Progetto S2/15
ESERCITAZIONE

CONSEGNA:

Dato il codice si richiede allo studente di:

- Capire cosa fa il programma senza eseguirlo.

- Individuare dal codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati).

- Individuare eventuali errori di sintassi / logici.

- Proporre una soluzione per ognuno di essi.

CODICE: (pagine seguente)

```
import datetime

def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        oggi = datetime.datetoday()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
    elif comando == "Che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
    elif comando == "Come ti chiami?":
        risposta = "Mi chiamo Assistente Virtuale"
    else:
        risposta = "Non ho capito la tua domanda."
    return risposta
while True
    comando_utente = input("Cosa vuoi sapere? ")
    if comando_utente.lower() == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

ANALISI DELLA CONSEGNA:

L’esercizio di oggi prevede l’analisi di una sezione di codice, la quale ha come protagonista una funzione che va a creare una sorta di “assistente virtuale” in grado di fornire la risposta ad alcune domande;

nello specifico i quesiti sono:

1)qual’ è la data di oggi?

2)che ore sono?

3)come ti chiami?

procediamo con ordine a spiegare ed analizzare le varie sezioni della funzione e successivamente, andremo a visionare con attenzione la presenza di casistiche non previste che possono verificarsi.

Inoltre, visioneremo la presenza di eventuali errori e proveremo a correggerli per poi proporre una soluzione per ciascuno di essi

1.

per capire il funzionamento del programma, possiamo frammentarlo in più segmenti.

partiamo intanto da

```
import datetime
```

quì stiamo importando la libreria datetime, che contiene una serie di oggetti e metodi utili ad ottenere informazioni su data e ora odierne, infatti come vedremo ⅔ delle domande che possiamo fare hanno come quesito la data e l’ora attuale.

1.2

```
import datetime

def assistente_virtuale(comando):

    if comando == "Qual è la data di oggi?":

        oggi = datetime.datetoday()

        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

    elif comando == "Che ore sono?":

        ora_attuale = datetime.datetime.now().time()

        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")

    elif comando == "Come ti chiami?":

        risposta = "Mi chiamo Assistente Virtuale"

    else:

        risposta = "Non ho capito la tua domanda."

    return risposta
```

eccoci di fronte alla funzione vera e propria.

la funzione **assistente_virtuale** presenta delle condizioni poste con la struttura "**if, elif, else**" e prende in ingresso un valore "comando" che poi vedremo in seguito al punto 1.6, viene fornito dall'utente.

le condizioni sono tutte impostate sulla veridicità del confronto tra 'comando' e l'input che fornirà il nostro utente, infatti, esempio:

se l'utente chiederà al programma "Che ore sono?"

questo assegnerà alla variabile **‘risposta’** una stringa preimpostata, concatenata al valore assegnato alla variabile ”ora_attuale’ e trasmetterà questa al di fuori della funzione tramite il comando ‘return’, ma vediamo più nello specifico ogni segmento:

1.3

```
if comando == "Qual è la data di oggi?":
    oggi = datetime.datetoday()
    risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
```

se chiediamo al programma ”Qual è la data di oggi?” otterremo la risposta

”La data di oggi è 06/12/2024”

formata dalla stringa da noi creata “La data di oggi è”

concatenata al valore della variabile oggi, ovvero “datetime.datetoday()”

che equivale alla data odierna

oggi.strftime(“%d/%m/%Y) non è altro che un metodo per convertire in stringa il valore assegnato precedentemente alla variabile oggi, in quanto questo viene evidentemente fornito in un formato differente.

1.4

```
elif comando == "Che ore sono?":
    ora_attuale = datetime.datetime.now().time()
    risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
```

nel secondo caso, se lanciamo il programma e chiediamo ”Che ore sono?”

otterremo la risposta

“L’ora attuale è (ora attuale, es 11:11)”

il procedimento dell’assegnazione del valore alla variabile e successivamente la concatenazione di questa(ora_attuale) alla stringa

“L’ora attuale è”

avviene nello stesso modo del punto precedente, all’interno dell’assegnazione alla variabile ‘risposta’

anche qui usiamo il metodo visto nel punto 1.3. per convertire il valore in una stringa, ma cambiamo le lettere nella parentesi perché si tratta di ore-minuti e non di giorni-mesi-anni

1.5

le ultime due condizioni poste fanno si che se l’utente richiede al programma il quesito

“Come ti chiami?”

