Esame S2L5

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.

Introduzione

Nell'esame di oggi, ci è stato chiesto di analizzare un codice, individuare eventuali errori e proporre soluzioni per correggerli. Questa attività simula una situazione reale di code review, un processo fondamentale nello sviluppo di un progetto software.

Un aspetto fondamentale che va considerato quando si parla di revisione del codice (code review) è quello della prevenzione della vulnerabilità, che viene affrontata tramite la pratica del secure coding, ovvero l'insieme di tecniche di programmazione volte a garantire la sicurezza del software, proteggendolo da eventuali attacchi hacker esterni. Errori comuni, come la gestione inadeguata degli input forniti dall’utente o la mancata validazione di dati, possono esporre il software a vulnerabilità che potrebbero essere sfruttati per attacchi informatici.

Per questo la revisione del codice rappresenta un aspetto essenziale nella vita di un software, identificando e correggendo eventuali rischi prima dell’effettivo rilascio del prodotto finale.

Il Codice Preso in esame

```
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))
```

1. Capire cosa fa il programma senza eseguirlo.

Analisi

Andrò a dividere l'analisi del programma in due fasi: nella prima parte, effettuerò un'analisi generale del programma, esaminando la sua struttura e funzionalità complessive mentre nella seconda parte, invece, procederò con un'analisi dettagliata, esaminando ogni singola riga del codice.

Analisi generale

Il programma emula un assistente virtuale che risponde a comandi specifici dati dall'utente, la data odierna, l'ora attuale ed il nome dell'assistente.

Possiamo vedere una divisione nel programma, la funziona dell'assistente ed un ciclo while.

Funzione, viene richiesto un input specifico ed in base alla scelta verrà restituita una risposta, in caso il comando in input non viene riconosciuto il programma restituirà a schermo “Non ho capito la tua domanda”.

Ciclo While, il programma stampa ripetutamente la domanda “Cosa vuoi sapere?” e richiede l’inserimento di un input, se l'utente scrive “esci ” il programma stampa come risposta “Arrivederci” e si stoppa, se l'utente inserisce un comando che il programma riconosce allora torna alla funzione e stampa la risposta a schermo.

Analisi nel dettaglio

Per l’analisi nel dettaglio analizzerò il codice riga per riga spiegandone il significato, il funzionamento, andrò ad evidenziare gli errori e a proporre soluzioni.

Riga 1: import datetime

Con questo comando viene importato il modulo “datetime”, che permette al programma di lavorare con date e orari.

Riga 3: def assistente_virtuale(comando):

“def” serve a definire una funzione, “assistente_virtuale” è il nome della funzione e “comando” è il parametro della funzione che può variare in base all’input fornito dall’utente.

Riga 5: if comando == "Qual è la data di oggi?":

“If” indica l’inizio di una condizione, il programma controlla se una condizione è vera, se lo è esegue il codice che segue ad if, sennò andrà ad ignorare i codici all’interno di if.

comando == “Qual’è la data di oggi?”: è la condizione da verificare, controlla se l’input inserito dall’utente corrisponde alla stringa “Qual’è la data di oggi”.

Qui però possiamo apportare una miglioria al programma modificando la riga 5, in quanto se la domanda non viene scritta in modo identico a come è nel confronto con comando non ci darà risposta, se vogliamo fare in modo di avere più libertà con le maiuscole e le minuscole dobbiamo apportare un cambio:

if comando.lower() == “qual è la data di oggi”

scrivendo così il programma accetterà ogni combinazione di maiuscole e minuscole grazie al comando .lower().

Riga 6: oggi = datetime.datetoday()

oggi è la variabile creata per fornire la data e l'ora

Qui abbiamo un errore in quanto la funzione scritta in modo corretto sarebbe:
oggi = datetime.date.today()

datetime.date.today() è il metodo che fa restituire la data odierna.

