U1W2 – Progetto

Analisi e correzione del codice.

Richieste:

Dato il codice in immagine

- analizzarlo per capire cosa fa;
- correggere gli errori di sintassi/logici
- individuare casi non gestiti
- proporre migliorie
- presentare una soluzione a ciascuno degli errori

```
1   import datetime
2
3   while True
4       comando_utente = input("Cosa vuoi sapere? ")
5       if comando_utente == "esci":
6           print("Arrivederci!")
7       break
8       else:
9           print(assistente_virtuale(comando_utente))
10
11  def assistente_virtuale(comando):
12      if comando == "Qual è la data di oggi?":
13          oggi = datetime.datetoday()
14          risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
15      elif comando == "Che ore sono?":
16          ora_attuale = datetime.datetime.now().time()
17          risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
18      elif comando == "Come ti chiami?":
19          risposta = "Mi chiamo Assistente Virtuale"
20      else:
21          risposta = "Non ho capito la tua domanda."
22      return risposta
23
```

Figura 1 Codice base

Analisi del codice:

Il codice si presenta come un programma di **assistenza virtuale** che fornisce informazioni semplici su data e ora rispondendo alle domande dell'utente.

Presenta però dei problemi logici, sintattici e problemi generali di comprensione del funzionamento per l'utente finale, che infatti si troverà a non sapere che domande può porre esattamente.

Il codice si struttura richiedendo un **input** all'utente che deve rispondere alla domanda "Cosa vuoi sapere?" e, in base alla richiesta posta dall'utente finale, viene selezionata una delle opzioni tramite una serie di *if-elif-else*, cercando una **corrispondenza esatta** con il quesito dell'utente.

La risposta alle domande utilizza principalmente la libreria *datetime* per permettere al programma di estrapolare data e ora del dispositivo su cui è in esecuzione.

Nel caso in cui **non** ci sia **corrispondenza esatta**, il programma risponderà con "Non ho capito la tua domanda." e chiederà un **nuovo input** da parte dell'utente con "Cosa vuoi sapere?" fino all'inserimento del comando "**esci**" (sempre con corrispondenza esatta). A questo punto l'Assistente virtuale saluterà con "Arrivederci!" e il programma si chiuderà.

Correzione degli errori di sintassi/logici:

Anzitutto è necessario dire che ci sono due modi differenti di procedere con la correzione del codice, ed essi sono basati sulla decisione di cambiare o meno l'import della libreria datetime.

Supponendo di cambiare l'import della libreria, gli errori sintattici si presentano così:

- r1 – import datetime → from datetime import datetime
- r3 – while True → assenza dei due punti dopo il True → while True:
- r7 – break → indentazione scorretta, necessario indentarlo maggiormente e metterlo in linea con il comando *print()* della r6 per permettere la corretta uscita dal ciclo
- r13 datetime.datetoday() → metodo inesistente → datetime.today() è il metodo corretto
- r16 datetime.datetime.now().time() → datetime.now().time()

Vi è inoltre un grosso errore logico nella disposizione dei blocchi di codice: il ciclo while, infatti, richiama un metodo dichiarato dopo di lui e, dato che in quel preciso snippet di codice stiamo utilizzando python in modo imperativo, la dichiarazione *di def assistente_virtuale(comando)* deve avvenire prima del ciclo. In alternativa si può ovviare al problema inserendo il ciclo *while* in una funzione *def main()* richiamata alla fine con *if __name__ == "__main__": main()*. Questo permette di definire tutte le funzioni all'inizio del file e consente inoltre di importare il modulo in altri script senza eseguire automaticamente il programma.

La decisione di definire una funzione *main()* è utile anche per la struttura del codice, in quanto si presenterà in maniera più pulita e professionale.

Come piccola nota a margine, è anche possibile usare *datetime.now()* sia per ottenere la data che l'ora, differenziandole solamente tramite la formattazione con *strftime()*. Questo approccio consente di usare un solo metodo invece di utilizzare *today()* per la data e *time()* per l'ora.