```
elif comando == "Come ti chiami?":
    risposta = "Mi chiamo Assistente Virtuale"
else:
    risposta = "Non ho capito la tua domanda."
return risposta
```

questo risponderà con una stringa predefinita

“Mi chiamo assistente virtuale”

Se invece dovessimo fornire una domanda diversa da tutte quelle analizzate ora e nei punti precedenti, il programma risponderà con un altra stringa contenente:

“Non ho capito la tua domanda”

grazie alla condizione “else” che prevede qualunque altro valore che non rispetti le condizioni degli ‘if’ precedentemente posti

il ‘return’ ci serve per far sì che la nostra funzione ci restituisca il valore di risposta e ci permetta di esportarlo dallo scope locale della funzione a quello globale, per poterlo poi in seguito stampare.

1.6

```
while True
    comando_utente = input("Cosa vuoi sapere? ")
    if comando_utente.lower() == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

quest’ultima porzione di codice serve a creare un ciclo while infinito che ci permetta di far interagire l’utente con la funzione da noi creata, cardine del programma:

grazie all’assegnazione di un valore ricevuto dall'input dell'utente alla variabile ‘comando_utente’

al quale abbiamo allegato la domanda tramite stringa

”cosa vuoi sapere?’

otteniamo l’input dell’utente che andremo a inserire nella funzione assistente_virtuale

’l’operatore “while” affiancato al valore True permettere di eseguire un ciclo infinito che serve ad interagire con l’utente fino al verificarsi della condizione “comando_utente.lower()== “esci”:

(n.b. è stato usato il metodo .lower() per una migliore user experience dell’utente, così facendo potrà digitare ”esci” con una indifferente commistione di lettere minuscole e maiuscole ma otterrà sempre l’interruzione del ciclo)

viene infatti posta una condizione “if” per allegare un funzionamento al comando “esci”

infatti:

-se l’input fornito dall’utente è uguale a “esci”, rispetteremo la prima condizione e quindi interromperemo il ciclo, stampando prima un cortese “Arrivederci” e poi uscendo dal programma

-se l’input fornito dall utente è un qualsiasi altro valore, otterremo come risultato il valore assegnato alla variabile “risposta”

2

ora che abbiamo compreso il funzionamento del nostro codice possiamo passare alla sua ottimizzazione e individuazione di eventuali casistiche non previste.

2.1

la prima di queste è a mio parere l’uso di maiuscole e minuscole, ovvero, il programma è CaseSensitive, se l’utente inserisce la domanda senza utilizzare gli stessi caratteri maiuscoli e minuscoli che abbiamo utilizzato noi per i valori abbinati a ‘comando’ nelle varie strutture if, non otterrà la risposta desiderata e visualizzerà sempre

“Non ho capito la tua domanda”

2.2

la domanda “cosa vuoi sapere?” è troppo generica perché l’utente sappia che le uniche domande realmente risolvibili sono:

Qual’è la data di oggi?

Come ti chiami?

Che ore sono?

potrebbe ad esempio chiedere “com’è il tempo fuori?” e otterrebbe sempre

”Non ho capito la tua domanda”’

senza aver realmente compreso quali sono le domande possibili da fare al programma.

inoltre non è stato previsto un indirizzamento verso il comando “esci” che per quanto intuitivo, il nostro utente potrebbe non conoscere

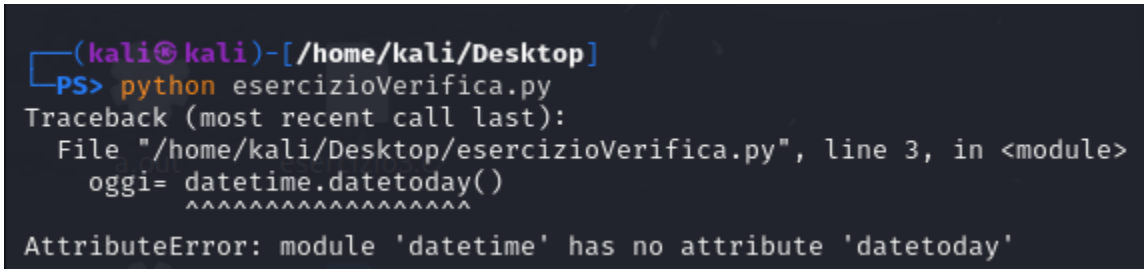
3

ERRORI DI SINTASSI:

per quanto riguarda gli errori all’interno del nostro programma, notiamo un errore di sintassi nella scrittura al punto 1.3

‘datetime.datetoday()’

infatti se proviamo ad ottenere il valore della variabile oggi otterremo quanto segue



bensì, la libreria ci permette di visualizzare la data di oggi, assegnata alla variabile ‘oggi’ con la dicitura:

oggi = datetime.date.today()

così facendo utilizzeremo correttamente il metodo datetime e andremo a prenderci la data odierna da assegnare alla variabile, direttamente dal modulo datetime importato.