Riga 7: risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

risposta è la variabile

“La data di oggi è”, è una stringa che verrà stampata a schermo insieme alla risposta

oggi.strftime("%d/%m/%Y"), strftime() è il metodo che converte la data in formato stringa. tra le parentesi c’è il formato day, month, year

Riga 9: elif comando == "Che ore sono?":

Anche qui come per la riga 5 apportiamo la stessa modifica, facendola diventare:

elif comando.lower() == “che ore sono?”

elif è utilizzata per specificare le condizioni aggiuntive, che verranno verificate solo se la condizione precedente non è stata soddisfatta

comando è il paramento

== confronto tra comando e la stringa successiva

“che ore sono?” è la stringa che l'utente potrebbe mettere in input come comando.

Riga 10: ora_attuale = datetime.datetime.now().time()

ora_attuale è la variabile

datetime.datetime.now() è la funzione che restituisce l’oggetto ora e la data corrente

.time() è il metodo utilizzato per avere solo la parte dell’orario

Riga 11: risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")

risposta è la variabile

“L’ora attuale è ” è la stringa che verra stampata a schermo insieme al risultato

ora_attuale.strftime("%H:%M") questo fa sì che l’oggetto time viene formattato in ora_attuale con il metodo .strftime, la stringa di formato %H che indica il formato 24 ore e %M minuti con due cifre

Riga 13: elif comando == "Come ti chiami?":

Qui come in riga 5 e 9 apportiamo la modifica trasformandola in:
elif comando.lower() == "come ti chiami?":

elif comando.lower() == (uguale a come spiegato in riga 9)
“come ti chiami?”: questa la stringa che l’utente potrebbe fare come domanda.

Riga 14: risposta = "Mi chiamo Assistente Virtuale"

risposta è la variabile in cui si memorizza la risposta da dare all’utente

"Mi chiamo Assistente Virtuale" è la stringa in risposta dell’utente

Riga 16: else:

else viene utilizzata per gestire il caso in cui nessuna delle condizioni precedenti è stata soddisfatta

Riga 17: risposta = "Non ho capito la tua domanda."

risposta è la variabile che appare in caso nessuna delle precedenti condizioni è stata soddisfatta

"Non ho capito la tua domanda." è la stringa che stamperà a schermo

Riga 19: return risposta

return viene utilizzata per restituire il valore da una funzione

risposta è la variabile con il messaggio da restituire

Riga 21: While True

qui abbiamo un errore in quanto dopo While True vanno messi i due punti trasformandola in:

While True:

While viene usata per creare un ciclo che continua ad eseguire dei codice finché la condizione che viene specificata è vera

True è una condizione che è sempre vera, il ciclo continuerà ad andare finché non incontrerà un comando che lo interrompa

Riga 22: comando_utente = input("Cosa vuoi sapere? ")

comando_utente è la variabile che memorizza ciò che l'utente inserisce in input. ogni volta che il ciclo continua esso riceve un valore nuovo.

input() permette di ricevere un input dall'utente

"Cosa vuoi sapere? " è la stringa stampata che appare all'utente

Riga 24: if comando_utente.lower() == "esci":

if serve sempre per iniziare una condizione che verifica se qualcosa è vero

comando_utente è la variabile contenente l'input dell'utente

.lower() è un metodo che serve per far accettare il comando anche se scritto con minuscolo o maiuscole diverse

== confronto

"esci" è la stringa che l'utente può digitare

Riga 25: print("Arrivederci!")

print serve a far stampare una stringa

"Arrivederci!" è la stringa

Riga 26: break

break viene utilizzata per uscire immediatamente da un ciclo, in questo caso quando vieni inserito in input "esci"

Riga 28: else:

else in questo caso è associato al ciclo while in questo caso l'input non è "esci" allora viene rimandato alla funzione e stampa la risposta

Riga 29: print(assistente_virtuale(comando_utente))

