

## REPORT S2/L5

---

Nel esercizio di oggi è richiesto di analizzare criticamente il codice con lo scopo di capire cosa fa il programma senza eseguirlo, individuare dal codice sorgente le casistiche non standard che non vengono gestite (comportamenti non contemplati), individuare eventuali errori sintassi/logici e proporre una soluzione per ciascuno di questi.

```
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 CODICE

---

Il codice preso in esame definisce una funzione **assistente\_virtuale()** al quale viene passato un parametro in ingresso “comando” con lo scopo di gestire alcune domande. La funzione è in grado di rispondere a tre tipi di domande diverse riguardanti la data, l’orario e il nome dell’assistente stesso, il tutto è possibile grazie al modulo `datetime`, importato ad inizio codice, il quale fornisce delle classi per la gestione della data e dell’ora. Nella parte successiva del codice all’interno di un loop (**while == true**) viene richiesto all’utente di porre una domanda alla macchina, nel caso in cui l’utente inserisca la stringa “esci” il programma termina **break** altrimenti viene richiamata la funzione definita precedentemente per rispondere alle richieste mediante una serie di **if-elif**.

## ANALISI ERRORI

Dopo aver eseguito il codice andiamo a correggere gli eventuali errori individuati:

- **Errore1:** Errore di sintassi, non è presente ":" dopo il **while**.

```
$ python progetto.py
File "/home/kali/Desktop/python/progetto.py", line 16
    while True
      ^
SyntaxError: expected ':'
```

- **Errore 2:** Un errore a runtime nel momento in cui si richiede la data, non esiste l'attributo **datetoday** del modulo **datetime** ma esiste **date.today()**.

```
oggi = datetime.datetoday()
^^^^^^^^^^^^^^^^^^^^^^^^
AttributeError: module 'datetime' has no attribute 'datetoday'
```

- **Errore3:** Nell'eventualità in cui il parametro passato alla funzione non venga riconosciuto non c'è nessuna definizione della variabile **risposta** se non nel **return**.
- **Errore4:** Nonostante il metodo presente nel codice **datetime.datetime.now().time()** produca effettivamente un risultato sarebbe sufficiente il metodo **.datetime.now()**

```
from datetime import datetime

current_dateTime = datetime.now()

print(current_dateTime)
# 2022-09-20 10:27:21.240752
```

<https://www.freecodecamp.org/news/python-datetime-now-how-to-get-todays-date-and-time/>

## GESTIONE CASI PARTICOLARI

Andiamo a capire se sia possibile gestire meglio eventuali casi particolari. Analizzando il codice, da una prima occhiata è evidente che in primo luogo manca un'indicazione data all'utente su quali siano gli input che possano effettivamente generare una risposta che non sia **"non ho capito la domanda"**, per risolvere questo problema si potrebbe immaginare di inserire per esempio ad inizio codice una **print()** che indichi a quali domande la funzione è in grado di rispondere. Un altro caso che può essere gestito è quello di andare a convertire in **lowercase** tutte le stringhe date in input in maniera tale da garantire una maggiore possibilità di matchare le domande a cui la funzione sa rispondere per farlo all'inizio della funzione posso inserire la riga **comando = comando.lower()** facendo lo stesso con le stringhe nei controlli negli **if**. A questo si può aggiungere quello di andare ad escludere l'eventualità di possibili errori di battitura fatti dall'utente in fase di inserimento dell'input, per farlo possiamo aggiungere una gestione più flessibile degli input dell'utente, escludendo la punteggiatura. Aggiungo al comando visto precedentemente il metodo **replace** sostituendo il "?", **.replace("?", "")**.

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

Posso poi aggiungere nell'if la riga: **if comando in ["qual è la data di oggi", "che giorno è"]**, in modo da rendere il controllo sull'input più versatile, ampliando la possibilità di matching dell'input.

```
if comando in ["qual è la data di oggi", "che giorno è"]:
    oggi = datetime.date.today()
    risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")
```

Andiamo quindi a vedere il nuovo codice corretto e provandolo in esecuzione con alcuni input per verificarne il funzionamento, in rosso sono evidenziate le aggiunte viste precedentemente.

```
kali@kali: ~/Desktop/python
File Actions Edit View Help
GNU nano 8.1 progetto.py

import datetime
def assistente_virtuale(comando):
    comando = comando.replace("?", "").lower()

    if comando in ["qual è la data di oggi", "che giorno è"]:
        oggi = datetime.date.today()
        risposta = "La data di oggi è " + oggi.strftime("%d/%m/%Y")

    elif comando in ["che ore sono", "che ora è"]:
        ora_attuale = datetime.datetime.now()
        risposta = "l'ora attuale è " + ora_attuale.strftime("%H:%M")

    elif comando in ["come ti chiami", "qual è il tuo nome"]:
        risposta = "Mi chiamo Assistente Virtuale"

    else:
        risposta = "Non ho capito la tua domanda?"
    return risposta

print("Il programma è in grado di rispondere alle richieste di orario, data e nome\n")
while True:
    comando_utente = input("Cosa vuoi sapere? ")
    if comando_utente.lower() == "esci":
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))
```

```
kali@kali: ~/Desktop/python
File Actions Edit View Help

(kali@kali)-[~/Desktop/python]
$ python progetto.py
Il programma è in grado di rispondere alle richieste di orario, data e nome

Cosa vuoi sapere? che giorno è?
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? che ora è
l'ora attuale è 11:21
Cosa vuoi sapere? come stai??
Non ho capito la tua domanda?
Cosa vuoi sapere? |
```