ERRORI LOGICI:

a livello di errori ‘LOGICI il programma non presenta criticità nel suo funzionamento una volta corretta la sintassi.

E’ però altresì vero che potremmo considerare errori logici inerenti alla user Experience, le criticità già viste nei punti 2.1,2.2., ovvero la NON predisposizione di alcune casistiche comuni che non vengono trattate nel programma

inoltre, il programma non è commentato, questo fa sì che uno sviluppatore, o un collega, abbia maggior difficoltà a leggerne il contenuto in maniera chiara e intuitiva.

4

Per andare a migliorare il nostro codice seguiremo i seguenti passaggi:

-correggeremo la sintassi come indicato nel punto 3.

-renderemo il codice caseSensitive per permettere agli utenti di inserire i quesiti con caratteri maiuscoli e/o minuscoli

-modificheremo la risposta assegnata alla condizione else “Non ho capito la tua domanda” per indirizzare correttamente l’utente

-modificheremo la domanda posta all’utente inserendo un indicizzazione verso il comando ‘esci’

-commenteremo il codice

4.1

```python
#importiamo la libreria datetime
import datetime

#definiamo la funzione assistente_virtuale
def assistente_virtuale(comando):
    if comando.lower() == "qual'è la data di oggi?":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
    elif comando.lower() == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
    elif comando.lower() == "come ti chiami?":
        risposta = "Mi chiamo Assistente Virtuale"
    else:
        risposta = "Non ho capito la tua domanda. Puoi chiedermi:\n- Qual'è la data di oggi?\n- Che ore sono?\n- Come ti chiami?\n"

    return risposta

# Ciclo while per prendere l'input dall'utente ed eseguire la funzione fino a break
while True:
    comando_utente = input("Cosa vuoi sapere?\n digita 'esci' per uscire dal programma ")
    if comando_utente.lower() == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

il codice ora si presenta come segue, ma vediamo nel dettaglio le migliorie apportate:

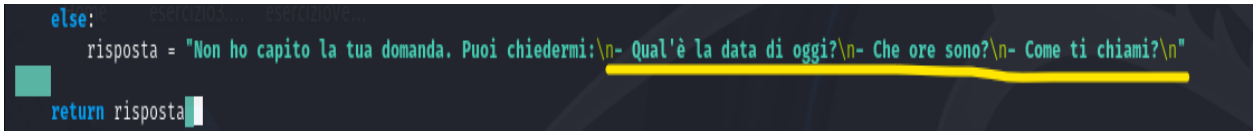
1)abbiamo corretto l'errore di sintassi

```python
#importiamo la libreria datetime
import datetime

#definiamo la funzione assistente_virtuale
def assistente_virtuale(comando):
    if comando.lower() == "qual'è la data di oggi?":
        oggi = datetime.date.today()
```

2)abbiamo reso il codice caseSensitive in modo da poter ricevere input con caratteri minuscoli e/ o maiuscoli

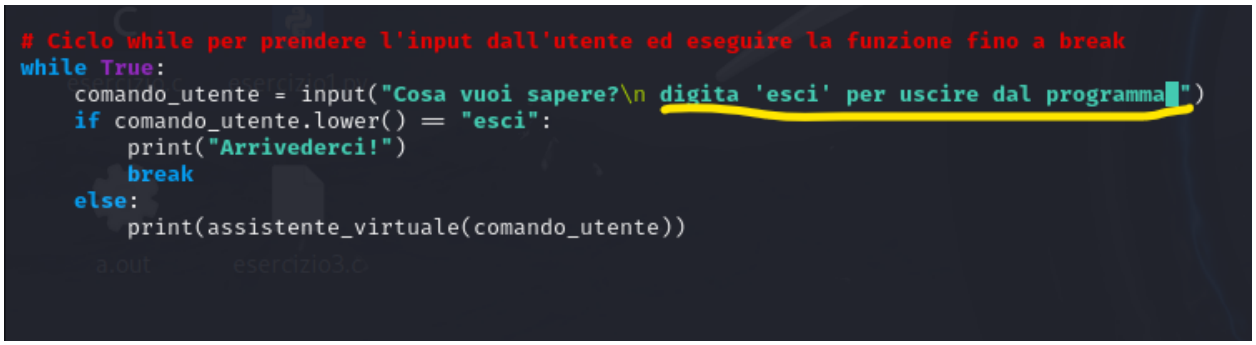
```python
def assistente_virtuale(comando):
    if comando.lower() == "qual'è la data di oggi?":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/
    elif comando.lower() == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%
    elif comando.lower() == "come ti chiami?":
        risposta = "Mi chiamo Assistente Virtuale"
    else:
        risposta = "Non ho capito la tua domanda. Puoi chiederm