(assistente_virtuale(comando_utente)) chiama la funzione assistente_virtuale e le passa il comando dell'utente la risposta verrà stampata grazie al print

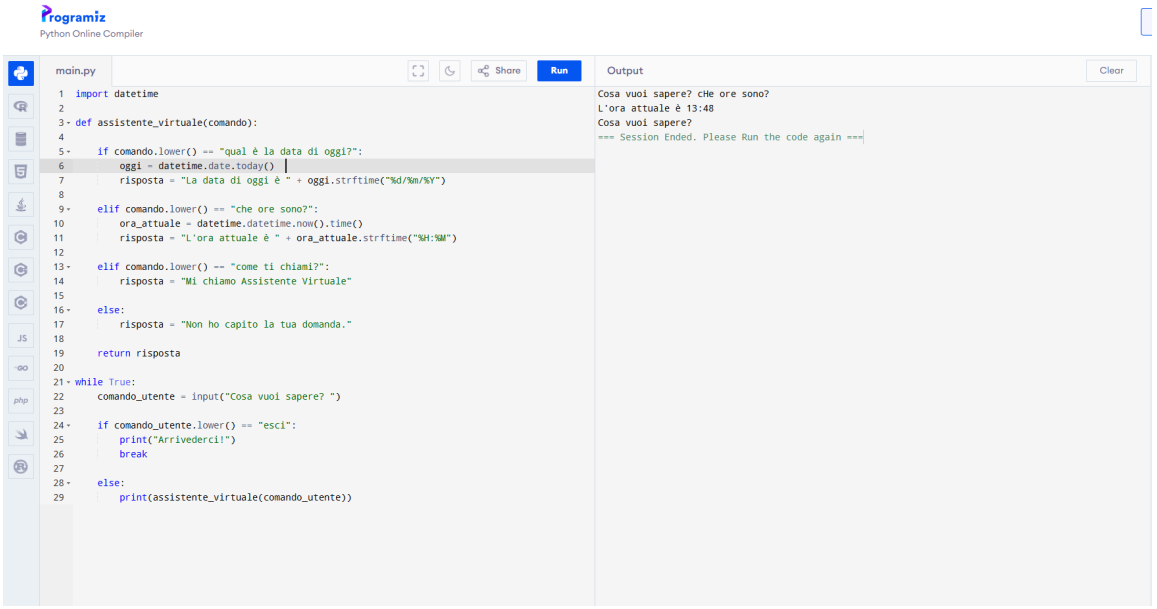
Codice Corretto

```
import datetime

def assistente_virtuale(comando):

    if comando.lower() == "qual è la data di oggi?":
        oggi = datetime.datetime.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

    elif comando.lower() == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H%M")

    elif comando.lower() == "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))
```