Individuazione casi non gestiti, proposta migliorie:

Un problema del codice, come accennato nella sezione “Analisi del codice”, è la presenza di evidenti **problemi di usabilità** che porteranno l’utente ad avere un’esperienza frustrante con il programma.

Tra questi problemi, si possono evidenziare:

- l’assenza di un **menù** o di istruzioni iniziali;
- l’assenza di **suggerimenti** in caso di input errato;
- l’**input case-sensitive**;
- la richiesta di **corrispondenza esatta** delle stringhe;
- l’impossibilità di uscire facilmente dal programma.

Per ovviare a questi problemi, suggerirei di inserire un **menù iniziale** che evidenzi le possibilità dell’utente e le domande che può porre all’assistente virtuale, oltre all’inserimento di alcuni **suggerimenti** nell’*else* finale. Oltre a questo è necessario **normalizzare l’input**, andando quindi a usare un metodo come *lower()* o *upper()* per trasformare il testo o interamente in lettere minuscole o in maiuscole, adattando le corrispondenze da noi create allo stesso modo. Sarebbe utile unire questa prima normalizzazione al metodo *strip()* per rimuovere eventuali spazi iniziali e/o finali inseriti dall’utente, così come tab o a capo, che andrebbero a far fallire la verifica di una **corrispondenza esatta** con le stringhe da noi proposte.

Inserendo un **menù iniziale** in cui si spiega all’utente come uscire, si ovvierebbe anche al problema dell’impossibilità di **uscire** facilmente dal programma.

Per ovviare ulteriormente al problema della **corrispondenza esatta**, si potrebbe inoltre andare a modificare il codice in maniera più significativa, proponendo sul menù delle **scelte numeriche** all’utente e gestendo una corrispondenza tra queste e i nostri casi proposti nei vari *if-elif-else*, mantenendo sull’else la risposta “Non ho capito la tua domanda”.

Si potrebbero inoltre usare le **stringhe formattate** invece di concatenare gli output per migliorare la manutenibilità del codice e le performance.

Presentazione di soluzioni agli errori:

Ho deciso di andare a proporre due possibili soluzioni agli errori, una con l'import della libreria modificato, uno con l'import della libreria lasciato come nel codice sorgente.

Anzitutto l'opzione con l'import della libreria uguale a quello del codice sorgente e l'assenza della definizione della funzione *main()*:

```
import datetime

def assistente_virtuale(comando):
    if comando == "qual è la data di oggi?":
        oggi = datetime.datetime.today()
        risposta = f'La data di oggi è {oggi.strftime("%d/%m/%Y")}'
    elif comando == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = f"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

print("=== ASSISTENTE VIRTUALE ===")
print("Puoi chiedermi:")
print("Qual è la data di oggi?")
print("Che ore sono?")
print("Come ti chiami?")
print("Esci (per uscire)")
print("--"*13) #per creare un divisore ---------- Evito che venga ripetuto

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

Figura 2 correzione del codice senza modificare l'import della libreria

Poi l'opzione con la modifica dell'import e con la definizione e l'utilizzo della funzione *main()*:

```python
from datetime import datetime

def assistente_virtuale(comando):
    if comando == "qual è la data di oggi?":
        oggi = datetime.today()
        risposta = f'La data di oggi è {oggi.strftime("%d/%m/%Y")}'
    elif comando == "che ore sono?":
        ora_attuale = datetime.now().time()
        risposta = f"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

def main() :
    print("=== ASSISTENTE VIRTUALE ===")
    print("Puoi chiedermi:")
    print("Qual è la data di oggi?")
    print("Che ore sono?")
    print("Come ti chiami?")
    print("Esci (per uscire)")
    print("--"*13) #per creare un divisore ---------- Evito che venga ripetuto

    while True:
        comando_utente = input("\nCosa vuoi sapere? ").lower().strip()
        if comando_utente == "esci":
            print("Arrivederci")
            break
        else:
            print(assistente_virtuale(comando_utente))

if __name__ == "__main__":
    main() #esecuzione del codice solo direttamente
```

Figura 3 Correzione codice, cambio import, definizione main()