    return risposta
```

3) abbiamo aggiunto alla risposta nel caso di ‘else’, ovvero quando l’input non corrisponde a nessun altra condizione, una specifica che segnala le domande da porre al programma



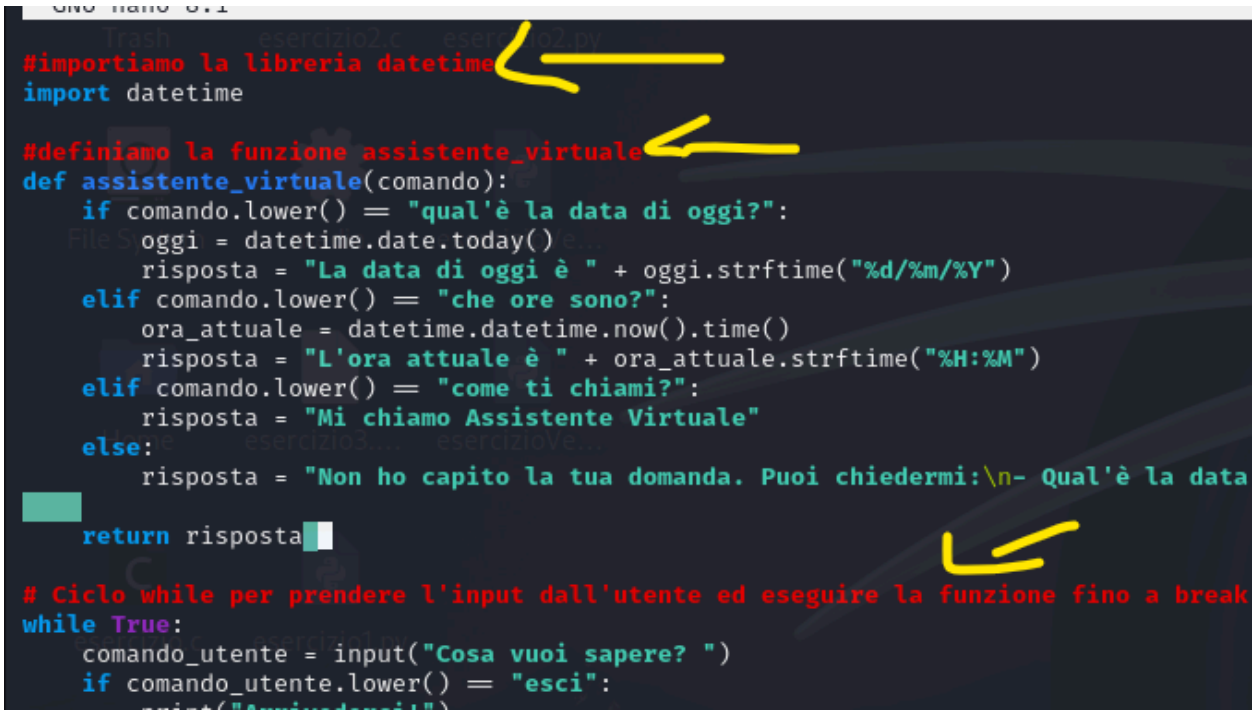
```
    else:
        risposta = "Non ho capito la tua domanda. Puoi chiedermi:\n- Qual'è la data di oggi?\n- Che ore sono?\n- Come ti chiami?\n"

