Esercizio BONUS - "Gestione di una lista della spesa"

Scrivi un programma Python per gestire una lista della spesa. Il programma deve permettere all'utente di:

1) Aggiungere un elemento alla lista.

2) Rimuovere un elemento dalla lista (se presente).

3) Visualizzare tutti gli elementi della lista ordinati in ordine alfabetico.

4) Salvare la lista su file.

5) Caricare una lista da file.

Il programma deve avere un menu che consente all'utente di scegliere le varie operazioni e deve terminare solo quando l'utente lo richiede.

Per lo svoglimento del codice iniziero spiegando una funzione alla volta la prima come anche richiesto è aggiungere elemento alla lista (ho deciso di fare una lista della spesa che comprenda anche le quantità quindi invece di usare una lista usero un dizionario)

Funzione aggiungi_spesa

```python
def aggiungi_spesa():
    articolo = input("Inserire Articolo Da Aggiungere Alla Spesa: ").capitalize().strip()
    while True:
        quantita = input("Inserire quantita dell'articolo: ")
        try:
            quantita = int(quantita)
            if quantita >= 0:
                if articolo in lista_spesa:
                    scelta = input("L'articolo sembra gia presente nella lista della spesa\n"
                                   "Digita 1 Se lo si vuole sovrascrivere\n"
                                   "Digita 2 Se si vogliono sommare le quantità\n"
                                   "Digita 3 per Bloccare l'inserimento\n")
                    if scelta == "2":
                        lista_spesa[articolo] += quantita
                        print(f"è stato aggiunto una quantità di {quantita} all'articolo {articolo} per un totale di {lista_spesa[articolo]}")
                        return
                    elif scelta == "3":
                        print(f"Inserimento bloccato al momento per l'articolo {articolo} la quantità è {lista_spesa[articolo]}")
                        return
                    else:
                        print("Scelta non valida di default si sovrascrivera la quantità dell'articolo")

                lista_spesa[articolo] = quantita
                print(f"é stato aggiunto alla lista l'articolo {articolo} in {quantita} quantità")
                return
            else:
                print("quantità non valida")
        except ValueError:
            print("La quantità inserita non è valida inserire un numero intero positivo")
```

Questa funzione è stata progettata per gestire l'aggiunta di articoli alla lista della spesa, cercando di coprire tutte le possibili casistiche (in linea con i requisiti del compito precedente).
All'inizio della funzione, viene richiesto all'utente di inserire il nome di un articolo. Tramite il metodo capitalize(), il nome viene formattato in modo che solo la prima lettera sia maiuscola, mentre tutte le altre lettere sono minuscole. Il metodo .strip() viene utilizzato per rimuovere eventuali spazi vuoti all'inizio e alla fine della stringa.
Successivamente, viene chiesta la quantità dell'articolo. Grazie al blocco try-except, la funzione verifica che la quantità inserita sia effettivamente un numero intero, e in caso contrario restituisce un messaggio d'errore. Un controllo aggiuntivo con if verifica che il numero sia positivo, poiché una

quantità negativa non sarebbe valida.
Se l'articolo è già presente nella lista (lista_spesa), l'utente può scegliere tra tre opzioni:
L'utente ha la possibilità di scegliere se sovrascrivere la quantità precedente con il nuovo valore, sommare la quantità inserita a quella già presente oppure bloccare l'inserimento, mantenendo invariata la quantità esistente**.**
Nel caso in cui l'utente inserisca una scelta non valida, il programma procede automaticamente con la sovrascrittura della quantità, assumendo questo comportamento come predefinito. Questo è indicato chiaramente nel messaggio all'utente.
La funzione utilizza return in vari punti per interrompere l'esecuzione una volta completata l'operazione scelta. Questo garantisce che il programma rispetti le scelte dell'utente e modifichi la lista della spesa di conseguenza.

