

Analisi statica codice

Kevin motti.

esercizio:

Analizzare il codice fornito (senza eseguirlo) e trovare all'interno di esso:

- casistiche non standard
- Individuare errori di sintassi/logici
- Proporre una soluzione per ognuno di essi

```
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
```

Comprensione del programma

Lo script fornito serve per la creazione di un programma che in base all'input dell'utente può fornire 3 informazioni:

- La data attuale
- L'ora attuale
- Il nome del programma stesso

Errori logici e di sintassi

1. Inversione della posizione tra funzione e ciclo while

Come prima cosa la definizione della funzione è buona prassi metterla all'inizio dello script e non all'interno del blocco di un ciclo "while".

Nella riga **13** dell'esempio appare un errore di sintassi in "`datetime.datetoday()`" la sintassi corretta sarebbe: "`datetime.date.today()`".

Nella riga **14** appare un errore in ("`%d%m%y`") dove la sintassi corretta della formattazione dovrebbe essere: ("`%d,%m,%y`") separata da virgole.

Nella riga **17** appare un altro errore di sintassi nella formattazione che dovrebbe essere separate da virgole es: ("`%H,%M`") non ("`%H:%M`")

2. Miglioramento funzione

Prima di iniziare il flusso di condizionali inserire una variabile con concatenazione di metodi es: "`comando_pulito = comando.lower().strip()`"

In modo da poter ovviare alla natura "case sensitive" di python nel caso l'utente inserisca maiuscole o spazi dove non dovrebbero essere.

3. Ciclo while (errori logici e di sintassi)

Il primo errore di sintassi che salta all'occhio è la mancanza dei due punti alla fine della condizione (riga 3), la sintassi corretta sarebbe " while True: " dato che i due punti indicano la fine della definizione del ciclo e aprono la porta al codice indentato che apparterrà a quel blocco.

4. miglioramenti

Per facilitare l'uso del programma a un utente non profondamente a conoscenza delle sue funzioni, nell "input" della variabile "comando_utente" inserirei il nome delle 3 funzioni principali che il programma svolge e qual è il metodo per uscire.

Io lo scriverei così: `comando_utente = input("chiedeimi l'ora, la data o il mio nome:\n(digita esci per chiudere): ")`

Nella condizione "if" della variabile "comando_utente" inserirei un'altra concatenazione di metodi per avviare a maiuscole e indentazioni mal inserite, lo scriverei così:

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