    return risposta
```

e abbiamo modificato la domanda come segue



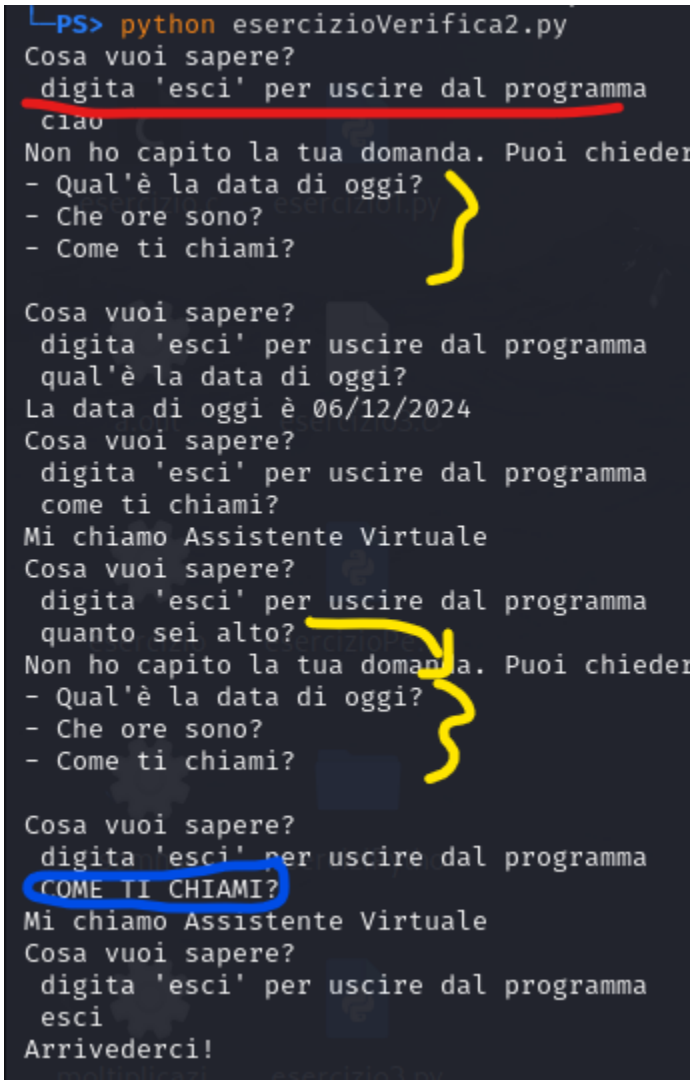
```
# Ciclo while per prendere l'input dall'utente ed eseguire la funzione fino a break
while True:
    comando_utente = input("Cosa vuoi sapere?\n digita 'esci' per uscire dal programma ")
    if comando_utente.lower() == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

4) abbiamo commentato il codice



```
#importiamo la libreria datetime
import datetime

#definiamo la funzione assistente_virtuale
def assistente_virtuale(comando):
    if comando.lower() == "qual'è la data di oggi?":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
    elif comando.lower() == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
    elif comando.lower() == "come ti chiami?":
        risposta = "Mi chiamo Assistente Virtuale"
    else:
        risposta = "Non ho capito la tua domanda. Puoi chiedermi:\n- Qual'è la data

    return risposta

# Ciclo while per prendere l'input dall'utente ed eseguire la funzione fino a break
while True:
    comando_utente = input("Cosa vuoi sapere? ")
    if comando_utente.lower() == "esci":
```

Testiamo ora il funzionamento del nostro nuovo codice che fornisce una User Experience di livello superiore:



grazie all'inserimento di "digita esci per uscire dal programma" il nostro utente è a conoscenza del comando 'esci'.

come possiamo notare dalle ultime domande, ora il programma è in grado di ricevere degli input e convertirli in minuscolo prima di fornire la risposta, questo fa sì che l'utente non riscontri problemi nel dover utilizzare esattamente i caratteri maiuscoli e minuscoli preimpostati da noi all'interno dalla funzione e abbia maggior facilità nel porre le domande

vediamo anche che alla domanda ”quanto sei alto?” ora il programma ci indirizzerà verso una delle domande che abbiamo impostato, fornendo anche quì un'esperienza più user Friendly

inoltre con il codice commentato chiunque vi ”metta mano” riscontrerà una maggior facilità nel comprendere il funzionamento dello stesso.

CONCLUSIONI

Abbiamo analizzato il codice contenente la funzione **assistente_virtuale**,

riscontrato eventuali errori logici e di sintassi e abbiamo proceduto ad effettuare delle migliorie per rendere l’esperienza più adatta ad un comune uso dell’utente, ovvero più User Friendly.

questa caratteristica ha una notevole importanza nello sviluppo dei programmi, tanto che alcune grandi aziende pur avendo dispositivi meno avanzati dal punto di vista delle potenzialità dell’HW, riescono a catturare l’attenzione del cliente proprio fornendo loro SW di proprietà che sono banalmente più semplici o più intuitivi da utilizzare, ne vediamo un esempio abbastanza classico con Apple, che spesso viene criticata per la scarsa qualità dell’HW ma viene comunque preferita dalla maggior parte degli utenti per l’user experience che fornisce attraverso software di loro proprietà, orientati alla semplicità d’utilizzo.

avremmo altresì potuto riscrivere da capo il codice utilizzando un’altra struttura come un ciclo while o strutture ancor più complesse, ma per una maggior leggibilità, e trattandosi di un codice molto semplice, questa è , e rimane la struttura più chiara e leggibile possibile (una volta inseriti i dovuti commenti)

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.

