

Analisi e Miglioramento di un Assistente Virtuale

1. *Introduzione*
2. *Descrizione del Programma*
3. *Casistiche Non Standard Non Gestite*
4. *Errori di Sintassi e Logici*
5. *Proposte di Soluzione*
6. *Screenshoot*
7. *Conclusioni*

Introduzione

Questa relazione ha l'obiettivo di analizzare un programma Python che crea un semplice assistente virtuale. Il programma è in grado di rispondere a comandi come la data e l'ora attuali e al nome dell'assistente. Tuttavia, ci sono alcuni errori di sintassi, problemi logici e casi non gestiti correttamente. Nella relazione, vedremo quali sono questi problemi e cercheremo di trovare delle soluzioni per migliorare il programma.

Descrizione del Programma

Il programma è un assistente virtuale che risponde a tre comandi principali:

1. **"Qual è la data di oggi?":** Il programma restituisce la data corrente nel formato giorno/mese/anno (gg/mm/aaaa).
2. **"Che ore sono?":** Il programma restituisce l'ora attuale nel formato ore:minuti (HH:MM).
3. **"Come ti chiami?":** Il programma risponde dicendo "Mi chiamo Assistente Virtuale".

*Se l'utente scrive qualcosa che il programma non riconosce, risponde con "Non ho capito la tua domanda".
Se l'utente scrive "esci", il programma si chiude.*

Casistiche Non Standard Non Gestite

Il programma non gestisce correttamente alcune situazioni non standard:

- **Input con spazi extra o maiuscole/minuscole:** Ad esempio, "CHE ORE SONO?" o "che ore sono" non vengono riconosciuti.
- **Formati alternativi di data/ora:** L'utente potrebbe volere la data in un formato più descrittivo (es. "12 ottobre 2023") o l'ora in un formato diverso.
- **Comandi parziali o simili:** Frasi come "Che giorno è oggi?" o "Dimmi l'ora" non sono gestite.

- **Input vuoto:** Se l'utente preme Invio senza scrivere nulla, il programma non fornisce un feedback.
- **Errori di sistema:** Non ci sono meccanismi per gestire eventuali errori del modulo `datetime`.

Errori di Sintassi e Logici

Sono stati identificati i seguenti errori:

1. **Errore di sintassi:** `datetime.datetoday()` non esiste. Il metodo corretto è `datetime.date.today()`.
2. **Errore di sintassi:** Manca il simbolo `:` nel ciclo `while True`.
3. **Errore logico:** Il programma non normalizza l'input (ad esempio, non converte l'input in minuscolo o rimuove spazi extra).
4. **Errore logico:** Non gestisce input vuoti o comandi simili ma non identici a quelli previsti.

Proposte di Soluzione

Per risolvere i problemi individuati, sono state proposte le seguenti soluzioni, suddivise in errori di **sintassi** ed **errori logici**:

Errori di Sintassi:

1. **Correzione della funzione per la data:**
Si deve usare `datetime.date.today()` invece di `datetime.datetoday()`, che è un errore di scrittura.
2. **Aggiungere i due punti nei cicli:**
Nel ciclo `while True`, bisogna mettere i due punti `:` per far partire il blocco di codice.

Errori Logici:

1. **Normalizzazione dell'input:**
Per evitare errori quando l'utente scrive comandi con spazi extra o in maiuscolo, è utile usare `.strip()` per rimuovere gli spazi e `.lower()` per rendere tutto minuscolo. Questo aiuta a confrontare meglio l'input.
2. **Gestione di input simili:**
Per fare in modo che il programma capisca comandi simili, ad esempio "data oggi" o "oggi data", si potrebbe usare una condizione tipo `if "data" in comando and "oggi" in comando` per cercare parole chiave nell'input.
3. **Gestione degli input vuoti:**
Se l'utente non scrive nulla e preme invio, bisogna verificare se l'input è vuoto e chiedere nuovamente di inserire un comando valido.
4. **Formati alternativi di data/ora:**
Potremmo offrire formati alternativi più completi per la data, come `"%d %B %Y"`, che mostra la data con il nome del mese (ad esempio: "07 febbraio 2025").
5. **Gestione delle eccezioni:**
Aggiungere blocchi `try-except` per catturare eventuali errori che potrebbero verificarsi con il modulo `datetime`, ad esempio se l'input dell'utente è formattato male o se ci sono problemi con la data o l'ora.

