

6/12/2024

# PROGETTO S2/L5

## 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:

- Capire cosa fa il programma senza eseguirlo.
- 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.

Questo è il codice che ci è stato fornito:

```
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?":

        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

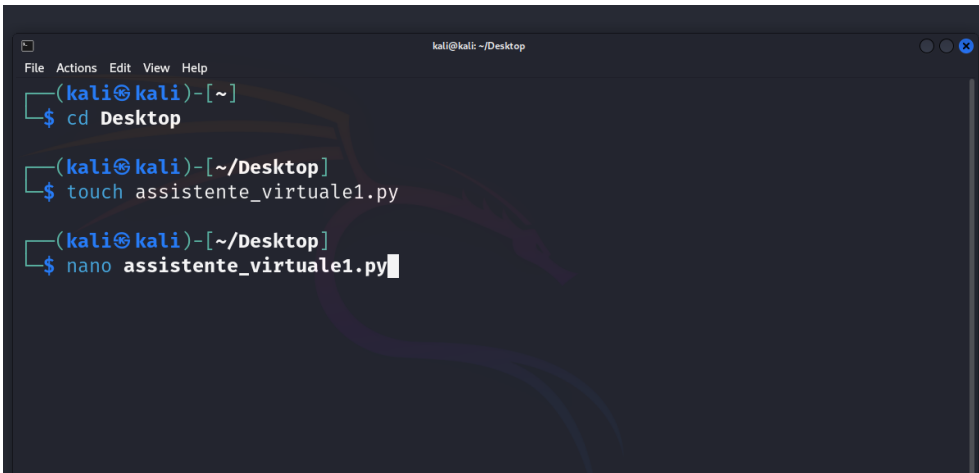
    else:

        print(assistente_virtuale(comando_utente))
```

Anche senza eseguirlo si può capire che il programma ci fornisce informazioni riguardo una data specifica (in questo caso è oggi) e l'orario attuale quando l'utente glie lo chiede. Il programma che si chiama

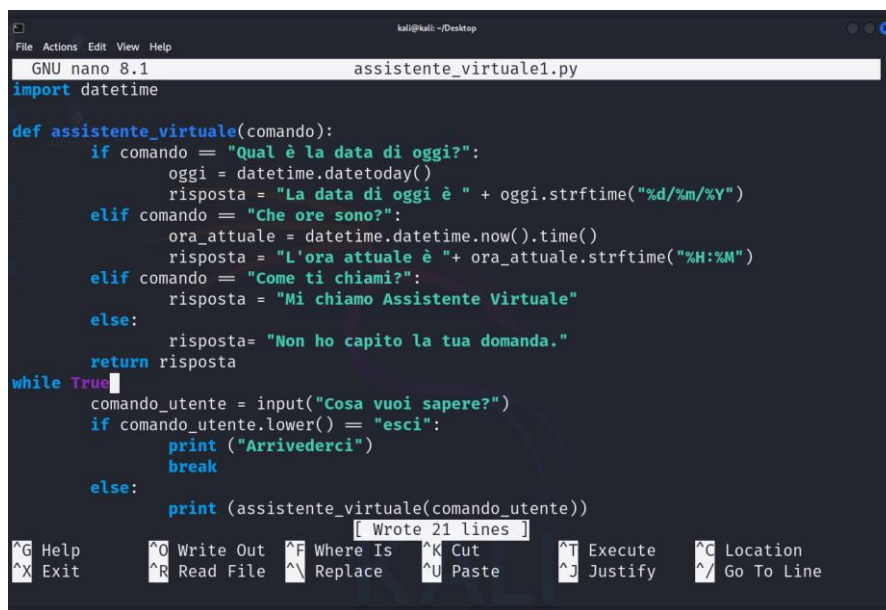
<assistente virtuale> ci fornisce anche il suo nome se richiesto, Esso ci darà le informazioni finché l'utente stesso deciderà di chiudere tramite il comando <esci>.

Per riscrivere il codice e poterlo analizzare meglio ho aperto il terminale su Linux, creato il file <assistente\_virtuale1.py> e aperto l'editor di testo:



```
kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)-[~]
$ cd Desktop
(kali@kali)-[~/Desktop]
$ touch assistente_virtuale1.py
(kali@kali)-[~/Desktop]
$ nano assistente_virtuale1.py
```

Ho copiato il codice:



```
GNU nano 8.1 assistente_virtuale1.py
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?":
        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))
```

Il secondo punto ci chiedeva di individuare nel codice sorgente le casistiche non standard che il programma non gestisce.

Possiamo notare come potrebbe mancare una funzione che migliori il riconoscimento dei comandi, indipendentemente dal fatto che siano scritti con lettere maiuscole, minuscole o senza segni di punteggiatura. Senza implementarla nel codice, se l'utente scrivesse: "che ore sono" (senza

maiuscolo e punto interrogativo) il programma non troverà corrispondenza e non ci darà la risposta che vogliamo ma ci dirà semplicemente; “non ho capito la domanda”.

Il terzo punto chiedeva di individuare eventuali errori di sintassi / logici.