ANALISI

per adempiere alla consegna dell esercizio, procederemo creando due funzioni, una perr mostrare all’utente il menu dell opzioni, e una con struttura ‘if,if else, else’, chiamata SPESA nella quale lanceremo la prima funzione che chiameremo MENU e successivasmente svolgeremo le operazioni richieste dall’esercizio per ottemperare alle richieste dell utente.

1.

creiamo la nostra funzione MENU come segue, inserendo i vari punti delle opzioni fornite all utente

```python
# funzione per mostrare il menu delle opzioni
def menu():
    print("\nLista della spesa")
    print("1. aggiungere un elemento")
    print("2. rimuovere un elemento")
    print("3. visualizzare la lista")
    print("4. salvare la lista su file")
    print("5. caricare la lista da file")
    print("6. uscire")
```

Come visto nell'esercizio prima, lavoriamo scomponendo la consegna in più parti, andiamo ora a visualizzare ciascuna delle richieste dell esercizio:

2. 1

```python
# funzione spesa per eseguire le opzioni mostrate in def menu
def spesa():
    spesa = []  # array vuoto nel quale inseriremo gli elementi della spesa

    while True:
        menu()
        scelta = input("\nCiao, scegli una delle precedenti opzioni: ")

        if scelta == "1":  # condizione per aggiungere un elemento
            elemento = input("Inserisci un elemento da aggiungere alla spesa: ")
            spesa.append(elemento)
```

abbiamo creato la funzione ‘spesa’ e successivamente abbiamo richiamato dentro di essa la funzione MENU in modo da rilanciare il nostro menu ad ogni ripetizione del ciclo while all'interno di SPESA

creiamo inoltre un array vuoto, che chiameremo ‘spesa’ (da ora in poi verrà distinto dalla funzione per la nomenclatura in minuscolo all’interno della nostra relazione)

nel quale andremo a lavorare per inserire e togliere gli elementi della spesa.

Poniamo quindi la condizione ‘while true’ per creare il nostro ciclo infinito ed inseriamo un input che assegni alla variabile ‘scelta’ un valore asserito dall’utente (un numero da 1 a 6)

poniamo ora la condizione ‘if’ per adempiere alla richiesta di aggiungere un elemento qualora l’utente desideri farlo e usiamo un altro input per far assegnare all’utente un valore alla nuova variabile ”elemento”

con il metodo '.append' andiamo ad inserire 'elemento' all'interno del nostro array 'spesa'

2.2

```
elif scelta == "2":  # condizione per rimuovere un elemento
    elemento = input("rimuovi un elemento: ")
    if elemento in spesa:
        spesa.remove(elemento)
```

nel secondo frammento di codice utilizziamo lo stesso ragionamento che abbiamo utilizzato nel punto 2.1, rimaniamo quindi all'interno della struttura 'if' ma poniamo la condizione 'elif scelta ==2', in modo da restare coerenti con l'indicizzazione del MENU.

per rimuovere l'elemento di nuovo salviamo l'input dell'utente dentro la variabile e procediamo questa volta a usare il metodo '.remove' per rimuovere l'elemento selezionato dall utente

2.3

```
elif scelta == "3":  # condizione per visualizzare la lista

    print("\nLa tua lista:")
    for i in sorted(spesa):
        print(i)
```

questa volta l'obiettivo è visualizzare gli elementi all'interno dell'array spesa in ordine alfabetico

. per farlo utilizzeremo un ciclo for, che per ogni elemento presente, stamperà l'elemento stesso in modo da visualizzarli tutti.

