**TRABAJO PRÁCTICO N°2 - ING. SOFTWARE II**

**ALUMNO**: FERNANDO PÉREZ

**DOCENTES**: Dr. Pedro E. Colla y Hernán Sanchez

**UADER-FCYT**

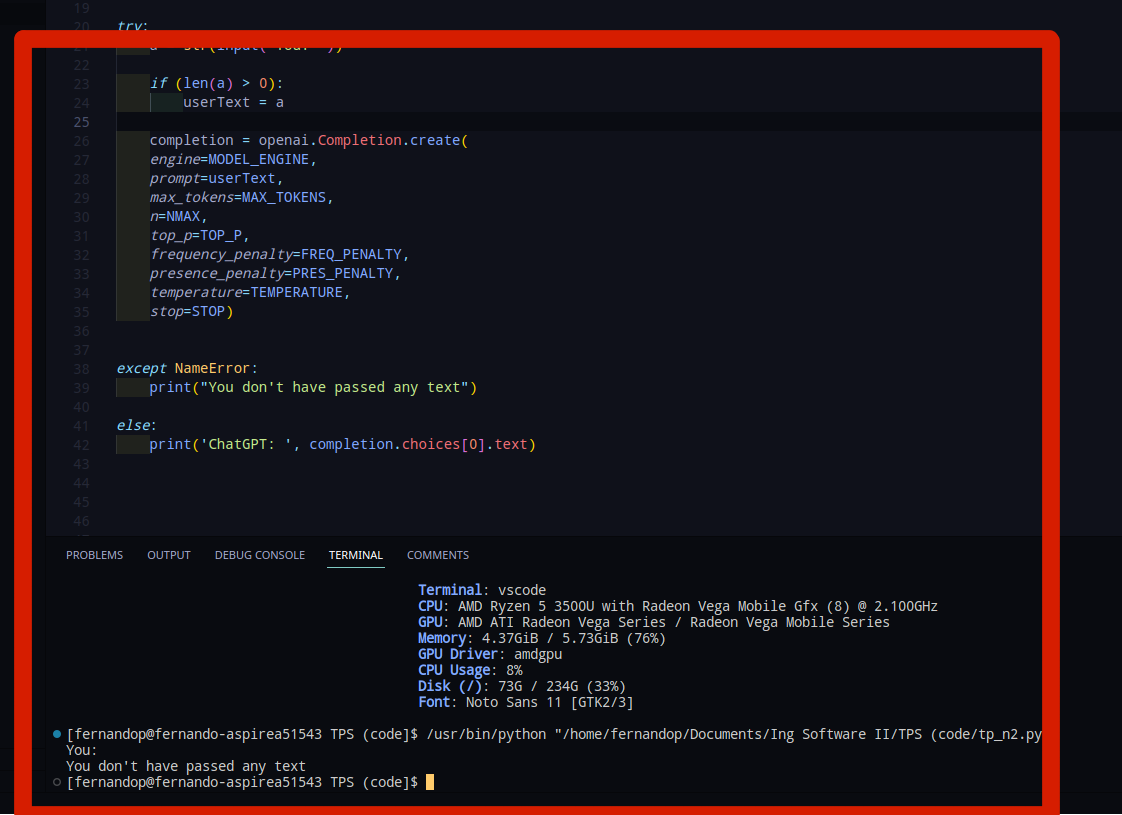
**1)** Se crea el programa que acepta consultas del usuario, contemplando si tiene alguna o no y se imprime su contenido, como se muestra a continuación:



Impresión en pantalla de la consulta, en este caso “Qué es Python?”, como se muestra a continuación:



**2)** En éste punto se contempla el error de no pasarle ningún parámetro, lo que hacemos es que si el usuario no pasa ningún mensaje no lo asigne a la variable, de ésta manera el error que se produce al no definirse la variable en el bloque try lo capturamos en el bloque exception y lanzamos un mensaje al usuario como se ve en la imagen siguiente:



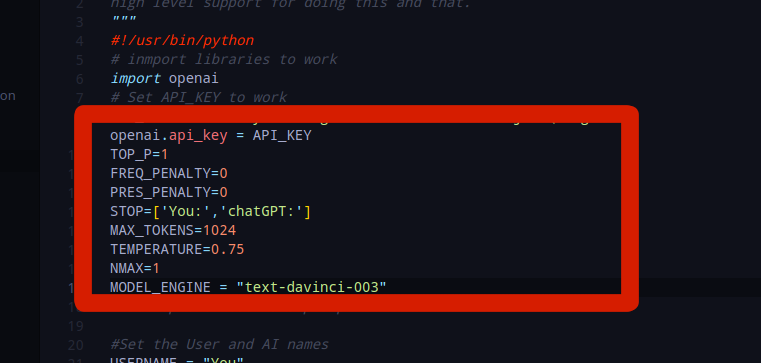
**3)** En ésta etapa se agrega la posibilidad de recuperar la última consulta realizada, mediante el uso de la librería readline (como se muestra en la siguiente imagen importada) y en la prueba se utiliza ciclo for para hacer prueba de la consulta:



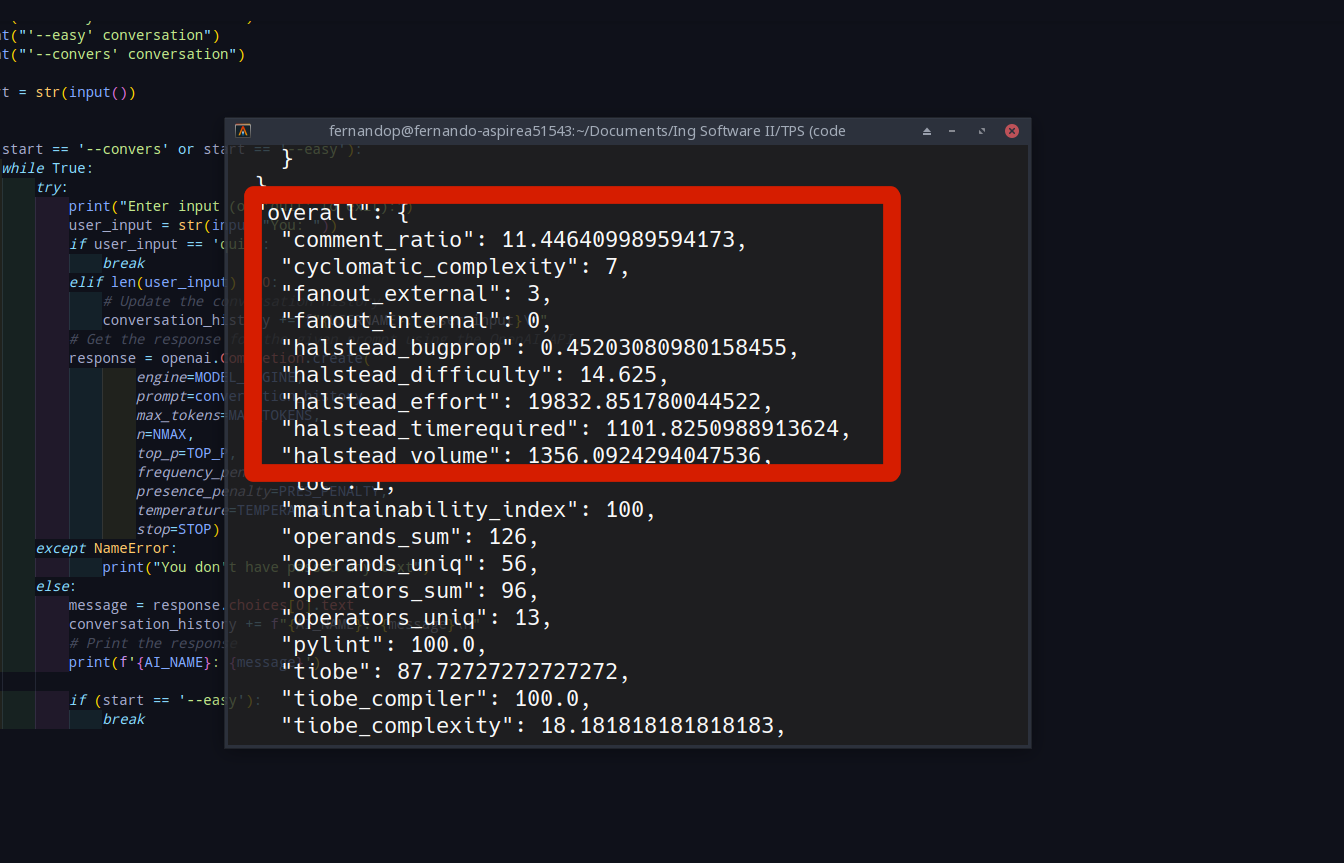
**4)** Se modifica el ingreso de un argumento ”--convers” en una consulta inicial la cuál hay dos caminos, el dicho anteriormente que entra en un modo de conversación en el cuál se va almacenando la conversación en un historial y se la vuelve a pasar como parámetro al chatGPT cada vez que se realiza una nueva consulta, y el segundo que simplemente se lo agregué personalmente que es el argumento ”--easy” que sólo es para realizar una única consulta y finalizar el programa. Se muestra a continuación respectiva imagen:



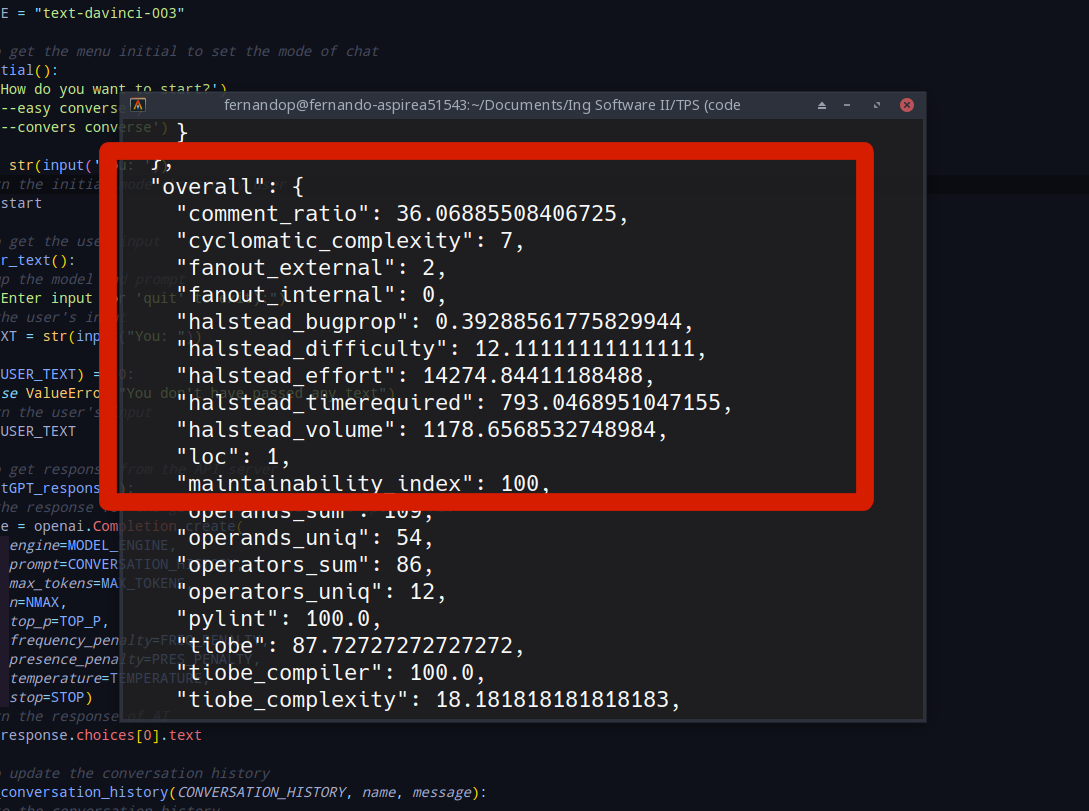
Y se le agregan los prefijos “You: ” y “chatGPT: ” al parámetro STOP como se ve a continuación:



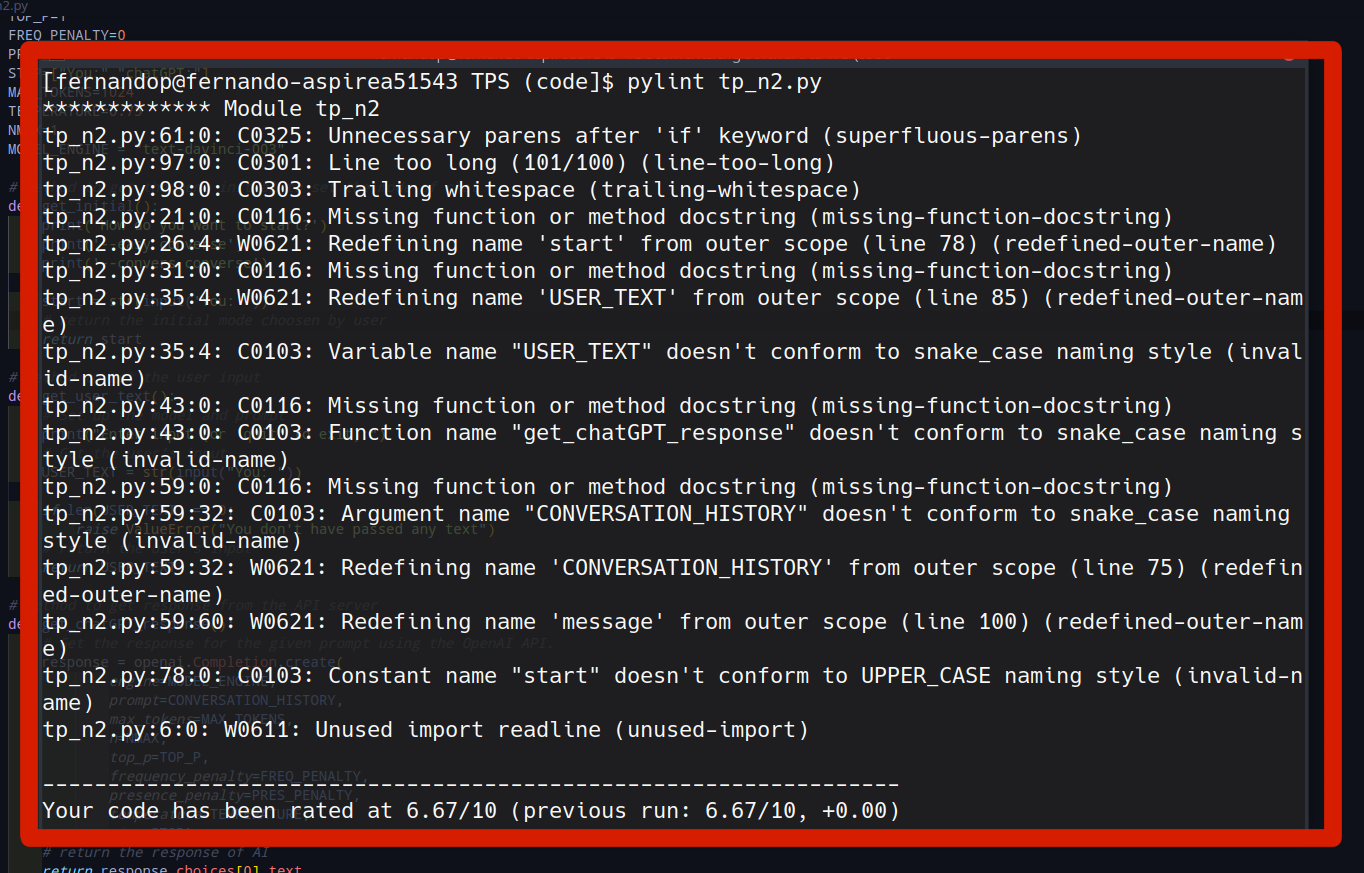
**5)** Se ejecuta el programa multimetric en el programa creado para consultar a la IA y de la sección “Overall” se obtienen los siguientes datos:



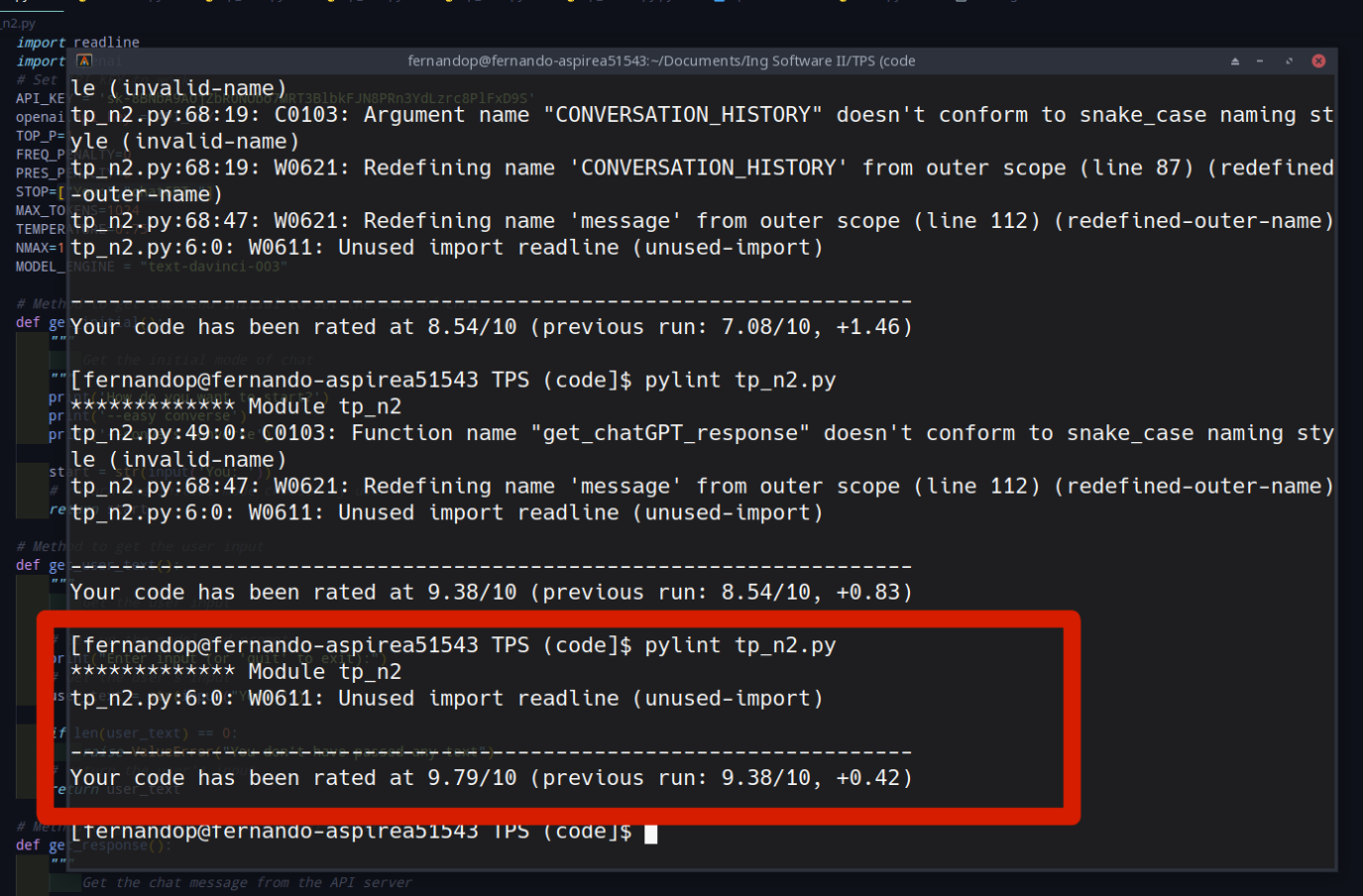
Al agregarle comentarios a las funciones y parámetros notamos que el comment\_ratio sube lo que facilita a la lectura del código y el entendimiento de dichas funciones, por otro lado para reducir la cyclomatic\_complexity de 7 a 3, y todos los halsted (bugprop, difficulty, effor, timerequired, volume) que bajen, lo solucioné separando el código en módulos, en una parte el input del usuario dentro de un método, en otro el get response de la API y demás lo cuál no solo simplifica el código sino que la complejidad ciclomática la reduce a valores muy buenos como se ve en la imagen a continuación:



**6)** Al ejecutar el programa pylint sobre el ejercicio de consulta a la API de chatGPT se obtuvo los siguientes datos con respecto a la programación del código:

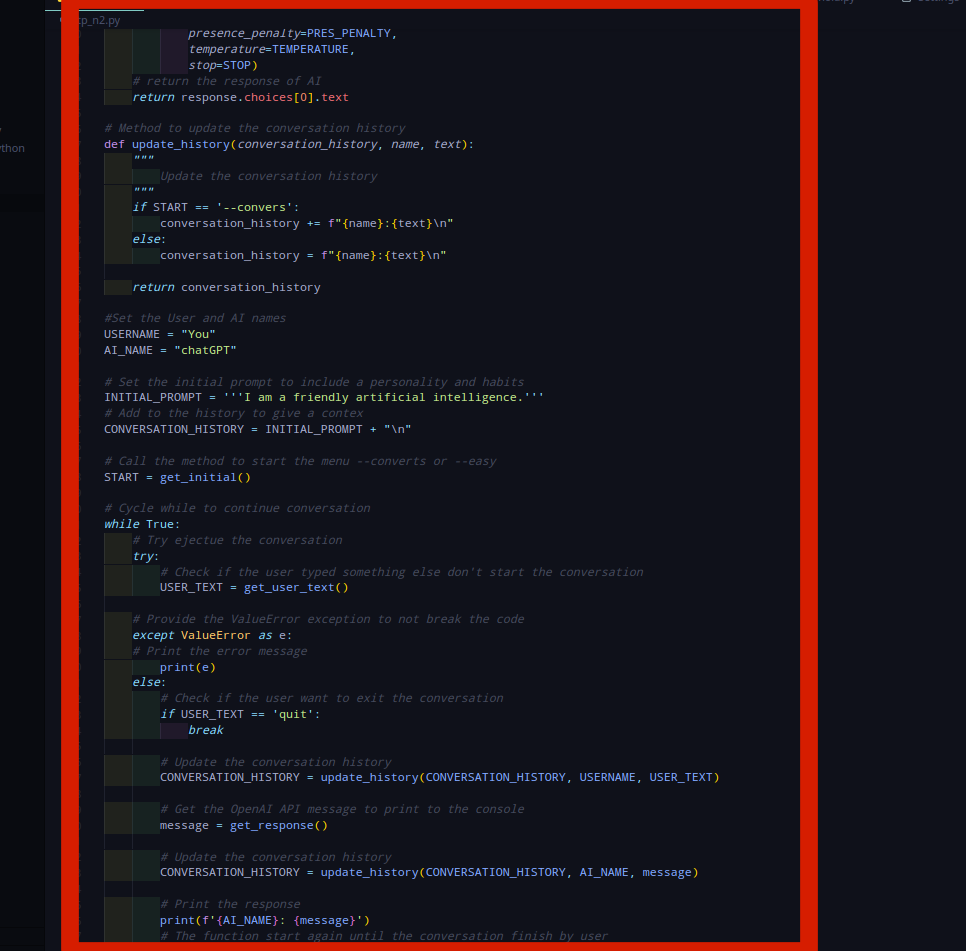


Al inicio de la siguiente imagen se puede ver los “errores” o recomendaciones dadas por el programa pylint hasta el final (encerrado con marco rojo) que se muestra que se cumplió con todas las recomendaciones brindas salvo una y obteniendo una puntuación de código de 9.79 / 10.



La razón por la cuál omití la recomendación o error es que como se ve dice que hay una librería readline en desuso lo cuál en el código es cierto que no es utilizada, pero si en la consola en la ejecución del programa. Con lo cuál si no la importo no se puede hacer uso de la tecla “cursor up” para obtener la última inserción en la consola, por lo cual decidí dejarla.

Por último, así quedó el código después de cumplir con todas las recomendaciones, en esta imagen a continuación: (se muestra una porción de código, para más detalles el archivo se encuentra en el repositorio de github en la dirección src/chatGPT/)



**GITHUB REPOSITORY:** <https://urlis.net/repo-chatGPT>