****

**UADER Facultad de Ciencia y Tecnología – CdelU**

***Carrera:* Licenciatura en Sistemas de Información**

***Asignatura:* Ingeniería en software II**

***Autores:* Escar Camilo**

***Profesores:* Dr. Pedro E. Colla**

**Ayudante: Hernán Sánchez**

***Año:* 3er Año**

***Ciclo lectivo:* 2023**

Consignas

1. Escriba un programa en lenguaje Python que basándose en el esqueleto previamente proporcionado acepte una consulta del usuario, verifique si la misma tiene texto, imprima su contenido, invoque el API de chatGPT con esa consulta e imprima en pantalla el resultado que se obtenga como respuesta. El contenido de la consulta debe agregársele “You:” antes de imprimirlo y de enviarlo. La respuesta de chatGPT deberá agregársele “chatGPT: “antes de imprimirse.
2. Agregue al programa anterior estructuras Try:/Except: para gestionar problemas en la ejecución, coloque un nido para la aceptación de consulta desde el usuario, otro para su tratamiento y un tercero para la invocación.
3. Agregue la posibilidad de recuperar la última consulta realizada para poder editarla y volver a enviar a chatGPT, esto debe hacerse con la tecla “cursor Up”.

<https://github.com/CamiloEscar/UADER_IS2_ESCAR/blob/af879cb5f2b16247cfdec61818908cb3dfc0a449/TP%20N2/Estructura_Basica.py>

1. Acepte un argumento de llamada “—convers” y en caso de existir genere un modo de “conversación” caracterizado por:

a. Asegúrese de revisar si el argumento existe y es el esperado, caso contrario debe ser ignorado (consejo: revise primero si se ha indicado algún argumento).

b. Cada vez que se acepta una consulta no nula se agrega a un buffer.

c. Cada interacción con chatGPT utilizará la última consulta realizada agregada a todas las anteriores.

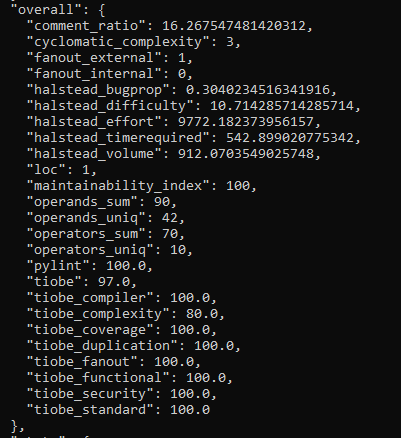
d. La respuesta de chatGPT se agregará al buffer para ser también re-enviada en las próximas consultas.

e. Al actualizar el contenido de la consulta o la respuesta al buffer asegúrese que se incluyen los prefijos “You: “ y “chatGPT: “, en las llamadas de API realizadas bajo ésta modalidad debe utilizarse un contenido del parámetro STOP con lo siguiente: STOP=[“You:”,”chatGPT:”]

* No llegue a completarlo, en caso de ser necesario vuelvo a entregarlo

1. Ejecute el programa multimetric sobre la última versión que disponga del programa utilizado en éste practico. Realice las siguientes consignas:

a. De la sección “Overall” del resultado verifique el resultado de “comment\_ratio” (proporción de comentarios). Si éste es menor de 1/3 (33%) explore medidas para mejorar éste parámetro hasta un valor en ese entorno.



Para aumentar el valor de “comment\_ratio”, la accion que tome fue aumentar la cantidad de comentarios en el codigo implementado



b. Explore el significado de y luego compare los valores de halstead\_effort y halstead\_timerequired con los que efectivamente le tomó el programa.

La fórmula de Halstead Effort se utiliza para calcular el esfuerzo requerido para implementar un programa de software se expresa de la siguiente manera:

E = N \* log2(n)

Halstead Time Required es otra medida de complejidad de software que se basa en la fórmula del esfuerzo de Halstead y que se utiliza para estimar el tiempo que se necesita para escribir o compilar un programa de software y se expresa de la siguiente manera:

T = E / 18 donde T es el tiempo requerido para implementar el programa en segundos y E es el esfuerzo estimado del programa calculado con la fórmula de Halstead Effort.



c. Como compara el valor halstead\_bugprop con la cantidad de defectos que tuvo que solucionar luego que lograra que el programa ejecute por primera vez (es decir, excluyendo errores de sintaxis).