N.B usiamo sorted(spesa) per sfruttare la funzione di python che ci permette di ordinare gli elementi dell'array in ordine alfabetico

2.4

```
        elif scelta == "4":  # condizione per salvare la lista su file
            nomeFile = input("Inserisci il nome del file: ")
            with open(nomeFile, "w") as file:
                for i in spesa:
                    file.write(i)
```

per salvare la lista su file ci serviremo di 'with', che ci permette appunto di aprire un file in diverse modalità.

a noi interessa salvare un file di tipo testuale e perciò utilizzeremo

'with open(nomeFile, "w") as file'

in modo da aprire il file specificato dall'utente in modalità scrittura (writing), successivamente cicliamo tramite 'for' gli elementi del nostro array 'spesa' e utilizziamo

'write' per scrivere nel file ogni elemento appartenente all'array

2.6

```
        elif scelta == "6":  # condizione per uscire
            print("Arrivederci e grazie")
            break
        else:
            print("La tua scelta non è valida, riprova con uno dei numeri assegnati!")

# lanciamo il nostro programma
spesa()
```

per poter permettere all'utente di uscire inseriamo un 'break' che interrompa il nostro ciclo

definitivamente al verificarsi della condizione 'scelta == 6'

infine nell'ultima riga, ovviamente, dobbiamo lanciare il nostro programma.

3

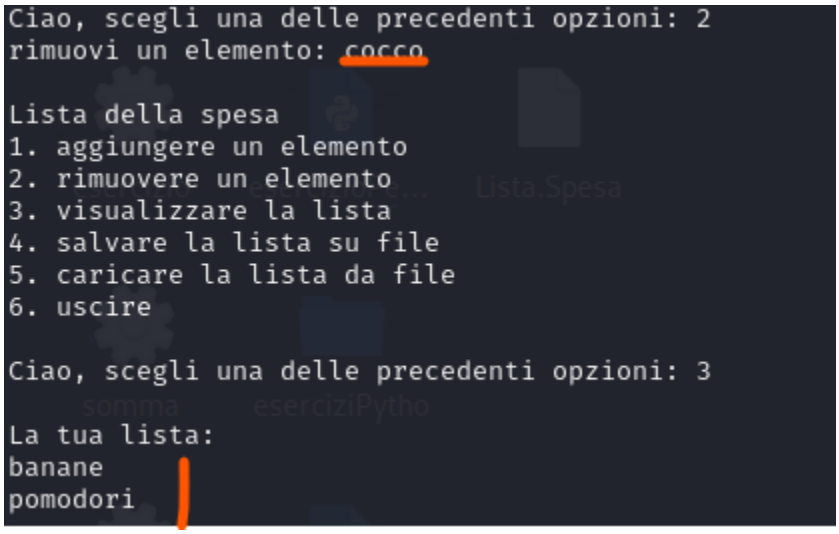
verifichiamo ora il funzionamento del nostro prgramma dal terminale

```
Ciao, scegli una delle precedenti opzioni: 1
Aggiungi un elemento: banane

Lista della spesa
1. aggiungere un elemento
2. rimuovere un elemento
3. visualizzare la lista
4. salvare la lista su file
5. caricare la lista da file
6. uscire

Ciao, scegli una delle precedenti opzioni: 1
Aggiungi un elemento: cocco

Lista della spesa
1. aggiungere un elemento
2. rimuovere un elemento
3. visualizzare la lista
4. salvare la lista su file
5. caricare la lista da file
6. uscire

Ciao, scegli una delle precedenti opzioni: 1
Aggiungi un elemento: pomodori

Lista della spesa
1. aggiungere un elemento
2. rimuovere un elemento
3. visualizzare la lista
4. salvare la lista su file
5. caricare la lista da file
6. uscire
```

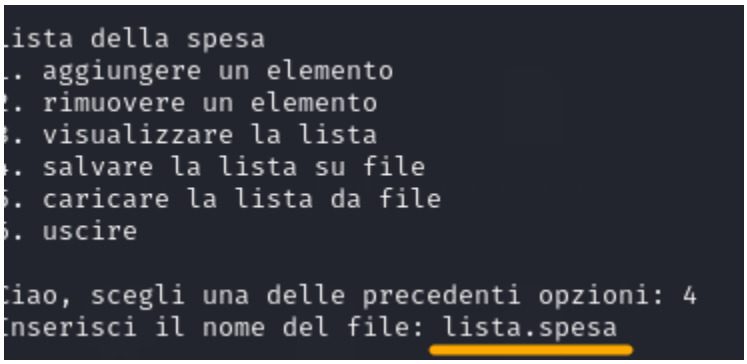
1)possiamo vedere come aggiungendo gli elementi non vengano visualizzati errori, ma procediamo col verificare se la lista della spesa è stata davvero aggiornata come vorremmo col comando '3'

```
Ciao, scegli una delle precedenti opzioni: 3

La tua lista:
banane
cocco
pomodori

Lista della spesa
1. aggiungere un elemento
2. rimuovere un elemento
3. visualizzare la lista
4. salvare la lista su file
5. caricare la lista da file
6. uscire
```

