Report Esercizio 06/12/2024

Programma Data/Ora Linguaggio Python Leonardo Catalano

“La traccia di oggi ci chiedeva di fare un lavoro “inverso” di osservazione, analizzazione modifica e correzione di un programma già esistente.
I punti da seguire dato il codice erano i seguenti:
-Capire cosa fa il programma senza eseguirlo.
-Individuare dal codice le casistiche non standard che il programma non gestisce (comportamenti analoghi dell’utente).
-Individuare eventuali errori di sintassi / logici.
-Proporre una soluzione modificando il codice per ognuno di essi.

Allora prima di iniziare a spiegare il mio ragionamento di analisi lascio in foto il codice sorgente.

```
import datetime

def assistente_virtuale(comando):
    if comando == "Qual è la data di oggi?":
        oggi = datetime.datetoday()
        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))
```

Il 1* punto, dice di capire cosa fa il programma senza eseguirlo.
Premetto che inizialmente non sapevo nello specifico come funzionasse il modulo datetime, ma leggendo il codice è abbastanza autoesplicativo.
Partendo dalla funzione assistente virtuale(comando) abbiamo che ha come argomento il comando della scelta dell'utente, e cominciano i vari if elif else, delle scelte che l'utente può fare e a seconda della scelta si va a far uscire in output la data, l'ora attuale, o il nome dell'assistente virtuale, else: l'utente ha scritto qualcosa di sbagliato e si dà un output dove si dice ciò.
Nel while abbiamo l'input dell'utente dove gli si chiede cosa vuole sapere, se il comando è esci allora si esce dal ciclo e il programma di interrompe, senò si fa la chiamata alla funzione e inizia la porzione di codice descritta precedentemente.

2* Punto, individua le casistiche non standard che il programma non gestisce.
Leggendo il programma cosi com'è prima di parlare degli errori di scrittura/gestione.
Abbiamo un problema alla radice perchè cosi' com'è il codice l'utente non ha idea di cosa deve fare, dirgli solo cosa vuoi sapere e non dargli le opzioni comporta al fatto che

il programma non gestisce nemmeno il caso standard che l'utente inserisce correttamente la richiesta.
A seguito di ciò non viene gestito nell'input, il caso in cui l'utente inserisca la richiesta con maiuscole, minuscole varie, viene gestito soltanto nell'if all'interno del while per vedere se la stringa è == "esci", ma poi se non è cosi la stringa dell'utente rimane invariata, ciò comporta che se l'utente dovesse scrivere "qual' è LA data DI Oggi?: nella funzione questa stringa non verrebbe riconosciuta perchè non corrisponde con esattezza a "Qual'è la data di oggi?", perchè è scritta con una struttura di maiuscole/minuscole differenti anche se le parole sono le stesse.
Con tutto ciò si finisce in un loop tornando al primo problema alla radice perchè il programma continuerebbe a chiedere input e l'unico modo per fermare il programma è che l'utente scriva esci, ma dal momento che non ha idea nemmeno di che faccia il programma perchè l'unica cosa che gli appare è Cosa vuoi sapere? Il programma rimarrà in un loop.

3* Punto, individua gli errori sintattici e logici.
Allora sono andato a vedermi come funziona il modulo datetime per correggere pure la funzione e ho visto che ci sono degli errori di scrittura sintattici nel codice:
nella 4* riga oggi= datetime.datetoday() è sbagliata la formula corretta è datetime.datetime.today() (modulo-classe-metodo) ciò permette di restituire la data.
Non è puramente un errore sintattico perchè funziona lo stesso, però se si vuole sapere l'ora esatta generalmente si intendono anche i secondi quindi nella riga:
risposta = "L'ora attuale è " + ora_attuale.strftime("%H: %M"), sarebbe corretto metterci anche i secondi ("%H: %M: %S") .
Infine nel while True mancano i (duepunti) : .

Per gli errori logici invece abbiamo l'errore per eccellenza generale del programma che non dice all'utente cosa fa il programma stesso, quindi cosi' comè al 100% a meno che per magia, il programma non funzionerà logicamente mai, perchè l'utente non vede il codice del programma ma solo la sua esecuzione.
Quindi bisogna modificare la fase input dell'utente e dirgli quali sono le scelte disponibili e quando non vuole più fare nessuna scelta di scrivere esci per terminare il programma.
Ora fatta questa modifica i problemi non sono finiti, perchè bisogna gestire meglio sia ancora la fase di input che quella dei controlli nella funzione, se l'utente dovesse scrivere per come è ora il programma in maniera differente le scelte che può fare con una lettera maiuscola/minuscola diverse il programma non funziona si finirà con avere la risposta "Non ho capito la tua domanda".
Quindi bisogna aggiungere un .lower() nella fase di input e cambiare anche nella fase dei controlli le stringhe "giuste" invece di "Qual'è la data di oggi?", bisogna mettere "qual'è la data di oggi?", cosi' aldilà se l'utente scrive maiuscolo, minuscolo non

cambia funziona, le parole sono quelle e si va a restituire l'output corretto.

