

PROGETTO SOFTWARE

1. RICHIESTE

- 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.

2. ESEMPIO E SOLUZIONI

```
1 import datetime
2
3 while True:
4     comando_utente = input("Cosa vuoi sapere? ")
5     if comando_utente == "esci":
6         print("Arrivederci!")
7         break
8     else:
9         print(assistente_virtuale(comando_utente))
10
11 def assistente_virtuale(comando):
12     if comando == "Qual è la data di oggi?":
13         oggi = datetime.datetime.today()
14         risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
15     elif comando == "Che ore sono?":
16         ora_attuale = datetime.datetime.now().time()
17         risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M")
18     elif comando == "Come ti chiami?":
19         risposta = "Mi chiamo Assistente Virtuale"
20     else:
21         risposta = "Non ho capito la tua domanda."
22     return risposta
23
```

2.1 FUNZIONALITÀ PROGRAMMA

Il programma definisce principalmente un'assistente virtuale a cui l'utente può rivolgersi per diverse richieste.

Nell'esempio gli input da stampare possono essere la **data corrente** (in formato gg/mm/aa), **l'ora attuale** (in formato h/m) e il **nome dell'assistente**.

2.2 INDIVIDUAZIONE COMPORTAMENTI NON CONTEMPLATI

1. Nel codice, ci può essere un'ipotesi dove l'utente scriva i comandi in **Uppercase**, ma dato che non è stato previsto dal programmatore, al momento dell'esecuzione **non stamperà le risposte**.
2. Un'altra ipotesi è che l'utente al momento dell'esecuzione **non sa gli input prestabiliti dal programmatore** data la mancanza di un'**interfaccia** che dà all'utente diversi comandi corrispondenti alle richieste.

2.3 ERRORI SINTASSI/LOGICI

- Il primo errore:mancanza dei (":") nel ciclo **While(riga 3)** quindi quel blocco di codice **non verrà ripetuta** generando un errore.
- Il secondo errore: inserimento sbagliato del **break, (riga 7)**, non si trova nell'**if** quindi anche se l'utente non scrive ("esci") il ciclo verrà lo stesso interrotto.
- Il terzo errore: definire la funzione(**assistente_virtuale**)(**riga 9**) dopo che è stata richiamata.
Come sappiamo, Phyton legge il codice dall'**alto verso il basso** quindi se la funzione viene richiamata prima del **def**, ci darà un errore di **funzione non definita**.
- Il quarto errore: comando errato (**datetime.datetoday**) (**riga 13**) quindi python, non riuscirà a trovare (.datetoday) nel modulo (**datetime**), generando un errore.

2.4 SOLUZIONI CASISTICHE

1. Problema uppercase

```
def assistente_virtuale(comando):
    comando = comando.lower()
    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,scegli tra queste domande"
    return risposta
```

In questo modo con l'aggiunta del **.lower()**, anche se l'utente inserirà tutto in **uppercase** nell'input, il programma lo convertirà automaticamente in minuscolo e stamperà il risultato desiderato

2. Problema interfaccia input

```
while True:  
    print("")  
    - 1: >>> Qual è la data di oggi?  
    - 2: >>> Che ore sono?  
    - 3: >>> Come ti chiami?  
    "")  
    comando_utente = input("Cosa vuoi sapere? >>> ").lower()  
  
    if comando_utente == "esci":  
        print("Arrivederci!")  
        break  
    else:  
        print(assistente_virtuale(comando_utente))
```

*In questo modo ho creato un **menù a tendina** dove farà vedere all'utente quali sono gli input per avere la risposta desiderata

2.5 SOLUZIONI ERRORI

1. Errore (“:”) ciclo while

```
while True:  
    comando_utente = input("Cosa vuoi sapere? ")  
    if comando_utente == "esci":  
        print("Arrivederci!")
```

*Aggiungendo i (“：“) si otterrà la ripetizione del blocco di codice

2. Errore posizionamento break

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

*Trovandosi nell' **if** e al momento dell'esecuzione l'utente non scrive(“esci”), il ciclo non verrà interrotto.

3. Errore richiamo funzione(**assistente_virtuale**)

```
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 == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

*Inserendo prima la funzione(**assistente_virtuale**), arriverà al momento del richiamo in (**riga 24**) senza mostrare un errore

4. errore comando(**datetime.date.today()**)

```
if comando == "Qual è la data di oggi?":
    oggi = datetime.date.today()
    risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
```

*Inserendo **datetime.date.today()** non darà errore e stamperà la data attuale