Continuando l'ordine della consegna questa è la funzione di rimozione di un elemento dalla lista
Funzione rimozione_spesa:

```
def rimozione_spesa():
    articolo = input("Inserire l'articolo da eliminare: ").capitalize().strip()
    if articolo not in lista_spesa:
        print("Articolo non presente")
    else:
        del lista_spesa[articolo]
        print(f"Eliminato con successo l'articolo {articolo} dalla lista della spesa")
```

Questa funzione è più semplice rispetto alla precedente. Chiede all'utente quale articolo desidera eliminare e, dopo averlo acquisito, normalizza il formato dell'input utilizzando i metodi .capitalize() e .strip() per garantire coerenza con il modo in cui i dati sono stati precedentemente inseriti. Se l'articolo non è presente nella lista, viene mostrato un messaggio a schermo che informa l'utente dell'assenza dell'articolo. Al contrario, se l'articolo è presente, viene rimosso dalla lista e viene visualizzato un messaggio che conferma l'eliminazione avvenuta con successo.

Per una visualizzazione della spesa a mio parere migliore ho deciso di importare la libreria pandas in modo da poter mostrare i dati in un dataframe

```
┌──(kali㉿kali)-[~/Desktop/Python]
└─$ pip install pandas
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in /usr/lib/python3/dist-packages (2.1.4+dfsg)
```

Installato tramite pip e importato sopra il codice as pd come da convenzione

```
import pandas as pd
```

Ora mostro la funzione di visualizzazione
Funzione visualizza_spesa

```
def visualizza_spesa():
    print("Lista Della Spesa:")
    if lista_spesa:

        df = pd.DataFrame(list(lista_spesa.items()), columns=["Articolo", "Quantità"])
        df = df.sort_values(by="Articolo")
        df = df.reset_index(drop=True)
        print(df)
    else:
        print("La Lista è Ancora Vuota")
```

Questa funzione inizia mostrando un semplice messaggio a schermo per indicare che sta stampando la lista della spesa. Successivamente, verifica se la lista della spesa non è vuota. Per farlo, il dizionario lista_spesa viene convertito in una lista di tuple utilizzando list(lista_spesa.items()), dove ogni tupla contiene una coppia chiave-valore. Questi dati vengono poi utilizzati per creare un DataFrame, dove la prima colonna, chiamata "Articolo", contiene le chiavi (nomi degli articoli), mentre la seconda colonna, chiamata "Quantità", contiene i valori (le quantità degli articoli). Il DataFrame viene ordinato alfabeticamente in base alla colonna "Articolo" utilizzando sort_values(by="Articolo"). Successivamente, si utilizza reset_index(drop=True) per reimpostare l'indice del DataFrame, garantendo che i numeri delle righe laterali siano ordinati correttamente. Infine, il DataFrame viene stampato. Se invece la lista della spesa è vuota, la funzione mostra semplicemente un messaggio che informa l'utente che la lista è vuota.

Funzioni salva_spesa

```
def salva_spesa(lista_spesa):
    lista_spesa = {chiave: lista_spesa[chiave] for chiave in sorted(lista_spesa)}
    with open("lista_della_spesa.txt", "w") as file:

        for chiave, valore in lista_spesa.items():
            file.write(f"{chiave}: {valore}\n")
```

Questa funzione richiede come parametro la lista della spesa (lista_spesa) e la salva in un file di testo chiamato lista_della_spesa.txt. Inizialmente, tramite una dictionary comprehension, la lista della spesa viene riorganizzata e ordinata alfabeticamente in base alle chiavi. Questo è ottenuto utilizzando sorted(lista_spesa), che restituisce le chiavi ordinate in ordine alfabetico, e successivamente costruisce un nuovo dizionario mantenendo l'ordine. Successivamente, la funzione utilizza il comando with open("lista_della_spesa.txt", "w") as file: per aprire (o creare, se non esiste) il file lista_della_spesa.txt in modalità scrittura ("w"). L'uso di "w" significa che, se il file esiste già, il suo contenuto verrà completamente sovrascritto. All'interno del blocco with, viene effettuata un'iterazione su lista_spesa.items(), che restituisce le coppie chiave-valore del dizionario. Per ogni coppia, il metodo file.write() viene utilizzato per scrivere una riga nel file, con il formato chiave: valore, seguito da un'interruzione di riga (\n). Infine, quando l'iterazione termina e il blocco with viene chiuso, il file viene automaticamente salvato e chiuso, garantendo l'integrità dei dati.