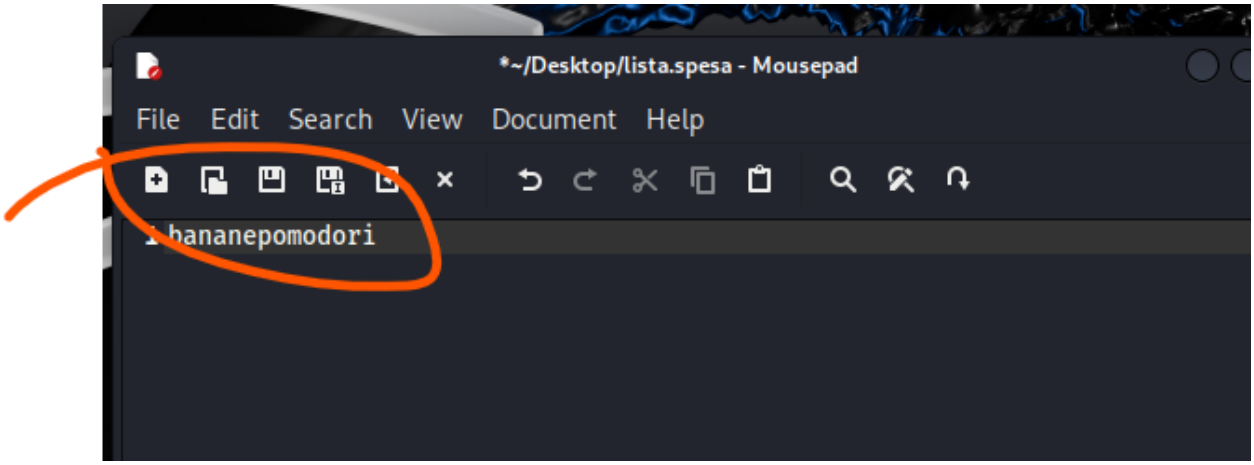
2)ecco la nostra lista della spesa, procediamo a rimuovere un elemento col comando ‘2’



3) ottimo, ora procediamo a salvare la lista della spesa su un file. non c’è bisogno di creare prima il file perchè se questo non esiste, python ne creerà uno col nome fornitogli



4)verifichiamo l’esistenza del file banalmente aprendolo dal desktop



per uscire lanciamo utilizziamo il comando assegnatovi, ovvero ‘6’

```
Lista della spesa
1. aggiungere un elemento
2. rimuovere un elemento
3. visualizzare la lista
4. salvare la lista su file
5. caricare la lista da file
6. uscire

Ciao, scegli una delle precedenti opzioni: 6
Arrivederci e grazie
```

Ecco infine il nostro codice completo

```python
def menu():
    print("\nLista della spesa")
    print("1. aggiungere un elemento")
    print("2. rimuovere un elemento")
    print("3. visualizzare la lista")
    print("4. salvare la lista su file")
    print("5. caricare la lista da file")
    print("6. uscire")

# funzione spesa per eseguire le opzioni mostrate in def menu
def spesa():
    spesa = []  # array vuoto nel quale inseriremo gli elementi della spesa

    while True:
        menu()
        scelta = input("\nCiao, scegli una delle precedenti opzioni: ")

        if scelta == "1":  # condizione per aggiungere un elemento
            elemento = input("Aggiungi un elemento: ")
            spesa.append(elemento)

        elif scelta == "2":  # condizione per rimuovere un elemento
            elemento = input("rimuovi un elemento: ")
            if elemento in spesa:
                spesa.remove(elemento)

        elif scelta == "3":  # condizione per visualizzare la lista

                print("\nLa tua lista:")
                for i in sorted(spesa):
                    print(i)

        elif scelta == "4":  # condizione per salvare la lista su file
            nomeFile = input("Inserisci il nome del file: ")
            with open(nomeFile, "w") as file:
                for i in spesa:
                    file.write(i)


        elif scelta == "6":  # condizione per uscire
            print("Arrivederci e grazie")
            break

        else:
            print("La tua scelta non è valida, riprova con uno dei numeri assegnati!")

# lanciamo il nostro programma
spesa()
```

come si nota, non sono riuscito ad adempiere alla consegna 5 perchè evidentemente sbaglio qualcosa nella realizzazione e avrei bisogno di più tempo per identificare e risolvere l'errore. Mi scuso in anticipo per la mancanza.

Daniele Balani