

Traccia S2/L5

Per agire come un Hacker bisogna capire come pensare fuori dagli schemi. L'esercizio di oggi ha lo scopo di allenare l'osservazione critica.

Dato il codice si richiede allo studente di:

- Capire cosa fa il programma senza eseguirlo.
 - Individuare nel codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati).
 - Individuare eventuali errori di sintassi / logici.
 - Proporre una soluzione per ognuno di essi.
-

Capire cosa fa il programma senza eseguirlo

Il programma che mi è stato fornito implementa un **assistente virtuale** di base che risponde a domande specifiche dell'utente. Utilizza la libreria **datetime** per recuperare la data e l'orario correnti. La funzione **assistente_virtuale()** gestisce tre comandi principali: "**Qual è la data di oggi?**", "**Che ore sono?**" e "**Come ti chiami?**". Per ogni comando, il programma restituisce una risposta predefinita. Se il comando non è riconosciuto, restituisce "**Non ho capito la tua domanda.**"

Il programma funziona in un ciclo infinito, chiedendo all'utente di inserire un comando tramite la funzione **input()**. Se l'utente inserisce "**esci**", il programma stampa "**Arrivederci!**" e termina. Se l'utente inserisce un comando valido, il programma risponde con la funzione appropriata. Ho notato alcuni errori nel codice iniziale, come errori di sintassi e problemi nel formato delle date e dell'orario, che ho corretto in seguito. Inoltre, ho reso i comandi **case-insensitive** per permettere all'utente di scrivere le domande sia in maiuscolo che in minuscolo. Ora il programma dovrebbe funzionare correttamente.

Individuare nel codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati).

Nel programma ci sono alcune casistiche non gestite che potrebbero causare problemi:

- **Comandi non riconosciuti o malformati:** Se l'utente scrive un comando con errori di battitura o una sintassi diversa, il programma non riesce a interpretarlo correttamente e risponde genericamente con "Non ho capito la tua domanda". Non c'è un messaggio che aiuti l'utente a capire cosa fare.
- **Input non valido:** Il programma non gestisce eventuali input imprevisti, come caratteri speciali o numeri al posto di comandi testuali, e non offre alcun feedback in caso di errore.
- **Mancanza di gestione degli errori di sistema:** Se c'è un problema con la libreria **datetime** (ad esempio, se il sistema non riesce a restituire la data o l'ora), il programma potrebbe non funzionare correttamente, ma non c'è una gestione delle eccezioni per questo caso.
- **Comandi incompleti o imprecisi:** Il programma non è tollerante a errori di sintassi minori, come la mancanza di un punto interrogativo o variazioni nei comandi, il che potrebbe impedire all'assistente di rispondere correttamente.

Per migliorare, suggerisco di aggiungere una gestione degli errori più robusta, consentire una maggiore tolleranza nei comandi e fornire messaggi più chiari per aiutare l'utente a interagire meglio con il programma.

Errori di sintassi nel codice

`datetime.datetime0:`

Questo è un errore di sintassi. `datetime.datetime0` non esiste.

Correzione: il metodo corretto per ottenere la data odierna è `datetime.datetime.today()`.

`ora_attuale = datetime.datetime.now).time():`

C'è una parentesi di troppo dopo `now`, ed è scritta male.

Correzione: la sintassi corretta è `datetime.datetime.now().time()`. La parentesi è corretta così.

`oggi.strptime(*%d/%m/% Y"):`

Sintassi errata nella funzione `strptime`. Dovrebbe essere `oggi.strptime("%d/%m/%Y")` (senza gli asterischi e con la corretta formattazione della data).

`ora_attuale strftime("%H:%M*):`

Mancano parentesi e virgolette per il metodo `strftime`.

Correzione: `ora_attuale.strftime("%H:%M")` (per formattare correttamente l'orario).

`if comando_utente lower == "esci". print("'Arrivederci!"):`

Il confronto `comando_utente lower == "esci"` è sbagliato. Dovrebbe essere `comando_utente.lower() == "esci"`. Inoltre, c'è un errore di sintassi con il punto e la parentesi.

Correzione: `if comando_utente.lower() == "esci":` e `print("Arrivederci!")` (senza il ' errato).

