

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 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.

Svolgimento

Codice sorgente:

```
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().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))
```

Anzitutto, **analisi il codice**, che è strutturato nel modo seguente:

- Importazione del modulo <<datetime>>, che fornisce classi riguardanti le date e gli orari.
 - Definizione della funzione <<assistente_virtuale>>, applicata all'input <<comando>>.
 - Struttura condizionale “if-elif-else”, che consente di eseguire parti del codice a seconda del valore di <<comando>>.
 - o Se il comando è <<Qual è la data di oggi?>>, si ricorre alla classe <<date>> e poi al metodo <<today>>, fornendo poi una risposta organizzata in “giorno/mese/anno”.
 - o E se il comando è <<Che ore sono?>>, si ricorre alla classe <<datetime>> e poi al metodo <<now>>, fornendo poi una risposta organizzata in “ore:minuti”.
 - o E se il comando è <<Come ti chiami?>>, la funzione risponde con il messaggio <<Mi chiamo Assistente Virtuale>>
 - o Altrimenti, se il comando non corrisponde a nessuna delle tre opzioni precedenti, la funzione risponde con il messaggio <<Non ho capito la tua domanda>>.
 - Il ciclo while esegue il blocco di codice fintanto che la condizione indicata è vera (come riportato dal <<True>>):
 - o Si richiede all'utente, con il messaggio <<Cosa vuoi sapere? >>, di inserire un input, che definirà <<comando_utente>>.
 - o Nel caso in cui il <<comando_utente>> inserito nell'input corrisponda a <<esci>>, si stamperebbe il messaggio <<Arrivederci!>>. Il successivo <<break>> interrompe il ciclo while. L'aggiunta del metodo lower, scrivendo <<comando_utente.lower()>> consente di considerare il messaggio in modo case-insensitive, senza distinzione tra caratteri minuscoli e maiuscoli.
 - o Invece, nel caso in cui la risposta dell'utente sia diversa da <<esci>>, si richiamerebbe la funzione <<assistente_virtuale>>, applicata al <<comando_utente>> inserito nell'input.
-

- **1. Capire cosa fa il programma senza eseguirlo**

Si tratta di un assistente virtuale che risponde ad alcune domande preimpostate (Qual è la data di oggi?; Che ore sono?; Come ti chiami?).

Nel momento in cui si avvia il programma, si chiede all'utente “Cosa vuoi sapere?” e in base al successivo input inserito dall'utente, se corrispondente a una delle tre opzioni previste, l'assistente virtuale risponde con la data di oggi, l'orario attuale e il nome dell'Assistente Virtuale.

Il programma continua a riproporre la stessa domanda all'utente, finché l'utente non inserirà l'input "esci", per interrompere il programma.

- **2. Individuare nel codice sorgente le casistiche non standard che il programma non gestisce.**

- Il programma considera l'input dell'utente in modo case-insensitive solamente per l'input <<esci>>, per interrompere il programma. Nel caso degli altri input (Qual è la data di oggi?; Che ore sono?; Come ti chiami?) non è stato usato il metodo lower, perciò il programma risponde con la risposta prevista solamente se l'utente inserisce il testo con le iniziali e le minuscole previste.

Soluzione: aggiungere una riga, sotto alla definizione della funzione <<def assistente_virtuale(comando)>>, con il metodo lower per convertire tutti i caratteri che verranno inseriti come <<comando>> dall'utente, nella struttura condizionale if-elif-else in minuscoli, e scrivere nel codice le tre domande/opzioni con caratteri minuscoli. Esempio:

```
def assistente_virtuale(comando):  
    comando = comando.lower()  
    if comando == "qual è la data di oggi?":
```

- L'utente non conosce le domande previste dal programma.

Soluzione: modificare il messaggio iniziale <<Cosa vuoi sapere? >> elencando le opzioni disponibili: <<Cosa vuoi sapere? (Qual è la data di oggi?/Che ore sono?/Come ti chiami?): >>

La soluzione prevederebbe però un messaggio piuttosto lungo, per cui si potrebbe semplificare con: <<Cosa vuoi sapere? Scrivi DATA o ORA o NOME: >>

In quest'ultimo caso, bisognerebbe modificare le domande previste nel codice, sostituendole con:

```
if comando == "data":  
    elif comando == "ora":  
    elif comando == "nome":
```

- L'utente non conosce il modo per interrompere il programma (ossia scrivere <<esci>>).

