

Progetto S2- L5

Analisi del Codice – **Assistente Virtuale**

1. Cosa vuole fare questo programma (senza eseguirlo):

Questo programma vuole essere un piccolo **assistente virtuale** che lavora nel terminale. L'idea è questa:

- Il programma fa una domanda all'utente: «Cosa vuoi sapere?»
- L'utente scrive una frase (un comando).
- Se l'utente scrive «esci», il programma deve salutare («Arrivederci!») e terminare.
- Se l'utente scrive una domanda diversa, il testo viene passato alla funzione `assistente_virtuale()`.
- La funzione legge la frase e decide quale risposta dare (data, ora, nome, oppure «Non ho capito...»).

In particolare, il programma dovrebbe riconoscere tre domande precise:

- Qual è la data di oggi? --- deve rispondere con la data di oggi.
- «Che ore sono? --- deve rispondere con l'ora attuale.
- «Come ti chiami? --- deve rispondere «Mi chiamo Assistente Virtuale.».
- Se la frase non è una di queste, la risposta deve essere «Non ho capito la tua domanda.»

2. Situazioni particolari (non standard) che il programma originale non gestisce bene:

Oltre agli errori veri e propri, il codice così com'è non è molto robusto. Ecco alcune situazioni reali che possono creare problemi:

- L'utente può scrivere le frasi in modo leggermente diverso (tutto minuscolo, tutto maiuscolo, senza punto interrogativo).

- L'utente può inserire spazi all'inizio o alla fine della frase.
- Il comando «esci» viene riconosciuto solo se scritto esattamente così, minuscolo e senza spazi.
- Se l'utente preme solo INVIO (stringa vuota), il programma non lo gestisce in modo particolare.
- Nel messaggio «Cosa vuoi sapere?» non è spiegato come uscire dal programma (non dice che si deve scrivere «esci»).

3. Errori di sintassi e di logica con numero di riga e correzione

- [Linea 3] **while True** → manca il «:». In Python tutti i cicli e gli if devono terminare con i due punti.
Corretto: **while True:**.
- [Linea 7] **break** è allo stesso livello del while. Così il ciclo si interrompe sempre dopo la prima volta, anche quando l'utente non ha scritto «esci». Il break deve stare **dentro** l'if.
- [Linea 8-9] L'**else** è agganciato al while (else del while), non all'if. In questo modo la funzione `assistente_virtuale()` non viene usata correttamente.
- La funzione `assistente_virtuale()` è definita **dopo** il ciclo while che prova a chiamarla. Se il codice venisse eseguito, Python solleverebbe un `NameError`.
- [Linea 13] Il metodo corretto, nel contesto di questo programma, è `datetime.date.today()`, che restituisce solo la data (anno, mese, giorno). `datetime.datetime.today()` è un altro metodo del modulo, ma serve a ottenere data + ora e non è necessario qui.

4. Versione corretta e migliorata del codice:

```
# Import del modulo necessario
import datetime

# Definizione della funzione assistente_virtuale(comando)
# Questa funzione riceve la frase scritta dall'utente (comando)
# e restituisce una risposta (stringa) da mostrare a schermo.
def assistente_virtuale(comando):
```

```

# Normalizzazione dell'input:
# strip() rimuove spazi prima e dopo
# lower() converte tutto in minuscolo
comando_norm = comando.strip().lower()

# Domanda sulla data
if comando_norm in ("qual è la data di oggi?", "che giorno è oggi?"):
    oggi = datetime.date.today()
    risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

# Domanda sull'ora
elif comando_norm in ("che ore sono?", "che ora è?"):
    ora_attuale = datetime.datetime.now().time()
    risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")

# Domanda sul nome dell'assistente
elif comando_norm == "come ti chiami?":
    risposta = "Mi chiamo Assistente Virtuale."

# Domanda non riconosciuta
else:
    risposta = "Non ho capito la tua domanda."

return risposta

# Blocco principale del programma
# Viene eseguito solo se lanciamo questo file direttamente.
if __name__ == "__main__":

    # Ciclo che continua finché l'utente non scrive "esci"
    while True:

        # Chiediamo all'utente cosa vuole sapere
        comando_utente = input("Cosa vuoi sapere? (scrivi 'esci' per uscire) ")

        # Controlliamo se l'utente vuole uscire
        if comando_utente.strip().lower() == "esci":
            print("Arrivederci!")
            break

        else:
            # Passiamo il comando alla funzione per ottenere la risposta
            risposta = assistente_virtuale(comando_utente)
            print(risposta)

```

- Miglioramenti possibili:

Oltre agli errori corretti, il programma può essere migliorato in diversi modi per renderlo più completo e più vicino a un assistente reale. Alcuni possibili miglioramenti sono:

- **Aggiungere più modi per chiedere la stessa cosa**
L'utente potrebbe scrivere le domande in tanti modi diversi. Espandere le varianti renderebbe l'assistente più naturale e meno rigido.
- **Gestire meglio gli input vuoti o incompleti**
Se l'utente preme INVIO senza scrivere nulla, sarebbe meglio mostrare un messaggio dedicato invece della risposta generica.
- **Organizzare il codice in modo ancora più chiaro**
Si potrebbero dividere alcune parti della logica in piccole funzioni, così da rendere il codice più leggibile e più facile da modificare.
- **Aggiungere un comando di "aiuto"**
Scrivendo "aiuto", l'utente potrebbe vedere un elenco delle domande che l'assistente è in grado di capire.
- **Supportare più lingue**
Si potrebbe aggiungere la possibilità di far rispondere l'assistente in lingua italiana o inglese, in base a una scelta dell'utente.
- **Registrazione degli input per scopi didattici**
Un semplice sistema di log permetterebbe di salvare tutte le domande fatte dall'utente, utile per capire come migliorare la logica e per esercitarsi nell'analisi del comportamento.

- Conclusione

In sintesi, questo programma funziona come un piccolo assistente virtuale da terminale. L'utente può scrivere una domanda e il programma, grazie alla funzione **assistente_virtuale()**, decide quale risposta dare in base al contenuto del messaggio. Prima di analizzarlo, il testo inserito dall'utente viene ripulito e trasformato in minuscolo per evitare errori dovuti a spazi o maiuscole.

Il programma riconosce tre richieste principali: la data di oggi, l'ora attuale e il nome dell'assistente. Per qualsiasi altra frase, restituisce un messaggio che informa che la domanda non è stata capita.

Il ciclo principale continua a fare domande finché l'utente non scrive "esci", permettendo così di chiudere il programma in modo chiaro e controllato.

In conclusione, il flusso è semplice:

1. Il programma chiede una domanda.
2. L'utente risponde.
3. Il programma interpreta la richiesta e restituisce la risposta corretta.
4. Il dialogo continua finché l'utente decide di uscire.

Questa struttura rende il codice ordinato, facile da comprendere e funzionale, garantendo un'interazione fluida tra utente e assistente.