utente, e visualizzi la sequenza di amminoacidi da essa tradotta. Per simulare il processo di traduzione, utilizzare un dizionario `genetic_code`, che, leggendo le informazioni riportate in tabella dal file `codice_genetico.csv`, memorizzi i codoni e gli amminoacidi da essi codificati, scegliendo in modo opportuno quale categoria utilizzare come chiavi e quale come valori. Poi, scorrere la sequenza inserita dall'utente fino a che non si trova una corrispondenza con uno dei codoni di `'start'`. Scorrere infine la parte successiva della sequenza, memorizzando il risultato della traduzione di ciascun codone in una lista `protein`. La traduzione si deve interrompere quando si incontra un codone di `'stop'`. Ad esempio, partendo dalla sequenza di nucleotidi (i codoni di `'start'` e di `'stop'` sono sottolineati), GUAUGCACGUGACUUUCCUCAUGAGCUGAU il programma deve visualizzare: MetHisValThrPheLeuMetSerstop. La prima occorrenza dell'amminoacido `'Met'` (Metionina), il cui codone corrisponde alla sequenza di `'start'` e il punto di `'stop'` sono sottolineati. Se lo scorrimento arriva alla fine della sequenza senza incontrare entrambi i codoni di `'start'` e di `'stop'`, allora il programma deve restituire un messaggio di errore.

11.2.3 Labirinto. Scrivere un programma che legga un file di testo (`maze.txt`) contenente l'immagine di un labirinto, come il seguente, in cui gli asterischi (`'*'`) rappresentano muri invalicabili e gli spazi (`' '`) rappresentano corridoi percorribili.

```
* *******
*   * * *
* ***** *
* * *   *
* * *** *
*   *   *
***** * *
*     * *
******* *
```

Creare un dizionario `corridors` le cui chiavi sono tuple `(riga, colonna)` di posizioni corrispondenti a un corridoio e i cui valori sono insiemi di posizioni anch'esse corrispondenti a un corridoio, e adiacenti alla posizione specificata dalla rispettiva chiave. Nell'esempio sopra, la chiave corrispondente alla tupla `(1, 1)`, evidenziata in blu, ha come posizioni adiacenti `{(1, 2), (0, 1), (2, 1)}`. Visualizzare il dizionario. [P8.20]

11.2.4 Filo di Arianna. Il filo di Arianna deve la sua fama al mito di Minosse e del Labirinto. Esso, infatti, servì a Teseo per trovare l'uscita dal labirinto di Minosse, tracciando ad ogni bivio la direzione verso l'uscita. Estendere il programma dell'esercizio 11.2.3 in modo che, a partire da un punto qualsiasi, trovi una via d'uscita dal labirinto, ad esempio utilizzando il seguente algoritmo:.

> Basandosi sul dizionario `corridors`, costruire un nuovo dizionario `paths` le cui chiavi sono tuple `(riga, colonna)` di posizioni corrispondenti a un corridoio e i cui valori sono inizialmente tutti uguali alla stringa `'?'`. Poi, scorrere le chiavi del dizionario `paths` e, per ogni chiave che rappresenti posizioni ai bordi del labirinto, e che quindi corrisponda ad un'uscita, sostituire la stringa `'?'` con una stringa che indichi la direzione del percorso di uscita: `'N'` (per 'North'), `'E'` (per 'East'), `'S'` (per 'South') o `'W'` (per 'West').
> Fatta questa sostituzione, utilizzando il dizionario `corridors` per localizzare le posizioni adiacenti, scorrere le chiavi del dizionario `paths` il cui valore è ancora `'?'`. Per ciascuna di queste posizioni, verificare se hanno almeno una posizione adiacente il cui valore in `paths` è diverso da `'?'`. In caso affermativo, sostituire la stringa `'?'` con una stringa che indica la direzione verso la quale si trova la posizione adiacente individuata.
> *Esempio*: la chiave corrispondente alla tupla `(1, 1)` ha come posizioni adiacenti `{(1, 2), (0, 1), (2, 1)}`. Se alla chiave `(0, 1)` è stato assegnato il valore `'N'`, nel dizionario `paths`, se all'iterazione corrente il valore assegnato alla chiave `(1, 1)` è ancora `'?'`, poiché nella posizione adiacente c'è un valore diverso da`'?'`, verrà assegnato il valore `'N'`, in quanto la posizione adiacente in questione si trova in direzione 'North'.
> Se durante una delle iterazioni di scorrimento di chiavi e valori non è possibile compiere sostituzioni, concludere l'esecuzione del programma.

Infine, visualizzare il labirinto ottenuto, dove in ciascuna posizione di corridoio è indicata la direzione che porta più rapidamente a un'uscita. Ad esempio:

```
*N*******
*NWW*?*S*
*N*****S*
*N*S*EES*
*N*S***S*
*NWW*EES*
*****N*S*
*EEEEN*S*
*******S*
```

[P8.21]