Progetto S2/L5

Traccia: 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:

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

CODICE DA ANALIZZARE

```
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
```

Spiegazione

Il codice analizzato ha lo scopo di creare un assistente virtuale che risponde a 3 domande preimpostate:

1. Qual è la data di oggi?
2. Che ore sono?
3. Come ti chiami?

Le domande sono state inserite all'interno di un ciclo infinito terminabile solamente con il comando "esci". Il codice presenta errori che ne compromettono la funzionalità e impediscono l'esecuzione dello stesso.

Analisi degli Errori

- **Mancanza dei due punti (`:`):**
  - **Errore:** Alla fine dell'istruzione `while True (riga 3).` L'istruzione `while True` apre un blocco di codice (il ciclo infinito). In Python, tutte le dichiarazioni che aprono un blocco devono terminare con i due punti.
  - **Soluzione:** `while True :`
- **Indentazione del `break`:**
  - **Errore:** Il comando `break(riga 7)` deve essere indentato all'interno dell'`if`. Se posto allo stesso livello, il ciclo si interromperebbe immediatamente dopo la prima esecuzione, rendendo inutile il `while.`
- **Metodo inesistente:**
  - **Errore:** Viene chiamato `datetime.datetoday()` , che non esiste nella libreria standard.
  - **Soluzione:** Il metodo corretto è `datetime.date.today()`.
- **Case Sensitivity (Sensibilità alle maiuscole):**
  - **Problema:** I controlli `if comando == ...` funzionano solo se l'utente rispetta esattamente le maiuscole/minuscole.
  - **Soluzione:** Convertire l'input utente con `.lower()` per rendere il programma più "intelligente" (ovvero, non fa distinzione tra maiuscole e minuscole).
- **Ordine di definizione:**
  - **Problema**: Il ciclo `while True` (linee 3-9) chiama la funzione `assistente_virtuale` alla linea 9, ma la funzione viene definita solo in seguito (alla linea 11). Quando si arriva alla linea 9, il programma andrà in crash perché non conosce ancora il nome `assistente_virtuale`.
  - **La Soluzione:** Bisogna spostare l'intera definizione della funzione (`def assistente_virtuale...`) **prima** del ciclo `while`.

Codice Corretto

Progetto _S2_L5.py > ...

```python
import datetime

#  La funzione deve essere definita PRIMA di essere usata
def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        # Sintassi corretta è datetime.date.today()
        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

# Ciclo principale
while True:
    comando_utente = input("Cosa vuoi sapere? ")
    if comando_utente == "esci":
        print("Arrivederci!")
        break
    else:
        # Ora la funzione è definita e può essere chiamata
        print(assistente_virtuale(comando_utente))
```