Funzioni importa_spesa

```
def importa_spesa():
    with open("lista_della_spesa.txt", "r") as file:
        for linea in file:
            chiave, valore = linea.split(":")
            lista_spesa[chiave] = int(valore)

    print("La lista della spesa è stata importata da lista_della_spesa.txt")
    visualizza_spesa()
```

Rispetto alla funzione precedente, il file viene aperto in modalità sola lettura utilizzando il parametro "r" (read). Questo significa che il file lista_della_spesa.txt può essere letto, ma non modificato. La funzione legge il file riga per riga. Per ogni riga, il contenuto viene separato (split) in due parti utilizzando il carattere ":" come delimitatore. La prima parte della riga viene utilizzata come chiave e la seconda come valore, che viene poi convertito in un numero intero tramite int(). Entrambi vengono

aggiunti o aggiornati nel dizionario lista_spesa. Se un articolo è già presente nella lista, la funzione sovrascrive il valore esistente con il nuovo valore importato dal file. Una volta completata l'importazione, viene stampato un messaggio che conferma che la lista della spesa è stata importata correttamente da lista_della_spesa.txt. Infine, viene chiamata la funzione visualizza_spesa per mostrare all'utente la lista della spesa risultante.

La funzione per salvare e importare sono state create pensando che gli unici file che si importano sono quelli creati tramite questo programma infatti il nome del file da importare non è inseribile dall'utente ma è il nome che da la funzione salva_spesa

Con questa sono finite le funzioni che servono per poter integrare le features richieste

```
lista_spesa = {}
print("Programma Per Creare Una lista Della Spesa\n")
print("Digita 1: Per Aggiungere Un Elemento Alla Lista\n"
      "Digita 2: Per Rimuovere Un Elemento Dalla Lista\n"
      "Digita 3: Per Visualizzare La Lista\n"
      "Digita 4: Per Salvare La Lista In Un File txt\n"
      "Digita 5: Per Importare Una Lista Già Creata\n"
      "Digita Qualsiasi Altra Cosa Per Chiudere Il Programma")

while True:
    scelta = input("\nDigita Il Numero Per L'Operazione Che Si Vuole Compiere: ")
    if scelta == "1":
        aggiungi_spesa()
    elif scelta == "2":
        rimozione_spesa()
    elif scelta == "3":
        visualizza_spesa()
    elif scelta== "4":
        salva_spesa(lista_spesa)
    elif scelta == "5":
        importa_spesa()
    else:
        print("Grazie Per Aver Usato Il Programma")
        break
    else:
        scelta = input("Sicuro Di Voler Chiudere Il Programma: Y/N").lower()
        if scelta == "y":
            print("Grazie Per Aver Usato Il Programma")
            break
```

Nella parte finale del programma avviene l'inizializzazione del dizionario lista_spesa, che viene utilizzato per salvare gli articoli e le relative quantità della lista della spesa. All'avvio, all'utente viene mostrata una spiegazione chiara su come utilizzare il programma, inclusi i comandi disponibili. Successivamente, viene avviato un **loop infinito** che consente all'utente di eseguire quante operazioni desidera, come aggiungere articoli, rimuoverli, visualizzare la lista, salvarla o importarla. l'utente inserisce un comando non valido, il programma chiede se si è sicuri di voler chiudere il programma. Se l'utente conferma digitando "y" o "Y" il loop termina e il programma si chiude. Altrimenti, il loop riprende, richiedendo nuovamente all'utente cosa desidera fare.

Allego Il File cosi da poter essere provato