Progetto S2/L5

Progetto assistente virtuale

Data la 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:

- Capire cosa fa il programma senza eseguirlo. Esercizio Progetto

- 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.

Codice fornito dalla traccia

```
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."
```

```
while  True
    comando_utente = input("Cosa vuoi sapere? ")
    if comando_utente.lower() == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

Sviluppo del primo punto

Capire cosa fa il programma senza eseguirlo

Come specificato anche dalla prima richiesta della traccia, possiamo capire l'obiettivo del programma anche senza eseguirlo:

Innanzitutto, possiamo notare che la prima cosa che fa è: "import datetime".

```
import datetime
def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        oggi = datetime.datetoday()
```

Grazie a questo possiamo immediatamente presupporre che vorrà avere una gestione migliore di informazioni quali data e ora.

Infatti, l'importazione della libreria "datetime" ci permette di analizzare, formattare e compiere operazioni aritmetiche sulle date.

Successivamente, vediamo che il codice va a definire la funzione dell'assistente virtuale: "def assistente_virtuale(comando)".

```
import datetime

def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        oggi = datetime.datetoday()
```

Come suggerisce anche il nome, andrà a configurare il nostro assistente virtuale, al quale potremmo fare delle domande, il tutto gestito grazie al parametro predefinito inserito tra le parentesi, ovvero, in questo caso, "comando".


```
import datetime

def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        oggi = datetime.datetoday()
```

A questo punto possiamo chiederci: come andiamo ad utilizzare questo parametro in questo contesto?

Possiamo utilizzare, per esempio, il costrutto ***if-elif-else*.**

Si tratta di istruzioni condizionali che vengono utilizzate quando vogliamo eseguire un blocco di codice in un determinato modo, a seconda appunto di certe condizioni.

Prendiamo ora il codice della traccia come esempio da analizzare:

```
    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
```

Il **funzionamento** è il seguente: abbiamo contemplato 3 domande, grazie ad **IF** ed **ELIF** riusciamo ad instradare

correttamente la domanda posta dall'utente in quelle già contemplate, e quindi ricevere una determinata risposta.

Qualora invece la domanda non fosse stata contemplata dal nostro costrutto, abbiamo l'opzione **ELSE,** il quale darà comunque un output all'utente grazie all'assegnazione risposta = "Non ho capito la tua domanda."

Per concludere, vediamo inserito alla fine della funzione "return risposta" per fare in modo che alla funzione venga riportata la risposta dell'assistente virtuale alle domande poste dall'utente.

Nell'ultimo blocco di codice riconosciamo un ciclo while True, ossia una struttura di controllo fondamentale che consente di eseguire un blocco di codice ripetutamente fintanto che una certa condizione è stata soddisfatta.

```
while  True
    comando_utente = input("Cosa vuoi sapere? ")
    if comando_utente.lower() == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

In questo caso specifico, come vediamo dall'immagine appena riportata, viene domandato all'utente che cosa gradirebbe sapere.

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

In questo modo viene rimandato alla funzione (vedi le 3 domande contemplate dal codice) eseguendo il loop all'infinito, finché non entra in gioco l'altra parte di codice che vediamo di seguito:

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

Come detto precedentemente grazie a quest'ultimo passaggio si può interrompere il loop dando all'utente la possibilità di scrivere "esci" (che in questo caso è case insensitive in quanto troviamo ".lower" prima).

In questo modo l'utente avrà comunque un output dal nostro codice che stamperà il saluto scelgo, in questo caso, "Arrivederci".

Sviluppo secondo punto

Individuare le casistiche non standard: comportamenti potenziali che non sono stati contemplati

Ho individuato principalmente una casistica non standard:

Dovremmo considerare il fatto che l'utente potrebbe digitare in diversi modi la domanda. Di conseguenza, rendere il comando case insensitive faciliterebbe l'inserimento della domanda rendendolo meno punitivo nel caso in cui l'utente non inserisse la domanda nell'esatto modo in cui è stata scritta nel codice.

Sviluppo terzo punto

Individuare eventuali errori di sintassi logici

Gli errori di sintassi che ho trovato sono i seguenti:

1. Se andassimo ad analizzare la 4 riga del codice, potremmo vedere un errore di sintassi nella parte di assegnazione della variabile oggi.

```
oggi = datetime.datetoday()
```

Come possiamo constatare, il modulo "datetime" è errato in quanto gli possono venir attribuiti solamente i metodi datetime e date. Possiamo quindi dire che datetime.datetoday è sbagliato.

2. Analizziamo ora il blocco logico composto dal ciclo while True. Come possiamo vedere dall'immagine, è presente un errore di sintassi:

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

Come possiamo notare manca il “:” dopo il True che serve a segnare i “confini” del blocco logico.

Gli errori logici che ho riscontrato è il seguente:

Trovo che l’utente al momento di fare la domanda non abbia un’idea chiara sul funzionamento dell’assistente vocale. Penso quindi sia poco intuitivo cosa possa o non possa andare a chiedere al nostro assistente vocale.

Sviluppo quarto punto

Proporre una soluzione per ognuno di essi

Per correggere il primo errore di sintassi è sufficiente andare a correggere il metodo in questo modo:

Da:

```
oggi = datetime.datetoday()
```

Ad:

```
oggi = datetime.date.today()
```

Ossia: oggi = datetime.date.today()

Per il secondo errore dobbiamo andare ad aggiungere il carattere “:” , allego immagini di dimostrazione di seguito:

Da:

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

Ad:

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

Il nostro codice finale apparirà come:

```
import datetime

def assistente_virtuale(comando):
        if comando== "Qual e` la data di oggi?":
                oggi = datetime.date.today()
                risposta = "La data di oggi e`" + oggi.strftime("%d/%m/%Y")
        elif comando == "Che ore sono?":
                ora_attuale = datetime.datetime.now().time()
                risposta = "L`ora attuale e`"+ 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))
```

Procediamo infine con la risoluzione dell'unico problema logico riscontrato. Andremo quindi ad aggiungere una parte iniziale per rendere più pratica e facile la parte di inserimento della domanda da parte dell'utente verso il nostro assistente virtuale.

Possiamo aiutare l’utente inserendo un messaggio di benvenuto con qualche linea guida, scrivendo il codice in questo modo:

print("Buongiorno! Di seguito una piccola guida sul funzionamento del nostro assistente virtuale:\n" "Digita una domanda per avere la tua risposta.\n" "Le domande disponibili sono:\n- Qual è la data di oggi?\n- Che ore sono?\n- Come ti chiami?\n" "- Digita “esci” per chiudere il programma.")