Progetto S2L5

Gianluca Barella

SCOPO: Dato il programma in Python presente nella prossima figura, capirne il funzionamento, individuare eventuali errori e correggerli migliorando il programma

```
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))
```

1) Funzionamento

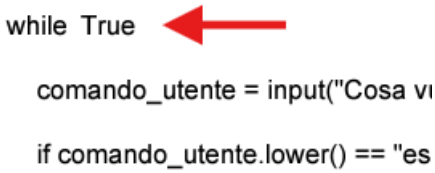
Ad una prima analisi il programma sembra emulare un assistente virtuale molto semplice che risponde ad alcune domande fatte dall'utente eventualmente inviando un messaggio di non comprensione se la domanda posta non è tra quelle preimpostate. Il programma è composto da un ciclo che svolge il lavoro del main e da una funzione che viene richiamata dentro di esso, da notare il "while true" quindi l'assistente continuerà a chiedere domande finché non sarà l'utente a terminare il ciclo inviando l'opportuno comando. La funzione riceve come parametro l'input dell'utente per poi dividersi in una struttura "if-elif" per poter stampare l'opportuna risposta in base alla domanda chiesta dall'utente.
Il programma inizia con una prima interazione da parte dell'assistente che chiede all'utente cosa vuole sapere, l'utente può decidere se interagire con una domanda oppure scegliere di terminare l'esecuzione. Se sceglie di chiedere una delle domande preimpostate avrà come risposta data, nome dell'assistente oppure un messaggio di non comprensione della domanda e questa cosa continuerà a ripetersi finché l'utente non invia la stringa preimpostata "esci" per far terminare l'esecuzione.

2) Casistiche non gestite

Questo programma si aspetta un perfetto inserimento della domanda da parte dell'utente quindi una casistica che non è stata gestita (o può essere gestita meglio) è quella nel caso l'utente sbagli a digitare, in questo caso si potrebbe provare a capire la domanda e quindi chiedere un chiarimento tipo "È questo quello che intendevi?" per poi proseguire con la domanda che il programma ha supposto.
Altre due situazioni simili potrebbero essere la dimenticanza del punto interrogativo o le lettere maiuscole/minuscole non messe correttamente.

3) Errori

Il primo errore riscontrato è la mancanza dei due punti dopo la condizione del ciclo "while"



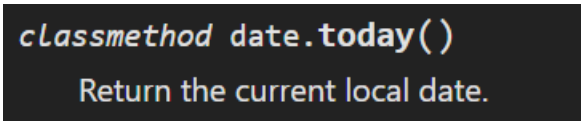
Cercando in internet il modulo "datetime" un altro errore che si può notare è l'assenza del punto nella riga rappresentata in figura:

4) Correzione

Per il problema degli errori di battitura/ortografia, dato che le domande preimpostate iniziano con parole diverse, un piccolo trucco che si potrebbe fare è, nel caso non venga riconosciuta la domanda si può basarsi esclusivamente sulla prima parola facendo un split() dell'input dell'utente per poi selezionare il caso scelto, questo ridurrebbe la possibilità di errore ad una sola parola rispetto ad una frase intera riducendo appunto la possibilità che ciò avvenga. Un problema di questa soluzione però è quella che se l'utente inserisce qualcosa del tipo "Che data è oggi?" avrebbe come risposta l'ora. Una soluzione che può essere migliore è quella di andare alla ricerca di parole chiave tipo "data", "ora", "nome" nell'input dell'utente per poi assegnare la risposta corretta, anche in questo caso si va a filtrare eventuali non corrispondenze al 100% rispetto alle domande preimpostate ma in in più

si corregge il problema della soluzione precedente. Per eseguire questa soluzione si userà sempre split() per ottenere una lista di stringhe per poi andare a comparare ogni parola con un ciclo “for” con le parole chiave e selezionare il caso scelto.
Se si vuole invece andare a gestire solo l’assenza del punto interrogativo basterà fare un OR come condizione dell’if-elif nei vari casi con la domanda con o senza punto interrogativo.
Come gestione delle maiuscole/minuscole basterà portare tutto l’input in lowercase/uppercase e lavorare con quello come fatto all’interno del while.

Nel caso degli errori basterà aggiungere la punteggiatura mancante cioè i due punti dopo la condizione del while e il punto tra “date” e “today”.
Le sintassi corrette sono rappresentate in figura:



Un esempio di programma con alcune delle soluzioni proposte è il seguente:

```
import datetime

def assistente_virtuale(comandoX):
# Rendo l'imput tutto minuscolo
        comando = comandoX.lower()
# Faccio i vari casi per poter capire la risposta da dare
        if (comando == "qual è la data di oggi?") or (comando == "qual"):
                oggi = datetime.date.today()
                risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
        elif (comando == "che ore sono?") or (comando == "che"):
                ora_attuale = datetime.datetime.now().time()
                risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
        elif (comando == "come ti chiami?") or (comando == "come") :
                risposta = "Mi chiamo Assistente Virtuale"
        else:
                risposta = "Non ho capito la tua domanda."
        return risposta

# Eseguo il ciclo per poter continuare a chiedere domande
while True:
        comando_utente = input("Cosa vuoi sapere? ")
        if comando_utente.lower() == "esci" :
                print("Arrivederci")
                break
        else:
                print(assistente_virtuale(comando_utente))
```

ESERCIZIO BONUS

SCOPO: scrivere un programma in Python che gestisce una lista della spesa e che permette di eseguire varie operazioni sull'elenco, caricare da file e che abbia un menù per far scegliere all'utente cosa fare.

