ESAME PRATICO S2

Analizzare il seguente codice, individuare gli errori e correggerli.
Spiegare cosa farà il codice una volta eseguito.
Suggerire eventuali metodi per migliorare il codice

CODICE

```
import datetime

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

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

```

In questo programma si cerca di creare un **assistente virtuale** grazie al quale, inserendo determinati prompt prestabiliti, ci verrà restituita data di oggi, orario attuale ed il nome dell'assistente vocale.

Le **risposte** in base al prompt inserito verranno gestite all'interno della funzione *assistente_virtuale()* in cui noi passeremo una variabile (L'input dell'utente) e la funzione ci restituirà il valore di risposta (*return risposta*).

Il codice tuttavia contiene alcuni errori di sintassi e di "ordine strutturale" che non permettono il corretto funzionamento del programma una volta eseguito.

ELENCO ERRORI

→**A riga 3** c'è un errore di sintassi, manca "**:**" dopo True che indicherà a partire da quale punto inizia il blocco di codice che il ciclo deve eseguire. La riga corretta è **while True:**

→**A riga 7** il **break** ha un'indentazione errata; dovrebbe trovarsi all'interno della **if**, non fuori.
Con questa struttura si uscirebbe subito dal ciclo a prescindere da cosa scriviamo nell'if interrompendo così il programma.

→**A riga 9** andiamo a chiamare la funzione *assistente_virtuale()*; con la struttura attuale, durante l'esecuzione del programma, riscontreremo un errore poiché quella funzione viene definita dopo il ciclo while.

Per risolvere questo propongo due modi:

- Spostare la funzione *assistente_virtuale()* PRIMA del ciclo in modo da chiamarla solo dopo averla definita.

- In alternativa occorrerebbe racchiudere il blocco di codice che va da riga 3 a riga 9 nella funzione *main()* e poi in basso chiamare la funzione main usando la struttura standard if ***–name-- == "—main—": main()***.

→ **A riga 13** è errato scrivere *datetime.datetoday()* poiché nel modulo datetime non è presente nessuna funzione *datetoday()*, esiste però *today()* compresa nella classe date che ci restituisce la data di oggi (ciò che serve a noi). Quindi scriverò **datetime.date.today()**.

→ **A riga 16** cerchiamo di chiamare la funzione time omettendo le parentesi finali. La versione corretta è *datetime.datetime.now().time()* .

VERSIONI CORRETTE DI CODICE

→ In questa versione ho semplicemente spostato la funzione *assistente_virtuale()* sopra il loop in modo da definirla prima di utilizzarla. Infine ho corretto gli errori di spaziatura e di sintassi identificati precedentemente.

Vediamolo:

```python
#Metodo 2 (Struttura semplice e diretta)
import datetime

def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        oggi = datetime.date.today()
        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 == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

→ In quest’altra versione ho creato una funzione *main()* in modo da far eseguire innanzitutto ciò che metto al suo interno (il ciclo while in questo caso). Ho corretto anche qui gli errori di spaziatura e di sintassi identificati precedentemente.

Vediamolo:

```python
#Metodo 1 (Più pulito ed efficente)
import datetime

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

def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        oggi = datetime.date.today()
        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

if __name__ == "__main__":
    main()

```

TIPS

Per una questione di **efficienza** del programma suggerisco di andare a semplificare i comandi che l'utente deve inserire al fine di ottenere una risposta in modo da semplificare la richiesta e limitare eventuali errori di sintassi nel prompt.
In questo caso io utilizzerei una **lista** di numeri interi [0,1,2,3 …] ad ogni numero corrisponde un determinato comando.
Oltre a questo farei il print della lista di comandi sia all'inizio del programma sia nel caso in cui si sbaglia il comando immesso (*un vero e proprio Tutorial per l'utente*).
Ho creato una funzione tutorial_comandi() in modo da poterla chiamare ogni volta che serve.

In questo modo la struttura del codice risulta più ordinata ed inoltre avremo semplificato la vita all'utente migliorandone l'esperienza!

Esempio:

```python
#Metodo con lista e tutorial
import datetime

lista_comandi = [0, 1, 2] #0 = data, 1 = ora, 2 = nome.

def main():
    tutorial_comandi()
    while True:
        comando_utente = input("Cosa vuoi sapere? ")

        if comando_utente == "esci":
            print("Arrivederci!")
            break
        else:
            print(assistente_virtuale(comando_utente))

def assistente_virtuale(comando):
    if comando == "0":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
    elif comando == "1":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
    elif comando == "2":
        risposta = "Mi chiamo Assistente Virtuale"
    else:
        risposta = "Non ho capito la tua domanda."
        tutorial_comandi()
    return risposta

def tutorial_comandi(): #creo una funzione che spara una lista di comandi disponibile da usare all'inizio del programma e qualora si commettesse un errore.
    print('ELENCO COMANDI \n 0 = data \n 1 = ora \n 2 = nome \n "esci" per uscire dal programma')

if __name__ == "__main__":
    main()
```

Nicolò Calì