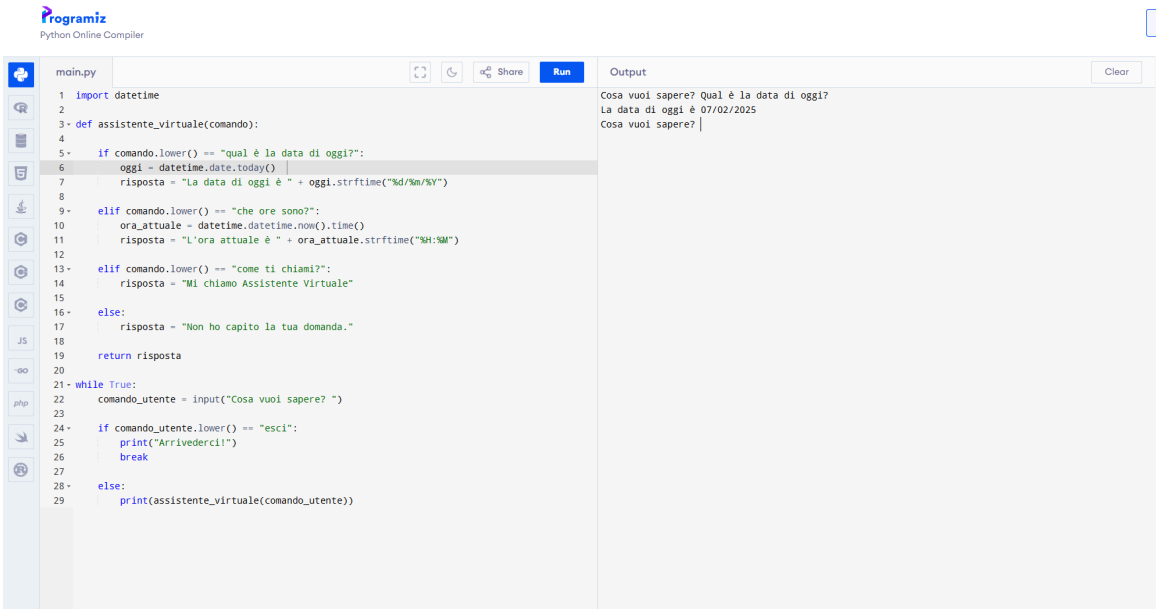
Prove

Il primo punto della traccia recita: Capire cosa fa il programma senza eseguirlo. Ciò vuol dire che una volta capito cosa fa nulla mi vieta di lanciare il programma e mettere delle prove per ampliare il mio scritto..

Prova minuscolo maiuscolo



Prove del funzionamento generale



Programiz
Python Online Compiler

main.py

```python
import datetime

def assistente_virtuale(comando):

    if comando.lower() == "qual è la data di oggi?":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

    elif comando.lower() == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")

    elif comando.lower() == "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))
```

Output

```
Cosa vuoi sapere? che ore sono?
L'ora attuale è 15:29
Cosa vuoi sapere?
```

