

ESERCIZIO S2_L5

Consegna:

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.*

```
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))
```

Analisi:

Iniziando a leggere il codice notiamo che il programmatore ha come prima cosa importato una libreria chiamata `datetime`. Come il nome suggerisce, tale libreria consente all'utente di gestire e manipolare i dati relativi a date ed orari.

Nella seconda riga notiamo che si procede a definire una funzione chiamata `assistente_virtuale()` al cui interno avrà un parametro denominato `comando`.

Il codice procede poi con una struttura condizionale *If* dove, nella fattispecie, si comanda al programma di eseguire determinate azioni qualora il parametro `comando` risultasse essere la stringa *"Qual è la data di oggi?"*.

Indentata all'interno dell'*if*, viene poi assegnata alla variabile chiamata `oggi` la funzione `datetime.datetime.today()`. Cercando però tra le librerie Python, non si trova alcuna corrispondenza con il nome di questa funzione; probabilmente l'autore voleva utilizzare la funzione `datetime.date.today()`.

Viene poi assegnato ad una variabile chiamata `risposta` quanto segue: `"La data di oggi è " + oggi.strftime("%d/%m/%Y")`.

Con tale assegnazione si vuole attribuire alla variabile `risposta` la frase *La data di oggi è* seguita dalla data del giorno (attribuita alla variabile `oggi`) con formato gg/mm/anno.

Il programma continua poi con un *elif* per andare a prevedere un valore diverso al parametro `comando`. Stavolta gli si attribuisce la stringa *"Che ore sono?"* ed i comandi da eseguire nel caso di riscontro della condizione sono:

- l'attribuzione dell'orario corrente tramite la funzione `datetime.datetime.now().time()` ad una variabile chiamata `ora_attuale`.
- la creazione della variabile `risposta` contenente *"L'ora attuale è "* + `ora_attuale.strftime("%H:%M")`. Tale dicitura, restituirà la frase *L'ora attuale è* seguita dal valore della variabile `ora_attuale` con formato HH:MM.

Abbiamo poi un ulteriore *elif* che ipotizza come valore per `comando` la stringa *"Come ti chiami?"* ed in questo caso viene creata la variabile `risposta` con valore *"Mi chiamo assistente virtuale"*.

Come ultimo componente della struttura *If* abbiamo dunque un *else* che entra in campo ogni qualvolta il parametro `comando` assuma valore diverso da quelli precedentemente indicati.

In questo caso il programma genererà una variabile `risposta` contenente la stringa *"Non ho capito la tua domanda."*

Ci viene poi presentato un ciclo *while True* al cui interno viene creata una variabile chiamata `comando_utente` a cui viene attribuito il valore di un input immesso dall'utente in risposta alla domanda *Cosa vuoi sapere?*.

Si procede nuovamente con una struttura condizionale *if* avente come protagonista la variabile appena creata. Si ipotizza che l'input dell'utente il lower case, tramite la funzione `.lower()`, sia *esci* ed in quel caso si istruisce il programma ad eseguire il seguente messaggio di output tramite terminale: *Arrivederci!*.

Si procede poi a terminare il ciclo *while* tramite un *break*.

Al termine del costrutto condizionale, il programma ritorna la variabile `risposta` tramite *return*, rendendone disponibile il valore qualora la si volesse stampare in output nel punto in cui viene eseguita la funzione.

Come alternativa a qualsiasi altro valore immesso dall'utente si prevede invece che il programma vada ad eseguire la funzione `assistente_virtuale()` spiegata in precedenza e a stamparne l'output utilizzando come parametro `comando` il valore contenuto all'interno della variabile `comando_utente`. Il tutto tramite il comando `print(assistente_virtuale(comando_utente))`.

Da quanto dedotto tramite un'analisi statica dell'algoritmo possiamo dichiarare quanto segue:

Eccetto per l'errore relativo a `datetime.datetime.today()`, il codice sembra essere scritto con correttezza sintattica e dovrebbe poter essere eseguito senza problemi.

Vi sono tuttavia alcune migliorie da poter implementare per rendere l'algoritmo più funzionale e fruibile.

La prima che salta all'occhio consiste sicuramente nel poter analizzare la richiesta dell'utente indipendentemente dall'upper o lower case dei caratteri inseriti ed eventualmente errori di spaziature extra. A tal riguardo proporrei dunque l'inserimento nei costrutti *if* della funzione `.lower()` per verificare il valore dell'input dell'utente senza tener conto del fatto che abbia utilizzato lettere maiuscole o minuscole e l'utilizzo di `.strip()` per rimuovere eventuali spazi non richiesti dal programma:

```
if comando.lower().strip() == "qual è la data di oggi?"  
elif comando.lower().strip() == "che ore sono?"  
elif comando.lower().strip() == "come ti chiami?"
```

Un altro punto importante, essendo questo codice quello che sembra essere il prototipo grezzo di un assistente virtuale, consiste nel far sapere all'utente, tramite funzione `print()`, che cosa aspettarsi dal programma che sta avviando

e quali sono le domande a cui attualmente l'algoritmo è in grado di rispondere in maniera soddisfacente.

Ecco un esempio di codice a cui ho pensato che andrà inserito a seguito di return **risposta** (a termine della prima slide di codice):

```
print("Ciao! Sono il nuovo prototipo di un assistente virtuale!")
print("Al momento posso solo assisterti con i seguenti semplici quesiti:")
print("Che ore sono?")
print("Qual è la data di oggi?")
print("Come ti chiami?")
while True:
    volontà = input("Ti va di provare a chiedermi una di queste cose? ")
    if volontà.lower().strip() == "si" or volontà.lower().strip() == "sì":
        print("Fantastico!")
```

CICLO WHILE DELLA SECONDA SLIDE DI CODICE

```
elif volontà.lower().strip() == "no":
    print("Peccato! Sarà per la prossima! :)")
    exit()
else:
    print("Mi spiace ma non comprendo quanto inserito.")
    print("Rispondere con si oppure no").
```

Conclusione:

In sintesi l'algoritmo si presenta come la bozza di un software di assistente virtuale che al momento è in grado solamente di ricevere come input tre domande predefinite e di rispondere ad esse in maniera coerente.

Il programma, se considerato come una bozza o prototipo, risulta essere ben concepito nella sua struttura generale. Presenta un unico errore di sintassi (`datetime.datetime.today()` anziché `datetime.date.today()`), e nel complesso appare coerente e funzionale.

Implementando le migliorie suggerite, tra cui l'uso di `.lower()` e `.strip()` per normalizzare l'input e un'introduzione iniziale per guidare l'utente, il programma risulterebbe più intuitivo, robusto e user-friendly.

Ulteriori punti di debolezza o opportunità di miglioramento potrebbero emergere attraverso un'analisi dinamica del codice, ovvero osservandone il comportamento effettivo durante l'interazione con l'utente e testando casi meno prevedibili o non contemplati in fase di progettazione.