Screenshoot

```
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."

while True

    comando_utente = input("Cosa vuoi sapere? ")

    if comando_utente.lower() == "esci":

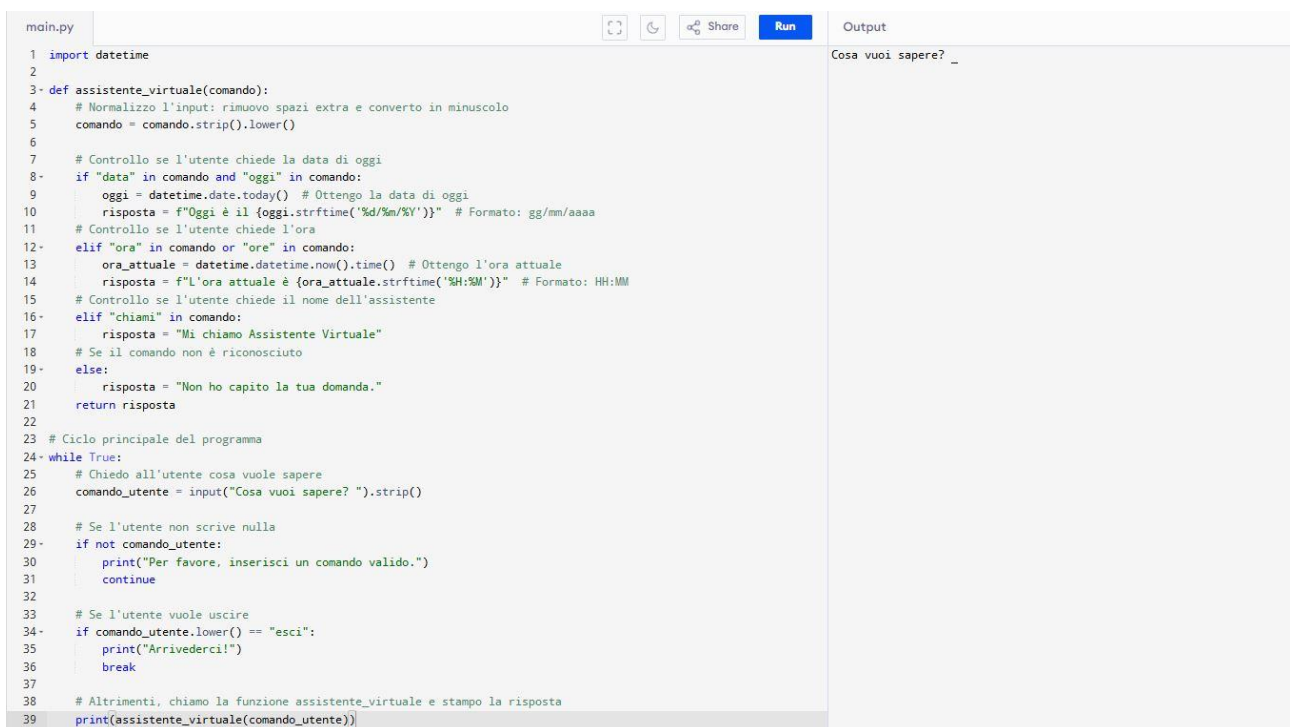
        print("Arrivederci!")

        break

    else:

        print(assistente_virtuale(comando_utente))
```

1- La figura rappresenta il codice con errori di casistiche non standard, sintassi o logici.



```
main.py  [Icons]  Share  Run  Output

1 import datetime
2
3 def assistente_virtuale(comando):
4     # Normalizzo l'input: rimuovo spazi extra e converto in minuscolo
5     comando = comando.strip().lower()
6
7     # Controllo se l'utente chiede la data di oggi
8     if "data" in comando and "oggi" in comando:
9         oggi = datetime.date.today() # Ottengo la data di oggi
10        risposta = f"Oggi è il {oggi.strftime('%d/%m/%Y')}" # Formato: gg/mm/aaaa
11        # Controllo se l'utente chiede l'ora
12    elif "ora" in comando or "ore" in comando:
13        ora_attuale = datetime.datetime.now().time() # Ottengo l'ora attuale
14        risposta = f"L'ora attuale è {ora_attuale.strftime('%H:%M')}" # Formato: HH:MM
15        # Controllo se l'utente chiede il nome dell'assistente
16    elif "chiami" in comando:
17        risposta = "Mi chiamo Assistente Virtuale"
18        # Se il comando non è riconosciuto
19    else:
20        risposta = "Non ho capito la tua domanda."
21    return risposta
22
23 # Ciclo principale del programma
24 while True:
25     # Chiedo all'utente cosa vuole sapere
26     comando_utente = input("Cosa vuoi sapere? ").strip()
27
28     # Se l'utente non scrive nulla
29     if not comando_utente:
30         print("Per favore, inserisci un comando valido.")
31         continue
32
33     # Se l'utente vuole uscire
34     if comando_utente.lower() == "esci":
35         print("Arrivederci!")
36         break
37
38     # Altrimenti, chiamo la funzione assistente_virtuale e stampo la risposta
39     print(assistente_virtuale(comando_utente))

Cosa vuoi sapere? _
```

2- La figura rappresenta il codice senza errori di casistiche non standard, sintassi o logici.

Conclusioni

Attraverso questa analisi, è stato possibile identificare e correggere gli errori del programma originale, migliorandone la solidità e l'usabilità. Il programma ora gestisce più casistiche, normalizza l'input e fornisce riscontri più descrittivi. Questo esercizio mostra quanto sia importante pensare in modo diverso senza soffermarsi mai sulle apparenze.