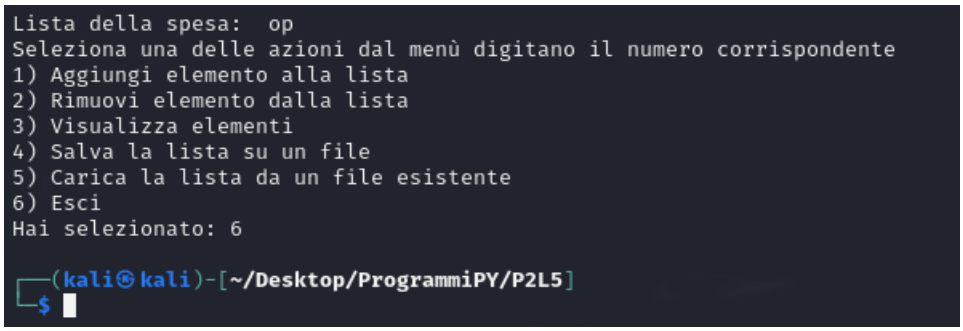
Per scrivere questo programma si è deciso di partire dalla creazione del nome della lista della spesa che dovrà inserire l'utente per proseguire con il programma.
Ora si crea una ciclo main, sarà un "while True:" che creerà e stamperà il menù con le varie opzioni che l'utente può scegliere e in base a ciò che sceglie ci sarà una struttura "if-elif" che gestisce la richiesta e trovato il caso selezionato eseguirà il comando, altrimenti darà un messaggio di errore e farà riprovare l'utente.
Per rendere il programma più leggibile è stato scelto di scrivere le varie funzioni e poi richiamarle all'interno dell'if.

Di seguito sono riportate delle figure sul funzionamento del programma:

```
L'elemento patate è stato aggiunto alla lista op


Lista della spesa:  op
Seleziona una delle azioni dal menù digitano il numero corrispondente
1) Aggiungi elemento alla lista
2) Rimuovi elemento dalla lista
3) Visualizza elementi
4) Salva la lista su un file
5) Carica la lista da un file esistente
6) Esci
Hai selezionato: 1

Inserisci l'elemento da aggiungere: carote
L'elemento carote è stato aggiunto alla lista op


Lista della spesa:  op
Seleziona una delle azioni dal menù digitano il numero corrispondente
1) Aggiungi elemento alla lista
2) Rimuovi elemento dalla lista
3) Visualizza elementi
4) Salva la lista su un file
5) Carica la lista da un file esistente
6) Esci
Hai selezionato: 1

Inserisci l'elemento da aggiungere: carne
L'elemento carne è stato aggiunto alla lista op


Lista della spesa:  op
Seleziona una delle azioni dal menù digitano il numero corrispondente
1) Aggiungi elemento alla lista
2) Rimuovi elemento dalla lista
3) Visualizza elementi
4) Salva la lista su un file
5) Carica la lista da un file esistente
6) Esci
Hai selezionato: 3
La lista opordinata in ordine alfabetico è:
carne
carote
patate
```

Nella figura precedente sono stati aggiunti tre elementi e sono stati messi in ordine alfabetico.
Nella figura successiva si provvederà a rimuovere un elemento, prima non presente e successivamente presente:

Ora verrà mostrata l'uscita dal programma in caso l'utente prema 6



Come ultima esecuzione verrà mostrato il salvataggio su file e il successivo caricamento, la lista della spesa era composta da patate e carote:

```
Lista della spesa:  lista
Seleziona una delle azioni dal menù digitano il numero corrispondente
1) Aggiungi elemento alla lista
2) Rimuovi elemento dalla lista
3) Visualizza elementi
4) Salva la lista su un file
5) Carica la lista da un file esistente
6) Esci
Hai selezionato: 4
La lista è stata salvata in un file di testo Lista_della_spesa.txt
```

Questa è un’esecuzione diversa dove è stata importata la lista precedente

```
Lista della spesa:  lista
Seleziona una delle azioni dal menù digitano il numero corrispondente
1) Aggiungi elemento alla lista
2) Rimuovi elemento dalla lista
3) Visualizza elementi
4) Salva la lista su un file
5) Carica la lista da un file esistente
6) Esci
Hai selezionato: 5
Inserisci il nome del file di testo da cui vuoi caricare la lista: Lista_della_spesa.txt
La lista caricata é:
 carote
 patate


Lista della spesa:  lista
Seleziona una delle azioni dal menù digitano il numero corrispondente
1) Aggiungi elemento alla lista
2) Rimuovi elemento dalla lista
3) Visualizza elementi
4) Salva la lista su un file
5) Carica la lista da un file esistente
6) Esci
Hai selezionato: 3
La lista listaordinata in ordine alfabetico è:
 carote
 patate
```