@jeani-rodas97 usuario de github

Manual técnico

Proyecto 1 201404421

Jeanifer Alejandra García Rodas

IMPORTACION DE LIBRERIAS PARA INTERFAZ GRAFICA Y CLASES

Compi_Proy1 > InterfazGrafica.py > ...

```
1  from tkinter import *
2  from tkinter import filedialog, scrolledtext, Canvas, messagebox
3  from AnalizadorHTML import HTML
4  from AnalizadorCSS import CSS
5  from AnalizadorJS import JS
6  from AnalizadorRMT import RMT
7  from Token import Tkn
8  from Error import ERROR
```

CREACIÓN DEL MENU

```
#Menu de barra
MenuOpArch = Menu(MenuSup, tearoff=0)
MenuOpArch.add_command(label = "Nuevo", command = self.MenuNuevo)
MenuOpArch.add_command(label = "Abrir", command = self.MenuAbrir)
MenuOpArch.add_command(label = "Guardar", command = self.ArchGuardar)
MenuOpArch.add_command(label = "Guardar Como", command = self.ArchGuardarComo)

MenuSup.add_cascade(label = "Archivo", menu = MenuOpArch)
MenuSup.add_cascade(label = "Analizar", command = self.Analizar)
MenuSup.add_cascade(label = "Salir", command = self.OpSalir)
```

ANALIZAR RUTAS

```
Ruta = self.Archivo
extension = Ruta.split(".")
mensaje = messagebox.showinfo("Tipo Archivo", "Es un archivo "+ extension[1])
if (extension[1] == "html"):
    HTML(self.editor.get(1.0, END))
elif (extension[1] == "css"):
    CSS(self.editor.get(1.0, END))
elif (extension[1] == "js"):
    JS(self.editor.get(1.0, END))
elif (extension[1] == "rmt"):
    RMT(self.editor.get(1.0, END))
else:
    messagebox.showerror("ERROR", "Extensión no aceptada")
self.Color()
self.EscribirConsola()
self.EscribirEstados()
```

COLOREAR LAS PALABRAS

```
def Color(self):
    listaC = []
    listaC = Tkn.ConsultaColor(self)
    i=0
    #print("Llegue a la lista ")

    for c in listaC:
        for item in range(len(c)-3):
            #print(item)

            #print("leyendo la lista")
            if(c[item+3] == "Signo"):
                color = "orange"
            elif(c[item+3] == "Cadena"):
                color = "yellow"
            elif(c[item+3] == "Reservada"):
                color = "red"
            elif(c[item+3] == "Comentario"):
                color = "gray"
            elif((c[item+3] == "Numero")|(c[item+3] == "Porcentaje")):
                color = "blue"
            elif(c[item+3] == "Variable"):
```

INSTANCIAR LA VENTANA

```
if __name__ == "__main__":
    ventana = Tk()
    app = Interfaz(ventana)
    Interfaz(ventana).editor.focus()
    ventana.mainloop()
```

TOKENS

INSTANCIA DE LAS LISTAS PARA ALMACENAR INFORMACION

```
class Tkn:

    def __init__(self, id, lex, tipo, fila, col):
        self.id = id
        self.lex = lex
        self.tipo = tipo
        self.fila = fila
        self.col = col
        listaTK.append([id, lex, tipo, fila, col])
        listaCol.append([fila, col - len(lex), col , tipo])
```

RETORNO DE LISTA PARA SU ANALISIS

```
def ConsultaColor(self):
    return listaCol
```

ESCRIBIENDO EL HTML DE REPORTE DE TOKENS

```
def ReporteToken(self):
    archivo = open("C:/Users/Jeany/Documents/Archivos_Python/Compi_Proy1/ReporteTk.html", "w")
    archivo.writelines("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\" />" + "\n" + "
    archivo.writelines("<HTML><HEAD><TITLE>COMPILADORES 1</TITLE></HEAD>" + "\n" + "\n")
    archivo.writelines("<H1><CENTER><B><FONT SIZE=\"12\" COLOR=\"PINK\">LISTADO DE TOKENS</FONT></B><BR></H1>
    archivo.writelines("<HR>" + "\n" + "\n")
    archivo.writelines("<BR><CENTER><TABLE BORDER=1>\n")
    archivo.writelines("    <TR>\n")
    archivo.writelines("    <TD ALIGN=\"CENTER\"><FONT COLOR=\"PURPLE\"><B>No.</B></FONT></TD>\n")
    archivo.writelines("    <TD ALIGN=\"CENTER\"><FONT COLOR=\"PURPLE\"><B>Id Token</B></FONT></TD>\n")
    archivo.writelines("    <TD ALIGN=\"CENTER\"><FONT COLOR=\"PURPLE\"><B>Lexema</B></FONT></TD>\n")
    archivo.writelines("    <TD ALIGN=\"CENTER\"><FONT COLOR=\"PURPLE\"><B>Tipo</B></FONT></TD>\n")
    archivo.writelines("    <TD ALIGN=\"CENTER\"><FONT COLOR=\"PURPLE\"><B>Fila</B></FONT></TD>\n")
    archivo.writelines("    <TD ALIGN=\"CENTER\"><FONT COLOR=\"PURPLE\"><B>Columna</B></FONT></TD>\n")
    i=1
    for t in listaTK:
        for item in range(len(t)-4):

            archivo.writelines("    <TR>\n")
            archivo.writelines(f"   <TD ALIGN=\"CENTER\"><FONT COLOR=\"RED\"><B> {i} </B></FONT></TD>\n")
            archivo.writelines(f"   <TD ALIGN=\"CENTER\"><FONT COLOR=\"BLACK\"><B> {t[item]}</B></FONT></TD>\
            archivo.writelines(f"   <TD ALIGN=\"CENTER\"><FONT COLOR=\"BLACK\"><B> {t[item+1]}</B></FONT></TD
            archivo.writelines(f"   <TD ALIGN=\"CENTER\"><FONT COLOR=\"BLACK\"><B> {t[item+2]} </B></FONT></T
            archivo.writelines(f"   <TD ALIGN=\"CENTER\"><FONT COLOR=\"BLACK\"><B> {t[item+3]}</B></FONT></TD
            archivo.writelines(f"   <TD ALIGN=\"CENTER\"><FONT COLOR=\"BLACK\"><B> {t[item+4]}</B></FONT></TD

            i +=1

    archivo.write("</TABLE>" + "\n")
    archivo.close()
```