El valor de Halstead Bug Prop se calcula utilizando la siguiente fórmula:

B = (E^2 / V) \* 0.667 donde B es el número de errores o bugs que se esperan encontrar en el programa, E es el esfuerzo calculado con la fórmula de Halstead Effort y V es el tamaño del vocabulario del programa, es decir, el número total de operadores y operandos únicos utilizados en el programa.

Una vez que se ha calculado el valor, se puede comparar con la cantidad de defectos que se encontraron después de que el programa se ejecutara por primera vez. Si el número de defectos es mayor, esto podría indicar que el programa es más complejo de lo que se esperaba y que puede requerir más trabajo para corregir los errores.

d. Que estrategias cree que se pueden aplicar para reducir el índice de McCabe (cyclomatic\_complexity) en un 10%.

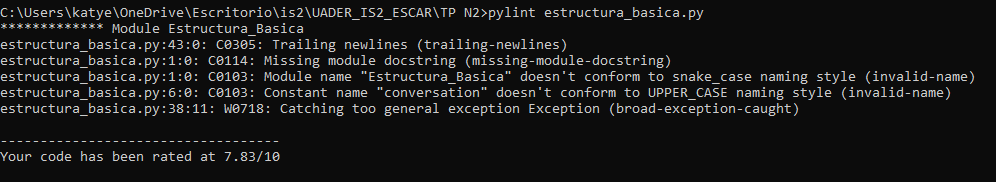


Algunas estrategias que pueden ayudar a reducir la complejidad ciclomática en un 10%:

* Descomponer funciones complejas en funciones más pequeñas: Si una función tiene una complejidad ciclomática alta, se puede descomponer en funciones más pequeñas y más simples. Esto puede reducir la complejidad ciclomática total del programa y hacer que sea más fácil de entender y mantener.
* Reducir la cantidad de condiciones y bucles anidados: Los condicionales y los bucles anidados pueden aumentar la complejidad ciclomática de un programa. Al reducir la cantidad de condiciones y bucles anidados, se puede reducir la complejidad ciclomática y hacer que el código sea más fácil de leer y mantener.
* Utilizar estructuras de control más simples: Las estructuras de control como switch o casos múltiples pueden ser reemplazadas por estructuras de control más simples como if o for. Esto puede reducir la complejidad ciclomática y hacer que el código sea más fácil de entender.
* Eliminar código muerto: El código que no se ejecuta nunca, también conocido como código muerto, puede aumentar la complejidad ciclomática innecesariamente. Al eliminar el código muerto, se puede reducir la complejidad ciclomática y hacer que el código sea más fácil de entender.
* Utilizar herramientas automatizadas de análisis de código: Las herramientas de análisis de código pueden ayudar a identificar áreas del código que tienen una complejidad ciclomática alta. Utilizando estas herramientas, se pueden identificar áreas específicas del código que requieren atención y se pueden aplicar las estrategias mencionadas anteriormente para reducir la complejidad ciclomática.

6. Instale un analizador estático de código denominado pylint (mediante el comando pip install pylint) y realice las siguientes acciones.

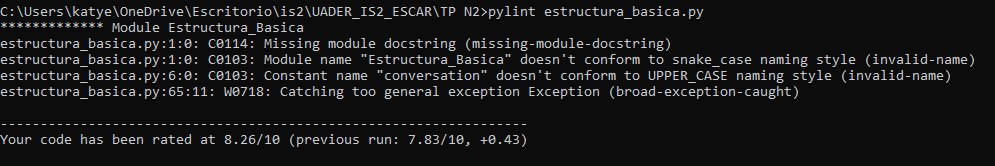
a. Ejecute el programa pylint sobre el programa python desarrollado como parte de las consignas del punto -4- anterior.



b. Analice el resultado que arroja.

i. Introduzca correcciones en el programa fuente para abordar los comentarios. Incluya los comentarios en la primera corrida y como queda luego de todas las correcciones realizadas.

Trailing newlines- use el método .rstrip(), este método elimina todo tipo de espacio en blanco



C0103: Constant name "conversation" doesn't conform to UPPER\_CASE naming style (invalid-name)

Al querer usar el estilo “snake\_case” me deja de tomar el valor de la constante

c. Aquellos comentarios que decida no abordar justifique brevemente porque decidió ignorar las recomendaciones.

- no realice ninguna corrección más ya que no lo considere importante para el funcionamiento general del programa, si para mejorar y tener buenas prácticas con el código