Soluzione: aggiungere al precedente messaggio anche questa opzione: <<Cosa vuoi sapere? Scrivi DATA o ORA o NOME o ESCI>>. Alternativa: <<Cosa vuoi sapere: DATA/ORA/NOME/ESCI>>.

- Nel caso in cui si mantenga la domanda <<Qual è la data di oggi?>>, è comune negli utenti l'errore ortografico <<Qual'è>>, che non verrebbe letto correttamente dal programma.

Soluzione: modificare il codice in modo da includere entrambe le opzioni:

```
if comando in ["qual è la data di oggi?", "qual'è la data di oggi"]:
```

In questo modo si introduce una lista, che prevede anche l'opzione "sbagliata", consentendo di fornire lo stesso una risposta all'utente.

- Nel caso in cui l'utente scriva aggiungendo spazi bianchi superflui all'inizio o alla fine del suo input, il programma non riuscirebbe a fornire la risposta.

Soluzione: utilizzare il metodo strip per rimuovere gli spazi bianchi all'inizio e alla fine.

```
def assistente_virtuale(comando):
```

```
    comando = comando.strip()
```

-
- **Individuare eventuali errori di sintassi/logici e proporre una soluzione per ognuno di essi.**

```
import datetime
```

```
def assistente_virtuale(comando):
```

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

```
        oggi = datetime.datetime.today() [Errore di sintassi, metodo sbagliato]
```

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

```
    elif comando == "Che ore sono?":
```

```
        ora_attuale = datetime.datetime.now().time() [Errore logico, metodo sbagliato]
```

```
        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 [errore di sintassi nel ciclo while]

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

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

    print("Arrivederci!")

    break

else:

    print(assistente_virtuale(comando_utente))
```

- **Proporre una soluzione per ognuno di essi.**

- **oggi = datetime.datetime.today()**
 - dal modulo <<datetime>> si chiama la classe <<date>> e da questa si chiama il metodo <<today>>, per cui la scrittura corretta è: <<**datetime.date.today()**>>.
 - È un errore di sintassi, perché <datetime>>, come scritto in origine, non corrisponde a un metodo del modulo <<datetime>>.
- **ora_attuale = datetime.datetime.now().time()**
 - dal modulo <<datetime>> si chiama la classe <<datetime>> e da questa si chiama il metodo <<now>>, per cui la scrittura corretta è: <<**datetime.datetime.now()**>>.
 - È un errore logico, perché si eseguirebbe lo stesso il programma.
- **while True**
 - mancano i due punti alla fine, la scrittura corretta è <<**while True:>>**>>.
 - È un errore di sintassi, poiché il ciclo while richiede di mettere i due punti affinché funzioni.

Pratica bonus S2/L5

Traccia: “Gestione di una lista della spesa”. Scrivi un programma Python per gestire una lista della spesa. Il programma deve permettere all’utente di:

- 1) Aggiungere un elemento alla lista.
- 2) Rimuovere un elemento dalla lista (se presente).
- 3) Visualizzare tutti gli elementi della lista ordinati in ordine alfabetico.
- 4) Salvare la lista su file.
- 5) Caricare una lista da file.

Il programma deve avere un menu che consente all’utente di scegliere le varie operazioni e deve terminare solo quando l’utente lo richiede.

