

Marco Falchi

Epicode S1L5

Il compito di oggi propone un'analisi di un dato codice in python tramite la seguente consegna:



Esercizio
Progetto

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.

Commenti pre esercizio: Il compito sembra abbastanza semplice, ho già esperienza nella programmazione, ma gli errori di sintassi specialmente senza l'utilizzo di visualstudio non sono facili da notare, a primo sguardo noto già qualcosa che non va, specialmente errori non contemplati, sarà sicuramente un esercizio che mi metterà alla prova a livello mentale.

Successivamente viene fornito il codice del programma da analizzare che è il seguente:



Esercizio
Progetto

```
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
```



Esercizio
Progetto

```
while True

    comando_utente = input("Cosa vuoi sapere? ")

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

        print("Arrivederci!")

        break

    else:

        print(assistente_virtuale(comando_utente))
```

Analizzo il programma (riassunto):

Il precedente programma Python crea un assistente virtuale testuale semplice e con poche funzioni.

Il programma opera in un ciclo continuo (usando la funzione while), chiedendo all'utente cosa vuole sapere. L'utente può porre tre domande specifiche ("Qual è la data di oggi?", "Che ore sono?", "Come ti chiami?") per ottenere risposte predefinite (la data corrente, l'ora corrente, o il nome dell'assistente). Se la domanda non è riconosciuta, l'assistente risponde che non ha capito. L'utente può terminare il programma digitando "esci" altrimenti dopo ogni domanda rinizierà il ciclo e l'assistente chiederà nuovamente "Cosa vuoi sapere?".

Errori di sintassi e correzione:

Non noto particolari errori di sintassi, dopo numerose analisi ho notato principalmente due errori di sintassi che sono i seguenti:

Errore 1:

```
while True

    comando_utente = input("Cosa vuoi sapere? ")

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

        print("Arrivederci!")

        break

    else:

        print(assistente_virtuale(comando_utente))
```

Nella funzione while Python richiede i due punti (:) alla fine della sua condizione per delimitare l'inizio della funziona stessa.

Correzione: while :

Errore 2:

```
import datetime

def assistente_virtuale(comando):

    if comando == "Qual è la data di oggi?":

        oggi = datetime.datetime.today()

        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

    elif comando == "Che ore sono?":
```

All'inizio del codice viene riportato il modulo/libreria "datetime" ma nella riga 4 nella stringa di codice "oggi = datetime.datetime.datetime.today()" si trova un errore di battitura, infatti la libreria datetime non contiene nessun attributo chiamato "datetime.today()".

Correzione: oggi = **datetime.date.today()**

Analisi delle casistiche non standard e correzioni:

Queste correzioni si concentrano sui comportamenti del programma che pur non essendo errori di sintassi o blocchi totali, ne limitano l'usabilità e l'intuitività. Ne ho trovati alcuni che elencherò qua sotto con le rispettive correzioni:

Problema 1:

Il programma nella funzione "esci" tratta ogni input dell'utente (maiuscoli o minuscoli) allo stesso modo, questo grazie al .lower (if comando_utente.**lower()** == "esci":) ma nelle funzioni interne al programma non si tiene conto di questo, quindi basterebbe una maiuscola in più o in meno per creare dei "problemi", si può quindi risolvere facilmente la problematica convertendo sempre gli input dati dall'utente in un formato totalmente minuscolo.

Problema 2:

Come nel problema sopraelencato il programma non tiene conto di eventuali errori da parte dell'utente in input, quindi oltre a maiuscole e minuscole bisogna tener conto anche di eventuali problemi di spaziature indesiderate nelle domande, problema anch'esso facilmente risolvibile con la funzione **.strip**.

Problema 3:

Il programma riconosce solo input contenenti la formulazione *esatta* dei comandi predefiniti. Non sono previste frasi alternative o sinonimi (esempio: "Che giorno è oggi?" invece di "Qual è la data di oggi?").

L'utente quindi dovrà "azzeccare" la frase esatta e ovviamente crea confusione nell'utente.

Questa "problematica" è risolvibile attraverso un utilizzo della funzione if/elif/else e della funziona or come ad esempio inserendo più input possibili come:

```
def assistente_virtuale(comando_alternativa):
    if comando_alternativa == "qual è la data di oggi?" or \
        comando_alternativa == "che giorno è oggi?" or \
        comando_alternativa == "data di oggi":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
    elif comando_alternativa == "che ore sono?" or \
        comando_alternativa == "l'ora attuale?" or \
        comando_alternativa == "dimmi l'ora":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
    elif comando_alternativa == "come ti chiami?" or \
        comando_alternativa == "chi sei?" or \
        comando_alternativa == "dimmi il tuo nome?":
        risposta = "Mi chiamo Assistente Virtuale"
    else:
        risposta = "Non ho capito la tua domanda."
    return risposta
```

Sono ovviamente state inserite solo alcune delle mille casistiche di domande possibili ma la vedo come una soluzione un pochino più adatta all'utente medio.

Cose che si potrebbero implementare: Le cose possibili da implementare sarebbero infinite ma ne elencherò solo alcune che renderebbero il programma migliore

- Un printf che dice all'utente le domande possibili
- Chiedere la città in caso di richiesta di orario dato che il fuso orario cambia
- Dire all'utente che per uscire dal ciclo bisogna inserire la scritta esci

Ecc...ecc...ecc..ecc

Conclusioni finali:

Anche questa settimana è finita con l'esecuzione di un compito che non mi aspettavo, difatti mi aspettavo un compito riguardante la programmazione, cosa a me molto più familiare e che ti offre maggiori possibilità e libertà ma l'analisi di un programma ha saputo

mettermi alla prova nella ricerca di errori di qualsiasi genere e tipo, concludo quindi lasciando qua sotto una versione che contiene le mie correzioni principali:

```
import datetime

def assistente_virtuale(comando_alternativa):
    if comando_alternativa == "qual è la data di oggi?" or \
        comando_alternativa == "che giorno è oggi?" or \
        comando_alternativa == "data di oggi?":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
    elif comando_alternativa == "che ore sono?" or \
        comando_alternativa == "l'ora attuale?" or \
        comando_alternativa == "dimmi l'ora?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
    elif comando_alternativa == "come ti chiami?" or \
        comando_alternativa == "chi sei?" or \
        comando_alternativa == "dimmi| tuo nome?":
        risposta = "Mi chiamo Assistente Virtuale"
    else:
        risposta = "Non ho capito la tua domanda."
    return risposta

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

    processed_command = comando_utente.strip().lower()

    if processed_command == "esci":
        print("Arrivederci!")
        break
    else:
        # Passa il comando già standardizzato alla funzione
        print(assistente_virtuale(processed_command))
```