Errori logici nel codice

Gestione delle risposte:

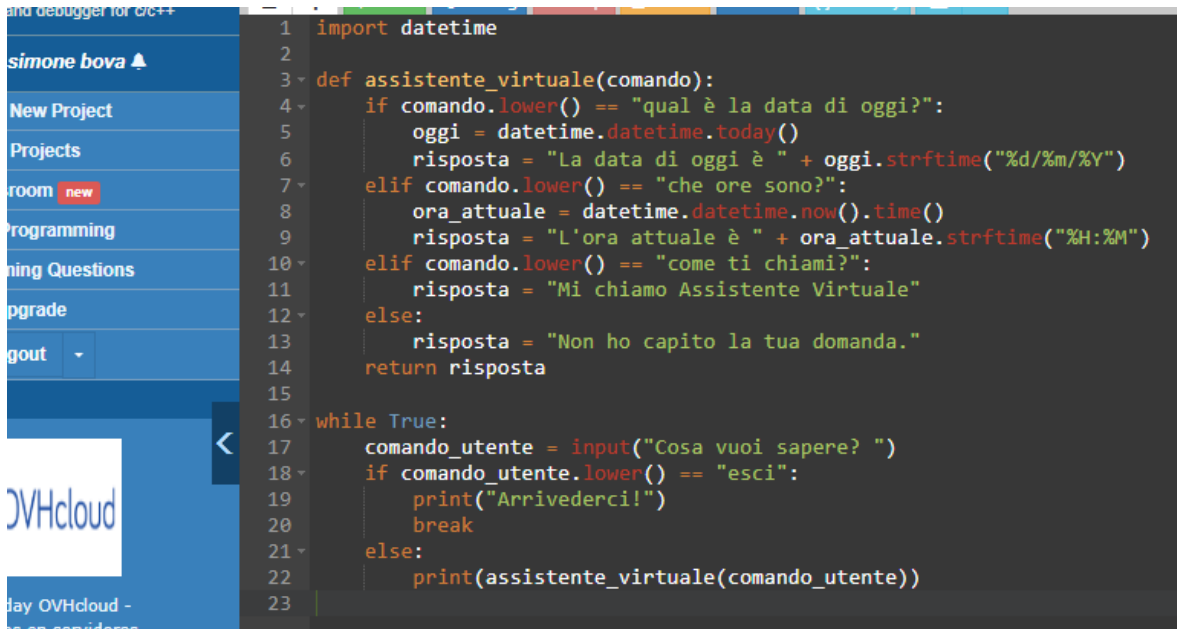
La funzione **`assistente_virtuale`** ritorna una risposta solo se uno dei comandi è esattamente uguale (**case-sensitive**) a quelli definiti nel codice. Sarebbe più utile rendere il controllo del comando **case-insensitive**.

Soluzione: usare **`comando.lower()`** per gestire anche i comandi scritti in maiuscolo o minuscolo.

Gestione degli errori:

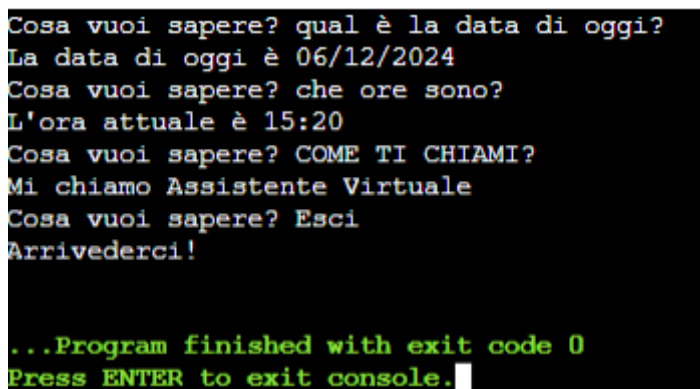
Se l'utente fornisce un comando non riconosciuto, il programma risponde con **"Non ho capito la tua domanda"**. Questo è corretto, ma si potrebbe migliorare l'esperienza dell'utente con risposte più dettagliate o una gestione più robusta degli errori.

Adesso allego uno screenshot del codice modificato, completo e corretto:

A screenshot of a code editor interface. On the left is a sidebar with a blue background containing navigation links: 'simone bova', 'New Project', 'Projects', 'room', 'Programming', 'ning Questions', 'pgrade', 'gout', and 'OVHcloud'. The main area shows Python code for a virtual assistant. The code includes imports, a function definition, and a main loop.

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

Inoltre grazie alla funzione **case-insensitive** (insensibile al maiuscolo/minuscolo) il programma ha la capacità di **ignorare le differenze tra lettere maiuscole e minuscole** quando confronta o gestisce stringhe di testo.

A screenshot of a terminal window with a black background and white text. It shows the interaction between a user and the virtual assistant. The user enters commands, and the assistant responds with the current date, time, its name, and a goodbye message. The terminal also shows the program's exit status.

```
Cosa vuoi sapere? qual è la data di oggi?
La data di oggi è 06/12/2024
Cosa vuoi sapere? che ore sono?
L'ora attuale è 15:20
Cosa vuoi sapere? COME TI CHIAMI?
Mi chiamo Assistente Virtuale
Cosa vuoi sapere? Esci
Arrivederci!

...Program finished with exit code 0
Press ENTER to exit console.
```

In questo programma ho corretto tutti gli errori di sintassi (come parentesi e metodi errati) e sistemato i vari comandi in modo da permettere la corretta esecuzione del programma.