4* Punto Proporre, una soluzione modificando il programma per ogni problema

Innanzitutto risolviamo i problemi sintattici, quindi si vanno a modificare quelle parti di codici scritte con una sintattica errata :
(oggi = datetime.datetoday() --> oggi = datetime.datetime.today()).
Non è proprio un errore ma per buona norma nel tempo si dovrebbero anche aggiungere i secondi:
risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M") --->
risposta = "L'ora attuale è " + ora_attuale.strftime("%H:%M: %S")
Il ciclo while è scritto in maniera errata mancano i (duepunti) :
while True --> while True :

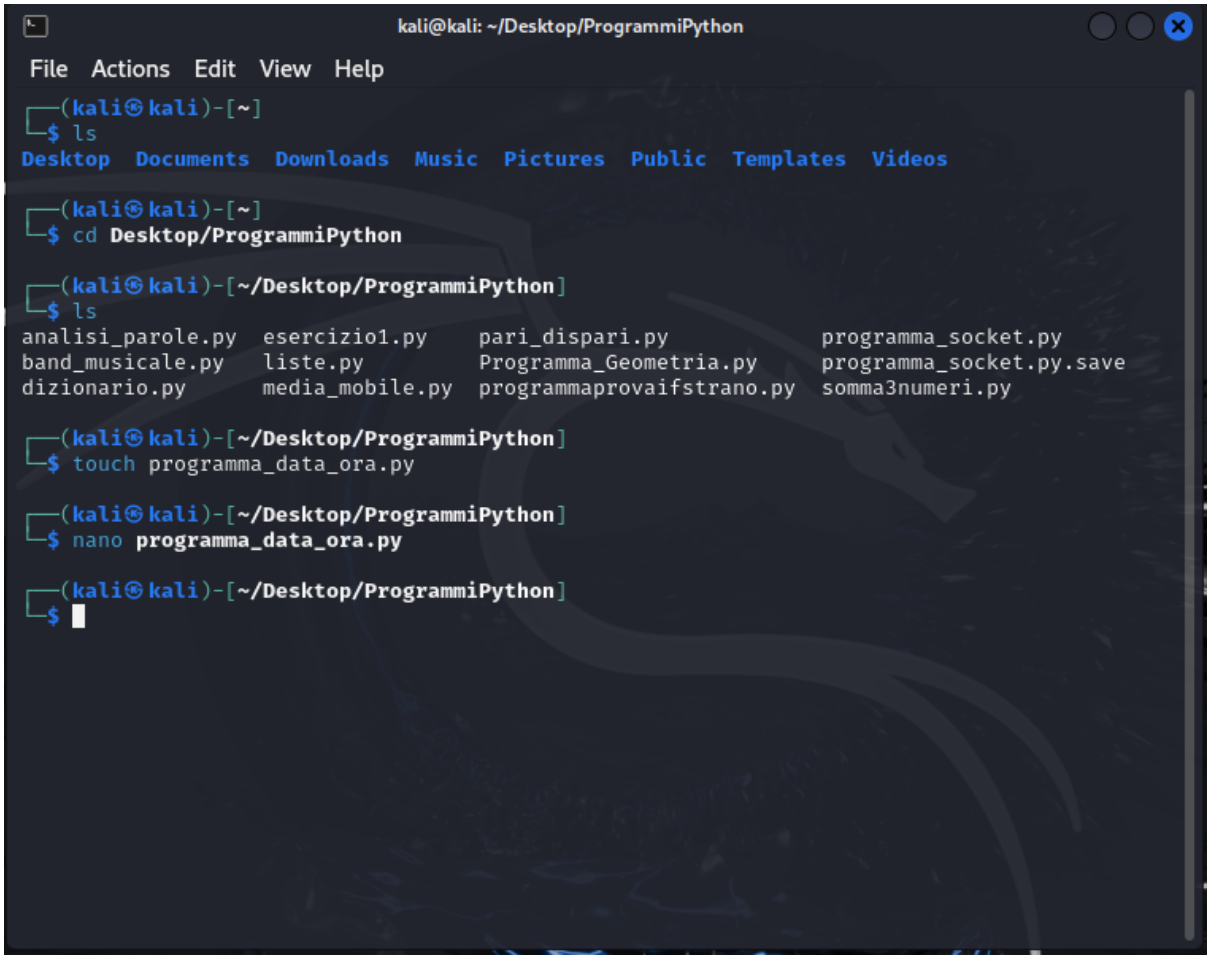
Poi si vanno a risolvere i problemi logici, per primo si va a risolvere il problema alla radice ossia che l'utente non ha la benchè minima idea di cosa faccia questo programma, quindi prima del ciclo si fa un print dove si dice all'utente che cosa vuole fare : (print=" Benvenuto utente questo è un programma che ti permette di effettuare queste determinate azioni") .
Nel ciclo while, nella fase di input si và a dire all'utente le scelte che può fare
comando_utente = input("Cosa vuoi sapere? ") --> comando_utente = input(": Qual'è la data di oggi? , Che ore sono?, Come ti chiami?, e se non vuoi più continuare a scegliere puoi interrompere il programma scrivendo esci ").lower()
Con .lower() si andrà a convertire l'input dell'utente tutto in minuscolo.
Nella parte della funzione andremo a modificare i controlli dell'if elif else, ossia :
if comando == "Qual è la data di oggi?": --> if comando == "qual è la data di oggi?":
elif comando == "Che ore sono?": --> elif comando == "che ore sono?":
elif comando == "Come ti chiami?": --> elif comando == "come ti chiami?"
else (risposta = "Non ho capito la tua domanda." --> risposta = "Mi dispiace hai inserito un input errato, non ho capito").