Ho individuato un errore di sintassi nella riga 15, dopo <while True> c'è bisogno di mettere i due punti “:” -----> while True:

```
while True:
    comando_utente = input("Cosa vuoi sapere?")
    if comando_utente.lower() == "esci":
        print ("Arrivederci")
        break
    else:
        print (assistente_virtuale(comando_utente))
[ Wrote 21 lines ]
```

Anche nella riga 5 c'è un errore, il metodo corretto è:

oggi == datetime.date.today() non oggi == datetime.datetoday()

```
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")
```

Inoltre, nella riga 8 la funzione: datetime.datetime.now() viene utilizzata per ottenere sia la data che l'ora corrente, quindi, non è necessario inserire anche .time() poichè nella riga sotto viene specificato (%H:%M)

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

L'ultimo punto ci chiedeva infine di proporre una soluzione per gli errori o mancanze trovate nel codice. Sono partito aggiungendo la funzione che mi permette di migliorare il riconoscimento dei comandi dati dall'utente:

```
GNU nano 8.1 assistente_virtuale1.py
import datetime

def assistente_virtuale(comando):
    comando = comando.lower().strip().replace("?", "").replace(",", "").replace(".", "").replace("!", "")
    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

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

`comando.lower()` assicura che il comando inserito dall'utente sia trattato in modo uniforme indipendentemente da come è stato digitato (maiuscole o minuscole).

`.strip()` rimuove eventuali spazi aggiuntivi che potrebbero essere stati inseriti dall'utente prima o dopo il comando.

`.replace("?", "").replace(",", "").replace(".", "").replace("!", "")` rimuove tutti i punti interrogativi, virgole, punti esclamativi, punti e due punti dal comando dell'utente.

Poi ho modificato gli errori di sintassi e logici che ho spiegato prima:

```
GNU nano 8.1 assistente_virtuale1.py
import datetime

def assistente_virtuale(comando):
    comando = comando.lower().strip().replace("?", "").replace(",", "").replace(".", "").replace("!", "")
    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"
    else:
        risposta = "Non ho capito la tua domanda."
    return risposta

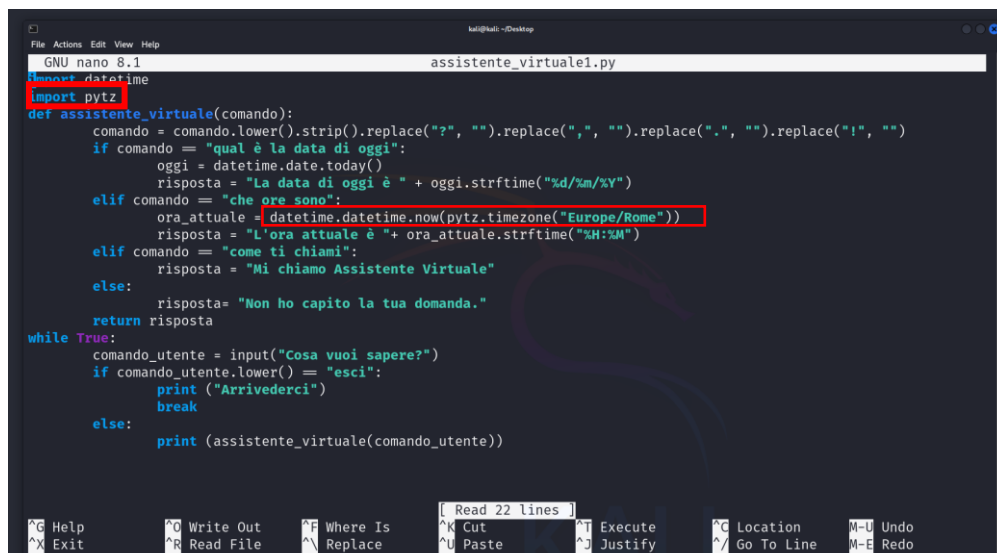
while True:
    comando_utente = input("Cosa vuoi sapere?")
    if comando_utente.lower() == "esci":
        print("Arrivederci")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

Ho avviato il programma per controllare se le modifiche funzionassero, sono partito col testare se maiuscole, minuscole e la punteggiatura non interferiscano con l'ottenimento delle risposte da parte del programma:

```
(kali@kali)-[~/Desktop]
$ python assistente_virtuale1.py
Cosa vuoi sapere?qual è LA DATA, di OGGI!
La data di oggi è 06/12/2024
Cosa vuoi sapere?qual è la data, di oggi?
La data di oggi è 06/12/2024
Cosa vuoi sapere?COME, TI chiami!
Mi chiamo Assistente Virtuale
Cosa vuoi sapere?Come, ti chIAMi!?
Mi chiamo Assistente Virtuale
Cosa vuoi sapere?Che ore sono
L'ora attuale è 07:13
Cosa vuoi sapere?che OrE, SoNO!
L'ora attuale è 07:13
Cosa vuoi sapere?esci
Arrivederci

(kali@kali)-[~/Desktop]
$
```

Come si può vedere il programma mi risponde anche se i miei comandi sono scritti in modo impreciso. Ho notato però che l'orario che mi dava era sbagliato, circa 6 ore indietro. L'unica cosa che mi è venuta in mente è stata quella di mettere un fuso orario di riferimento dal quale il programma potesse prendere l'ora esatta. Per questo ho aggiunto un modulo **pytz** che mi permette di attingere ad una libreria di tutti i fusi orari. Per ottenere il fuso orario ho passato l'oggetto **pytz** come parametro alla funzione **datetime.now**. L'oggetto **pytz** ha un attributo **timezone** che prende informazioni del fuso orario che sto cercando, in questo caso "Europa/Roma".



```
GNU nano 8.1 assistente_virtuale1.py
import datetime
import pytz
def assistente_virtuale(comando):
    comando = comando.lower().strip().replace("?", "").replace(", ", "").replace(".", "").replace("!", "")
    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(pytz.timezone("Europe/Rome"))
        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))
```

Infine, ho rilanciato il programma per verificare che l'orario sia finalmente corretto.

```
(kali@kali)-[~/Desktop]  
$ python assistente_virtuale1.py  
Cosa vuoi sapere?che ore sono  
L'ora attuale è 13:22  
Cosa vuoi sapere? 
```

\*sono passati circa 10mn tra questa verifica e lo screen della pagina precedente (risposta con orario 07:13)