MANUAL TECNICO

\*.- GUI

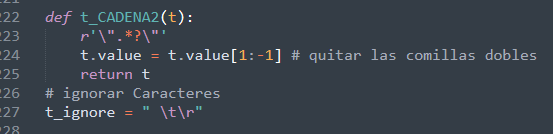
Funcion de determina el inicio del proyecto tanto UI como UIX siendo ***“def iniciar (self)”*** Metodo y/o constructor que arranca la gui del proyecto así como IUX, en la línea 126 carga ventana para desplegar la interfaz, posteriormente en la línea 129 mediante “window.title(“Proyecto”)” despliega el título del proyecto el cual muestra en la ventana asi como las dimensiones del proyecto siendo la modificación que permite determinar dimensiones de ventana “window.geometry(“650x600”)” considerando la pixelacion del proyecto.



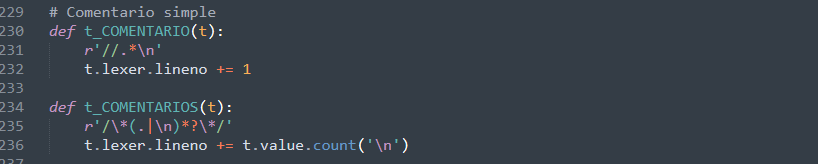
De la Línea 136 a la línea 148 se crean botones siendo btn, btn2, 3 y 4 los cuales despliegan los reportes de tablas, reportes en reporteerr mediante el método de asignación “partial(reporteerr, self)”. En este caso se agregan los componentes de scr = Scrollbar() para validar la parte de vistas mediante “command=tex.yview”.

\*.- Estructura Descendente:

En el método de la figura 1.a se observa que recibe el parámetro de siendo un char o Cadena del cual valida e ignora los comentarios en cadena.

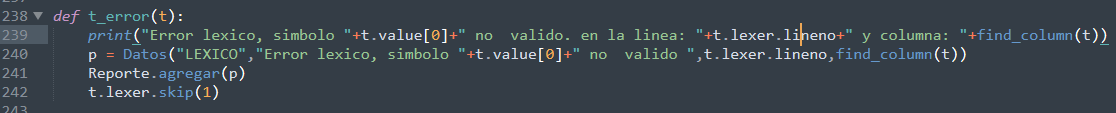


Para la validación de comentarios Simples en el editor se tendrá las respectivas validaciones siendo que mediante el analizador léxico vaya concatenando uno por uno según se requiera ò asi mismo que reciba comentarios multiples mediante la expresión regular “r'//.\*\n'” la cual permite validar y llevar un contador por línea que se vaya agregando al editor de sql.

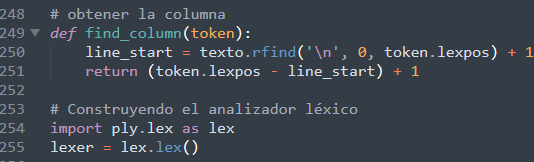


Se aprecia que para el primer método mediante la expresión regular “r'\".\*?\"'” se valida que sean comentarios simples y mediante la siguiente expresión regular “r'//.\*\n'” se valida que sean comentarios de forma más extendida siendo multi-comentarios.

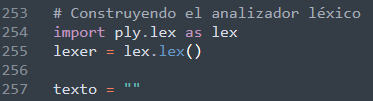
\*.- Líneas de validación:

Metodo que define: ”def t\_error(t)” si se agrega un valor que no sea léxico en la línea de lectura en el editor de texto que valida sentencias sql.

Mediante el parametro de instancia p Se agregan al método del constructor “"LEXICO","Error lexico”, “simbolo "+t.value[0]+" no valido”, “t.lexer.lineno,” y “find\_column(t)” siendo por cada parametro el primero como String, String, y por cada valor para la tabla de símbolos se agrega ”find\_column(t)” el cual valida si recibe los parámetros por cada uno.

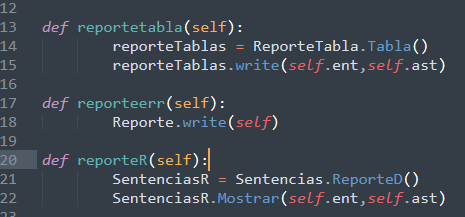


Se agrega el valor por cada parametro en la conlumna del método “find\_column(token)” el cual permite validar mediante el testeo de “line\_start = texto.rfind('\n', 0, token.lexpos) + 1” y el valor permitente return tanto para el token como para el valor de lexpos por cada valor que se obtenga para posteriormente se construye mediante:

Así mismo el valor de construcción para el analizador se carga siendo en cada valor así como ply.lex en cada valor que se importa la librería en el valor de texto.

\*Tablas:

El método “reportetabla(self)” crea y obtiene instancia respecto a la tabla en la cual se obtiene dos parámetros siendo la entrada y el valor del ast.



Al segundo método “reporterr(self)” se envía el parámetro el cual va a escribir a la instancia del método.

Por ultimo del archivo gui.py en la línea 20 del método “def reporteR(self)” se crea la setencia y asigna a “Sentencias.ReportedD()” para posteriormente enviar los dos objetos como parámetros quedando asi “SentenciasR.Mostrar(self.ent, self.ast)” en tal caso ir a mostrarlos por cada método en el árbol.

Para ambos método se utiliza el objeto de instancia siendo self el cual hace referencia a dicho parámetro del solicitado.

En cada método se agrega su instancia respectiva por reporteR(self), reporteerr(self) y reportetabla(self), por cada método también se tienen las instancias asigandas y parametrizadas.