Facendo questi cambiamenti il programma funziona logicamente e correttamente come è stato pensato. (lascio uno screen con le modifiche al programma effettuate)

```
 GNU nano 8.2                                                programma_data_ora.py
import datetime

def assistente_virtuale(comando):
    if comando == "qual'è la data di oggi?":
        oggi = datetime.datetime.today()  # restituisce la data
        risposta = "\nLa data di oggi è " + oggi.strftime("%d/%m/%Y\n")  # strftime() è un metodo che formatta l'oggetto datetime
    elif comando == "che ore sono?":
        ora_attuale = datetime.datetime.now().time()  # restituisce l'ora, restituisce la data/ora come oggetto di tipo datetime
                                                      # in questo caso restituisce solo l'ora visto che è specificato .time()
        risposta = "\nL'ora attuale è " + ora_attuale.strftime("%H:%M:%S\n")  # qui ho aggiunto anche i secondi %S
    elif comando == "come ti chiami?":
        risposta = "\nMi chiamo Assistente Virtuale\n"
    else:
        risposta = ("\nMi dispiace hai inserito un input errato, non ho capito, le scelte possibili sono le seguenti \n")
    return risposta

print("Benvenuto utente Questo è un programma che ti permette di effettuare queste determinate azioni: \n") #ho aggiunto in input prima del ciclo perchè
                                                                                                             #senò questa scritta riapparirebbe ogni volta
while True:
    comando_utente = input(
        ": Qual'è la data di oggi? \n: Che ore sono?\n: Come ti chiami?\nse non vuoi più continuare a scegliere "
        "puoi interrompere il programma scrivendo 'esci'.\n"
    ).lower()  # Qui ho aggiunto .lower() per modificare l'input in minuscolo, così è più semplice confrontare le stringhe

    if comando_utente == "esci":  # Ho rimosso parentesi extra da comando_utente()
        print("Arrivederci!")
        break
    else:
        print(assistente_virtuale(comando_utente))  # Chiamata alla funzione
#Ho aggiunto vari \n negli output sia all'inizio che alla fine per creare un'output concatenato più visibile all'utente senò usciva
#una stringa sotto l'altra attaccate senza spazi e non era leggibile.
```

Comandi cmd per creare il file.py:

Aperta la shell, con il comando cd ci andiamo a spostare nella directory interessata, in questo caso visto che avevo creato una cartella in precedenza sul Desktop con nome ProgrammiPython il comando sarà : cd Desktop/ProgrammiPython (la / in mezzo serve a concatenare gli spostamenti).
Con il comando ls andremo a vedere il contenuto della cartella.
Con il comando touch nomefile andremo a creare il file ricordandoci di mettere l'estensione.py alla fine del nome.
Con il comando nano nomefile andremo ad aprire un semplice editor di testo dove andremo a creare/modificare il nostro programma.

