Progetto S2/L5

Consegna

Il progetto pratico di oggi ha lo scopo di allenare l'osservazione critica. Gli obiettivi della consegna sono di:

- Capire cosa fa il programma senza eseguirlo
- Individuare dal 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 da analizzare e correggere

```
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))
```

Analisi del codice dell'esercizio

Il codice è scritto in Python e importa un modulo chiamato "datetime". Ho consultato la documentazione ufficiale Python per assicurarmi del corretto uso dei metodi nel programma. Presenta una funzione principale chiamata "assistente_virtuale", che accetta un parametro "comando", che viene chiamata all'interno di un ciclo while. Il ciclo while, dopo aver ricevuto in input un comando dall'utente, attraverso uno statement if-else, verifica la volontà di utilizzare il programma o di uscire.

Scopo del codice

Il codice ha la funzione di creare un semplice "assistente virtuale" che sia in grado di ricevere un input dall'utente. In base alla stringa ricevuta, è in grado di fornire la data, l'orario al momento

dell'esecuzione e il proprio nome. Presenta inoltre semplici controlli sull'input che, ad esempio, permettono al programma di terminare l'esecuzione, solo se l'utente inserisce "esci", o di chiamare la funzione "assistente_virtuale" per ritornare la risposta e stamparla all'utente.

Potenziali comportamenti inattesi

Il codice (al netto degli errori di sintassi e logici)) prevede dei controlli sull'input fornito dall'utente, ma non comunica quali siano le sue funzionalità e quali comandi può immettere l'utente per interagire con l'assistente. Nel caso in cui l'utente non inserisca nessuna delle stringhe attese, l'else presente prima del comando return all'interno della funzione, comunica che il valore inserito non è stato "compreso" dal bot. Poiché non viene specificato all'utente il tipo di input che si aspetta il programma, l'assistente creato risponderà sempre "Non ho capito la tua domanda". Inoltre non vi è nessun tipo di gestione della stringa inserita, pertanto l'utente dovrà indovinare esattamente la stringa per permettere al codice di eseguire il rispettivo statement if-elif.
Dato che il programma non fornisce informazioni sul suo utilizzo, l'utente potrebbe non riuscire ad uscire dal programma correttamente, in quanto l'input atteso è la stringa "esci". Questo potrebbe potenzialmente portare il programma ad attendere il corretto input all'infinito.

Errori del codice

La consegna richiedeva di individuare gli errori di sintassi e logici del programma.
Gli errori di sintassi sono gli errori che si verificano quando il codice viola le regole di scrittura (sintassi) del linguaggio utilizzato. Ad esempio sono errori di sintassi la mancata presenza di ":" dopo uno statement, la mancanza di indentazione, la mancata chiusura o apertura di una parentesi.
Gli errori logici invece sono errori nella logica della programmazione e non comportano un errore durante l'interpretazione o la compilazione del codice. Esempi di errori logico sono: passaggio di parametri di un tipo di dato errato, dimenticanze di condizioni di controllo o di cicli, assegnazione di valori errati alle variabili.

- Il metodo **datetime.datetimetoday()** è sbagliato, il metodo corretto per ottenere la data è **datetime.date.today()**
- Il metodo **datetime.datetime.now().time()** è scorretto, il metodo corretto è **datetime.datetime.now()**
- Mancano i **:** dopo **while True**

Proposta soluzione del codice

L'immagine che segue riporta la mia proposta del programma aggiornato, ora il programma funziona correttamente, senza errori e permette all'utente di capire quale input fornire.
Ho evidenziato gli errori presenti nel codice originale ed ho apportato delle modifiche sia all'interno della funzione che nel ciclo while.

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

    elif comando == "Come ti chiami?":
        risposta = "Mi chiamo Assistente Virtuale"

    elif comando == "Arrivederci":
        risposta = "Arrivederci!"

    else:
        risposta = "Non ho capito la tua domanda."

    return risposta
```

Modifica del metodo in datetime.date.today()

Modifica del metodo in datetime.datetime.now()

```python
#Ho inserito una funzione print per spiegare all'utente le funzionalita' e i comandi
print("""** Benvenuto nell'assistente virtuale semplice. Sono in grado di fornirti la data di oggi, l'orario e di presentarmi! **\n
Inserisci: 1 se vuoi conoscere la data di oggi - 2 se vuoi sapere l'ora - 3 se vuoi sapere il mio nome - 4 per uscire\n""")

start = True #valore inmpostato a True avvia il ciclo while che viene terminato in caso di match case 4

while start == True:

    try:
        comando_utente = int(input("Cosa vuoi sapere? >>> ")) #Converto in intero l'input ricevuto

        match comando_utente: #Implementazione Python di uno switch-case, confronto con l'input castato
            case 1:
                comando_utente = "Qual è la data di oggi?"
            case 2:
                comando_utente = "Che ore sono?"
            case 3:
                comando_utente = "Come ti chiami?"
            case 4:
                comando_utente = "Arrivederci"
                start = False #Termino l'esecuzione del programma

        print(assistente_virtuale(comando_utente))

    except ValueError: #Se l'utente non inseerisce un numero, il cast in intero fallisce e controllo l'errore chiedendo un nuovo numero
        print("Inserisci un numero!")
```

Ho aggiunto i : dopo True

Ho semplificato l'interazione con l'utente. In questo modo si evitano rischi di errore di immissione delle stringhe e facilitano l'esperienza all'utente. Ho aggiunto una variabile start che viene impostata a False quando l'utente sceglie di uscire dal programma e fa si che non venga eseguito il while loop.

Infine ho gestito la possibilità di errore nel caso in cui l'utente non inserisca un numero come richiesto ma ad esempio un carattere o una stringa. In questo modo, il programma chiederà di fornire un nuovo input, senza terminare il programma con un errore.

L’immagine che segue riporta un esempio di esecuzione del codice, testando le diverse possibilità di input e di risposta dell’assistente la gestione degli errori.