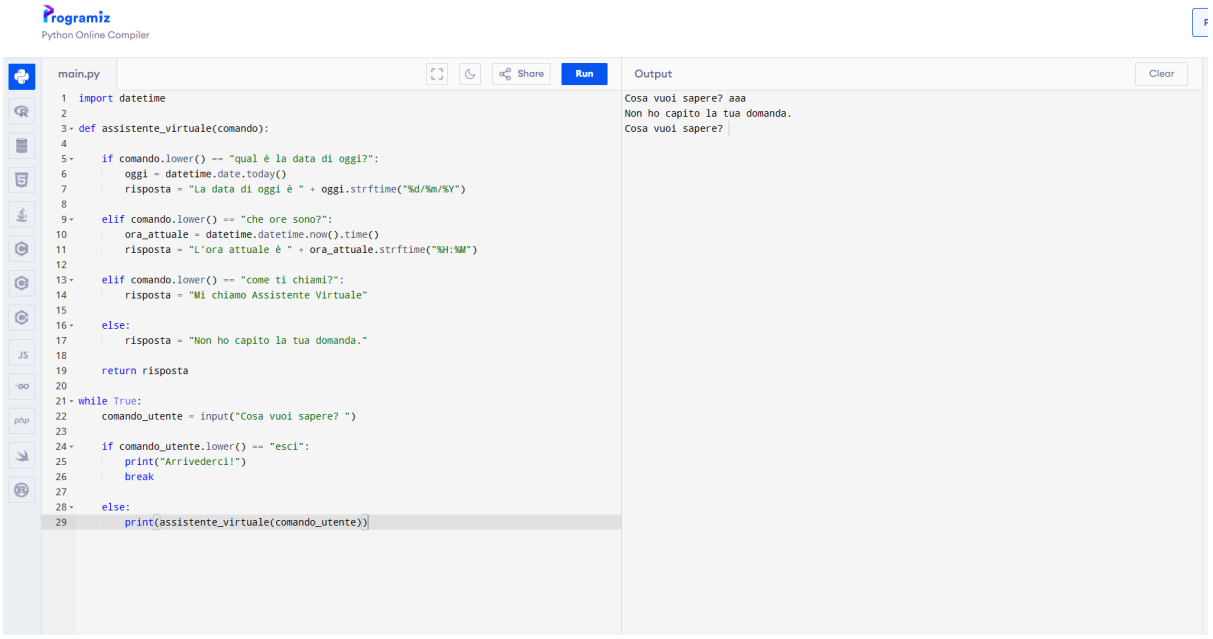
Programiz
Python Online Compiler

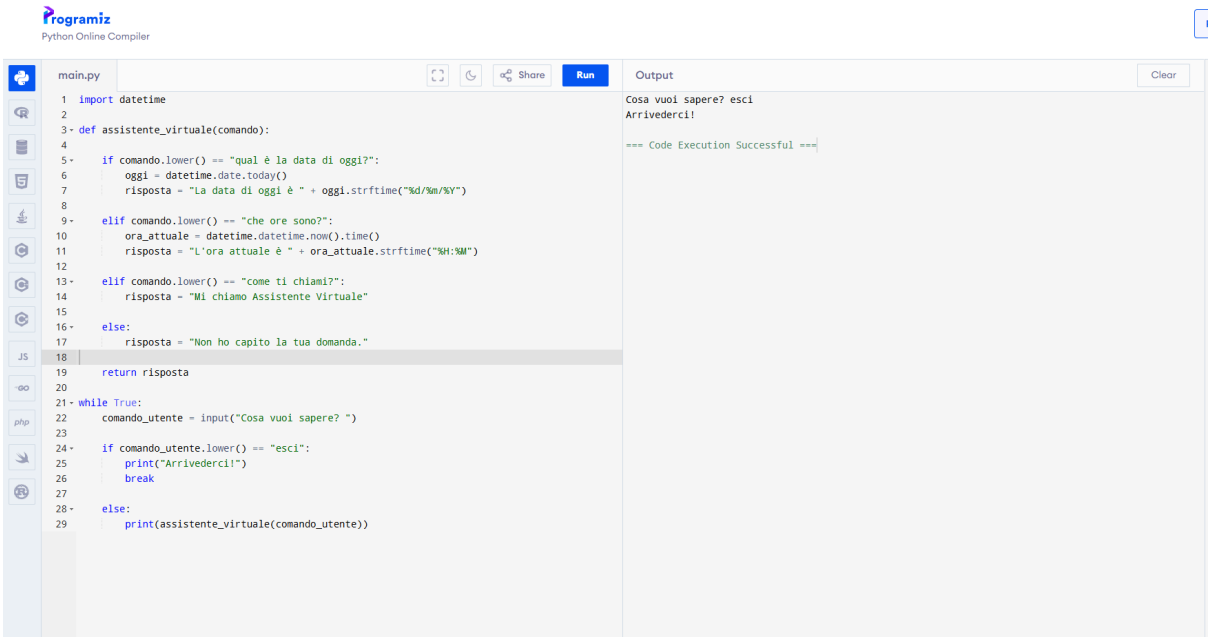
main.py

```python
import datetime

def assistente_virtuale(comando):

    if comando.lower() == "qual è la data di oggi?":
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

    elif comando.lower() == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()
        risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")

    elif comando.lower() == "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))
```

Output

```
Cosa vuoi sapere? Come ti chiami?
Mi chiamo Assistente Virtuale
Cosa vuoi sapere?
```





Conclusione

Nel corso di questa analisi, abbiamo esaminato passo per passo il codice dell'assistente virtuale, evidenziando alcuni errori comuni e proponendo le opportune correzioni. Il programma ora è in grado di rispondere correttamente alle domande sull'ora, la data e il nome dell'assistente virtuale, e permette di uscire quando l'utente lo desidera, migliorando l'interazione con l'utente finale. Con queste correzioni e miglioramenti, il programma è ora più sicuro, efficiente e pronto per un utilizzo.