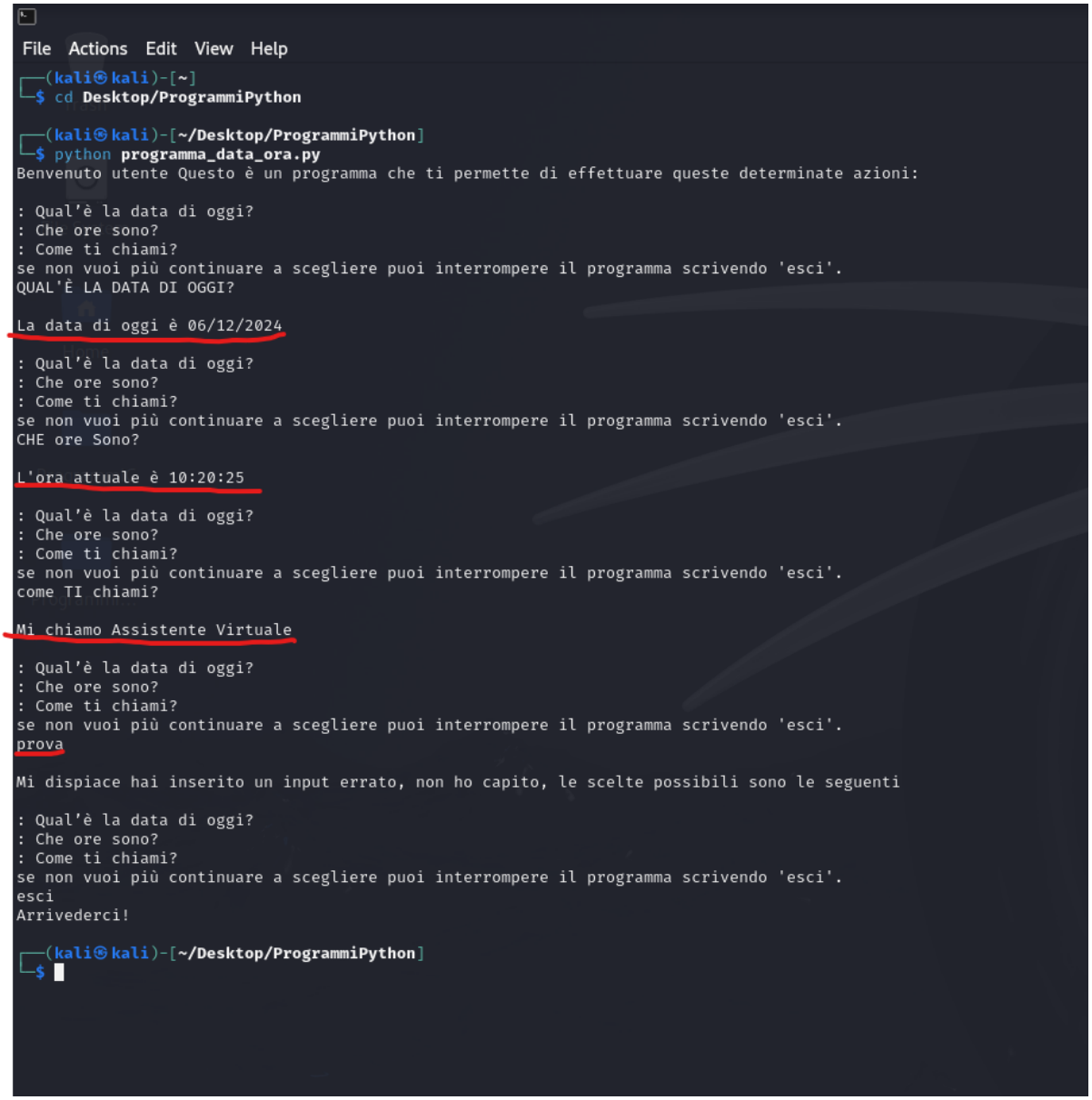


Per salvare sotto nano si utilizza la combinazione di comandi Ctlr + o + invio

Per tornare alla shell si utilizza la combinazione Ctlr + x

Come si esegue il file.py?

Per eseguire il file.py si utilizza il comando python nomefile.py



Se il programma è privo di errori riuscirà ad eseguire tutto il programma e non avremo in output errori.
In questo caso il programma funziona, facendo vari test scrivendo lettere maiuscole/minuscole i risultati appaiono, scrivendo per esempio prova quindi non un'opzione disponibile, non si esce dal ciclo e il programma ti fà rifare l'input.
Scrivendo esci, il programma darà in output Arrivederci! e il programma termina.