Report - Assistente Virtuale in Python

Dopo un'analisi approfondita del codice, abbiamo individuato e corretto diversi errori di sintassi e logica, migliorando l'efficienza e la chiarezza generale del programma. Inoltre, abbiamo affrontato alcune limitazioni strutturali presenti nella versione originale, come l'assenza di un menu interattivo e la mancata gestione degli input dell'utente (spazi vuoti, inserimenti non validi). Le modifiche apportate hanno reso il programma più intuitivo, robusto e orientato all'utente, offrendo un'esperienza d'uso più fluida e comprensibile.

Codice iniziale - Errori

Durante l'analisi del codice iniziale, abbiamo riscontrato una serie di errori sia **di sintassi** che **di logica** che compromettevano il corretto funzionamento e la chiarezza del 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))
```

Errori di sintassi (Evidenziati nel codice con commento)

1. Mancanza dei `:` alla fine del ciclo `while True`.
2. Errore nella scrittura di `datetime.datetoday` (metodo inesistente).

Errori logici (Evidenziati nel codice con commento)

1. Uso non necessario di `.time()`, in quanto si può ottenere lo stesso risultato utilizzando direttamente `.now()` con formattazione tramite `.strftime()`.

Limiti del programma originale

Il programma originale **non offriva un menu** che aiutasse l'utente a comprendere le funzionalità disponibili. Inoltre, **non gestiva correttamente input con spazi** o con lettere maiuscole/minuscole.

Modifiche apportate

- Aggiunta dei `:` mancanti per il corretto funzionamento del ciclo `while`.
- Correzione della sintassi: sostituito `datetime.datetoday` con `date.today()` dopo un'importazione mirata.
- Eliminazione dell'utilizzo di `.time()` e semplificazione del codice tramite `.now().strftime()`.
- Creazione di un **menu numerico interattivo** per guidare l'utente e rendere l'esperienza più chiara. Poiché l'interazione avviene tramite **numeri**, non è stato necessario utilizzare `.lower()`, in quanto non vengono richiesti input testuali soggetti a variazioni di maiuscole o minuscole.
- Ottimizzazione dell'importazione: abbiamo importato **solo** ciò che serve (`from datetime import date, datetime`) per evitare ambiguità e confusione.
- Utilizzo della funzione `.strip()` per **eliminare eventuali spazi** all'inizio o alla fine dell'input utente, evitando errori di confronto e migliorando la robustezza del sistema.

```python
from datetime import date, datetime

def menu_utente():
    print("\nMenu Assistente Virtuale")
    print("1. Qual è la data di oggi?")
    print("2. Che ore sono?")
    print("3. Come ti chiami?")
    print("4. Esci\n")

def assistente_virtuale(comando):
    if comando == "1":
        oggi = date.today()
        risposta = "\n --> La data di oggi è " + oggi.strftime("%d/%m/%Y")
    elif comando == "2":
        ora_attuale = datetime.now()
        risposta = "\n --> L'ora attuale è " + ora_attuale.strftime("%H:%M")
    elif comando == "3":
        risposta = "\n --> Mi chiamo Assistente Virtuale."
    elif comando == "4":
        return None
    else:
        risposta = "\n --> Comando non valido. Inserisci un numero da 1 a 4."
    return risposta

while True:
    menu_utente()
    comando = input("Inserisci il numero del comando: ").strip()

    risposta = assistente_virtuale(comando)

    if risposta is None:
        print("Arrivederci!\n")
        break
    else:
        print(risposta)
```

Conclusione

Le modifiche effettuate hanno permesso di trasformare un codice inizialmente instabile e poco chiaro in un programma solido, facilmente comprensibile e più vicino a un’esperienza d’uso professionale. L’approccio orientato all’utente e la cura dei dettagli tecnici hanno contribuito a rendere il progetto più affidabile ed efficace.