Esercitazione S2/L5 **Contu Gianluca** **06/07/2024**

L’esercitazione di oggi prevedeva di analizzare con sguardo critico un codice sorgente e capirne diversi punti che tra poco andremo ad analizzare. Il principio dell’esercizio è quello di farci agire come un Hacker e pensare fuori dagli schemi.

Come dicevo, dato un codice sorgente dobbiamo analizzarlo e :

- Capire cosa fa il programma senza eseguirlo.

- Individuare nel codice sorgente le casistiche non standard che il programma non gestisce (esempio, c omportamenti potenziali che non sono stati contemplati)

- Individuare eventuali errori di sintassi / logici

- Proporre una soluzione per ognuno di essi.

Procediamo quindi con ordine. Per prima cosa legiamo il codice sorgente e capiamo che tipo di programma abbiamo davanti e cosa fa

```
import datetime

def assistente_virtuale(comando):

        if comando == "Quale 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))
```

Per avere una lettura più chiara, ho aperto Kali Linux su VM, ho creato il file ed ho trasferito il codice così come ci è stato fornito

Analizzando il codice possiamo vedere che il programma scritto qua sopra è un programma con funzione di assistenza, in cui l'utente può chiedere al programma 3 domande: Data, Ora e Nome dell'Assistente

```
if comando == "Quale 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"
```

Vediamo poi che nel caso l'utente non scriva in maniera corretta la domanda, il programma risponde con "Non ho capito la domanda".

E grazie al comando "While True", essendo un comando a loop, abbiamo la possibilità di fare un altra domanda.
E questa possibilità l'avrò all'infinito.

```
      else:

              risposta = "Non ho capito la tua domanda."

      return risposta

      while True
```

Infine il programma ci dice che inserendo la parola "esci" ci stamperà la parola "Arrivederci!" e di conseguenza, grazie al comando "Break" il programma si chiude

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

            print("Arrivederci!")

            break
```

Il programma entra in funzione grazie al comando:

-“comando_utente = input(“Cosa vuoi sapere?”)

Questo si ricollega alla funzione iniziale “def assistente_virtuale(comando)”. Quindi quando lo avvii, il programma stamperà la scritta “Cosa vuoi sapere?”

Bene ora che abbiamo visto e capito il codice che ci è stato fornito possiamo passare al secondo punto, ovvero individuare quali sono le casistiche non standard che il programma non gestisce.

Partendo dall’inizio notiamo subito delle casistiche nelle domande che la macchina non ci fornirà. Analizziamole una per volta.

Un problema è se l’utente non scrive la domanda con lettere maiuscole e minuscole nel modo in cui il programma la riesce a riconoscere.
In questo caso il programma risponderebbe con “Non ho capito la domanda”.

Nella parte riguardante l’orario, quindi l’utente che chiede che ore sono, notiamo anche che la risposta che stamperà, sarà solo in formato 24h, quindi se voglio che mi stampi l’ora nel formato 12h am/pm il programma non lo farà.

Passiamo ora al terzo punto, ovvero individuare errori di sintassi/logici.

Sono presenti due criticità nel codice che ci è stato fornito.

Il primo è un errore nel comando “datetime.datetoday”. Non esiste un metodo chiamato “datetoday” nel modulo “datetime”.

Il secondo non è un errore di sintassi ma di logica. Non è presente la funzione “\n” all’interno del comando:
“comando_utente = input(“Cosa vuoi sapere?”)”.
In questo caso, all’avvio del programma dopo che ci viene chiesto Cosa vuoi sapere, la risposta che scriveremo sarà attaccata alla domanda, senza andare a capo.

Passiamo ora all’ultimo punto, quindi offrire delle soluzioni rispetto agli errori e alle criticità trovate

Per quanto riguarda il problema delle maiuscole e minuscole nella formulazione della domanda aggiungiamo una nuova stringa per impostare questa opzione:

“comando = comando.lower()”

Per quanto riguarda il formato dell’orario possiamo aggiungere la possibilità di fare un altra domanda da parte dell’ultente, che quindi chiederà “che ore sono in formato 12h?”. Quindi in scrittura diventa

```
elif comando == “che ore sono in formato12h?”
   ora_attuale _ datetime.datetime.now().timw()
   risposta = “L’ora attuale è” + ora_attuale.strftime(“%I:%M:%p”)
```

Il comando errato “datetime.datetoday” diventa “datetime.date.today”

Nella stringa “comando_utente = input(“Cosa vuoi sapere?”)” inseriamo “\n”, quindi diventa:

“comando_utente = input(“Cosa vuoi sapere? \n”)”