Svolgimento

```
GNU nano 8.1                                programmalista1.py *
#Definisco la funzione menu_lista
def menu_lista(opzione_menu, lista):
    #La prima opzione è l'aggiunta di un elemento alla lista. Chiedo all'utente di scrivere l'elemento.
    # Uso il metodo append per aggiungere l'elemento alla lista. Stampo il messaggio di conferma.
    if opzione_menu == "1":
        merce = input("Scrivi l'elemento che vuoi aggiungere alla lista: ")
        lista.append(merce)
        print("L'elemento è stato aggiunto!")
    #La seconda opzione è la rimozione di un elemento. Chiedo all'utente di scrivere l'elemento.
    # Uso il metodo remove per rimuoverlo dalla lista. Stampo il messaggio di conferma (e prevedo messaggio se non è in lista).
    elif opzione_menu == "2":
        merce = input("Scrivi l'elemento che vuoi rimuovere dalla lista: ")
        if merce in lista:
            lista.remove(merce)
            print("L'elemento è stato rimosso!")
        else:
            print("L'elemento inserito non è nella lista!")
    #La terza opzione è la visualizzazione della lista. Uso il metodo sort per ordinare gli elementi in ordine alfabetico.
    #Stampo l'elenco, usando "for" per elencare tutti i "merce" in "lista".
    elif opzione_menu == "3":
        lista.sort()
        for merce in lista:
            print(merce)
    #La quarta opzione è il salvataggio della lista su un file. Faccio inserire all'utente il nome del file.
    # Uso il costrutto "with" per aprire il file in modalità scrittura (w) o per crearlo.
    # Con for scrivo ogni "merce" contenuta in "lista" all'interno del file. Con "\n" faccio scrivere gli elementi su righe separate.
    elif opzione_menu == "4":
        nome_del_file = input("Inserisci il nome del file su cui vuoi salvare la lista: ")
        with open(nome_del_file, "w") as file:
            for merce in lista:
                file.write(merce + "\n")
        print("La lista è stata salvata sul file ", nome_del_file)
#Definisco la funzione lista_spesa
def lista_spesa():
    #Creo una lista vuota
    lista = []
    #Avvio un ciclo while sempre vero, finché l'utente non scriverà l'opzione 6
    while True:
        print("*****")
        print("Menù di scelta")
        print("1) Aggiungere un elemento alla lista")
        print("2) Rimuovere un elemento dalla lista")
        print("3) Visualizzare gli elementi della lista")
        print("4) Salvare la lista su file")
```

```

#Definisco la funzione lista_spesa
def lista_spesa():
    #Creo una lista vuota
    lista = []
    #Avvio un ciclo while sempre vero, finché l'utente non scriverà l'opzione 6
    while True:
        print("*****")
        print("Menù di scelta")
        print("1) Aggiungere un elemento alla lista")
        print("2) Rimuovere un elemento dalla lista")
        print("3) Visualizzare gli elementi della lista")
        print("4) Salvare la lista su file")
        print("6) Chiudi il programma")
        print("*****")
        #faccio scegliere all'utente l'opzione
        opzione_menu = input("Quale operazione vuoi eseguire? Scrivere il numero corrispondente: ")
        #chiamo la funzione menu_lista nel caso in cui l'utente scelga 1, 2, 3 o 4. Se sceglie 6, interrompo il ciclo con un break.
        if opzione_menu in ["1", "2", "3", "4"]:
            menu_lista(opzione_menu, lista)
        elif opzione_menu == "6":
            print("Torna a trovarci!")
            break
        else:
            print("Hai inserito un'opzione non valida. Riprova!")
#Esecuzione del programma, con cui chiamo la funzione lista_spesa
if __name__ == "__main__":
    lista_spesa()

```

Verifica

- 1) Aggiungere un elemento alla lista.


```

File Actions Edit View Help
(kali@kali)-[~]
$ python programmalista1.py
*****
Menù di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 1
Scrivi l'elemento che vuoi aggiungere alla lista: pasta
L'elemento è stato aggiunto!
*****
Menù di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 1
Scrivi l'elemento che vuoi aggiungere alla lista: pane
L'elemento è stato aggiunto!
*****
Menù di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 1
Scrivi l'elemento che vuoi aggiungere alla lista: limoni
L'elemento è stato aggiunto!
*****
Menù di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente:

```


2) Rimuovere un elemento dalla lista

```
Menu di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 3
limoni
pane
pasta
*****
Menu di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 2
Scrivi l'elemento che vuoi rimuovere dalla lista: pane
L'elemento è stato rimosso!
*****
Menu di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 3
```



3) Visualizza gli elementi della lista in ordine alfabetico

```
*****
Menu di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 3
limoni
pane
pasta
*****
Menu di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 2
Scrivi l'elemento che vuoi rimuovere dalla lista: pane
L'elemento è stato rimosso!
```



4) Salva la lista su file

```
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 4
Inserisci il nome del file su cui vuoi salvare la lista: listadellaspesaprova
La lista è stata salvata sul file  listadellaspesaprova
*****
Menù di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 6
Torna a trovarci!
```

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

- Termina il programma solo quando l'utente lo richiede.

```
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 4
Inserisci il nome del file su cui vuoi salvare la lista: listadellaspesaprova
La lista è stata salvata sul file  listadellaspesaprova
*****
Menù di scelta
1) Aggiungere un elemento alla lista
2) Rimuovere un elemento dalla lista
3) Visualizzare gli elementi della lista
4) Salvare la lista su file
6) Chiudi il programma
*****
Quale operazione vuoi eseguire? Scrivere il numero corrispondente: 6
Torna a trovarci!
```

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