LIMPIAR LISTA

```
def LimpiarHTML(self):
    #remove("C:/Users/Jeany/Document
    del listaCol[:]
    del listaTK[:]
```

ANALIZADORES LEXICO

LEER LOS CARACTERES

```
class HTML:
    def __init__(self, TextoHTML):
        self.TextoHTML = TextoHTML
        #print("Lee bien el texto \n " + TextoHTML)
        estado = 0
        lexema = ""
        fila = 1
        columna = 0
        colorear = 0
        #En este for vamos a leer caracater por caracter de la cadena, ahora hay que enviarlo al estado
        #Cambiaremos a while porque el for siempre avanza y necesitamos retroceder una iteración
        i = 0
        while i < len(TextoHTML):
            caracter = TextoHTML[i]
            if (estado == 0):
                lexema = ""
                if ((caracter == '<')|(caracter == '>')|(caracter == '/')|(caracter == '=')|(caracter == '('
                    columna+=1
                    colorear+=1
                    lexema += caracter
                    i+=1
                    estado = 1
                elif((caracter == '"')|(caracter == '\'')|(caracter == '”')):
                    columna+=1
                    colorear+=1
                    lexema += caracter
                    i+=1
                    estado = 5
                elif (caracter.isalpha()):
                    columna+=1
```

LISTA DE PALABRAS RESERVADAS

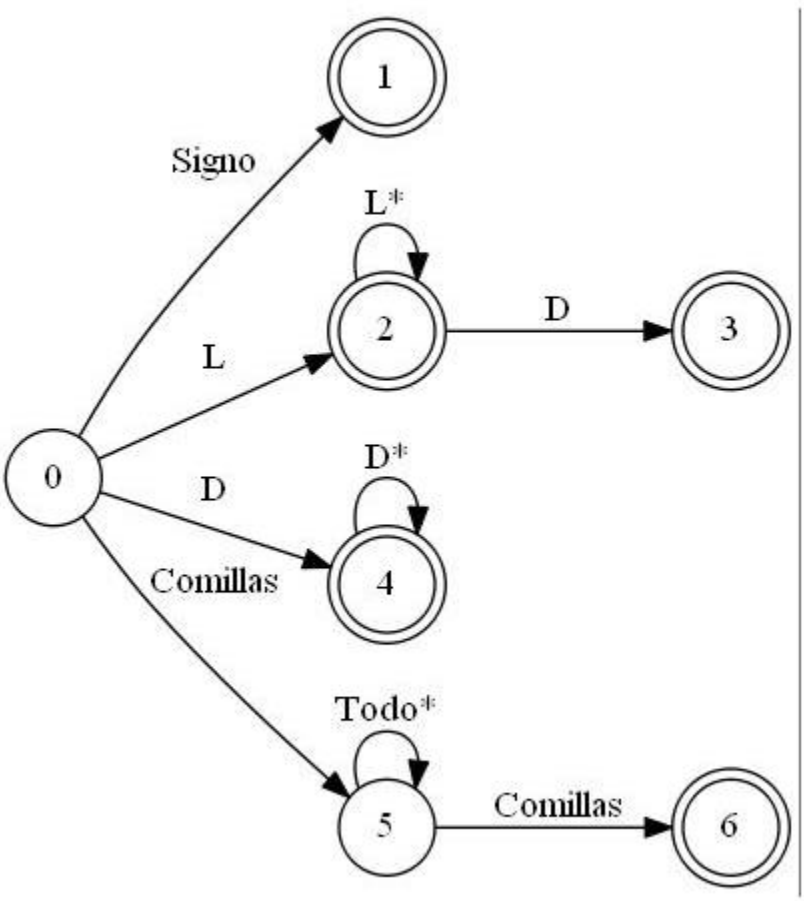
```
def Reservadas(self, palabra):
    #Convierto la cadena a mayusculas con el .upper()
    #Si la quiero todas en minusculas uso el .lower()
    id = 2
    if ((palabra.upper() == "HTML")|(palabra.upper() == "HEAD")|(palabra.upper() == "TITLE")|(palabr
        id = 3
    elif ((palabra.upper() == "H1")|(palabra.upper() == "H2")|(palabra.upper() == "H3")|(palabra.upp
        id = 3
    elif((palabra.upper() == "IMG")|(palabra.upper() == "SRC")|(palabra.upper() == "STYLE")):
        id = 3
    elif((palabra.upper() == "A")|(palabra.upper() == "HREF")|(palabra.upper() == "UL")|(palabra.upp
        id = 3
    elif((palabra.upper() == "TABLE")|(palabra.upper() == "BORDER")|(palabra.upper() == "CAPTION"))
        id = 3
    elif((palabra.upper() == "TR")|(palabra.upper() == "TH")|(palabra.upper() == "TD")):
        id = 3
    elif((palabra.upper() == "COLGROUP")|(palabra.upper() == "COL")|(palabra.upper() == "THEAD")|(pa
        id = 3
    else:
        id = 2
```

LEER EL PATH

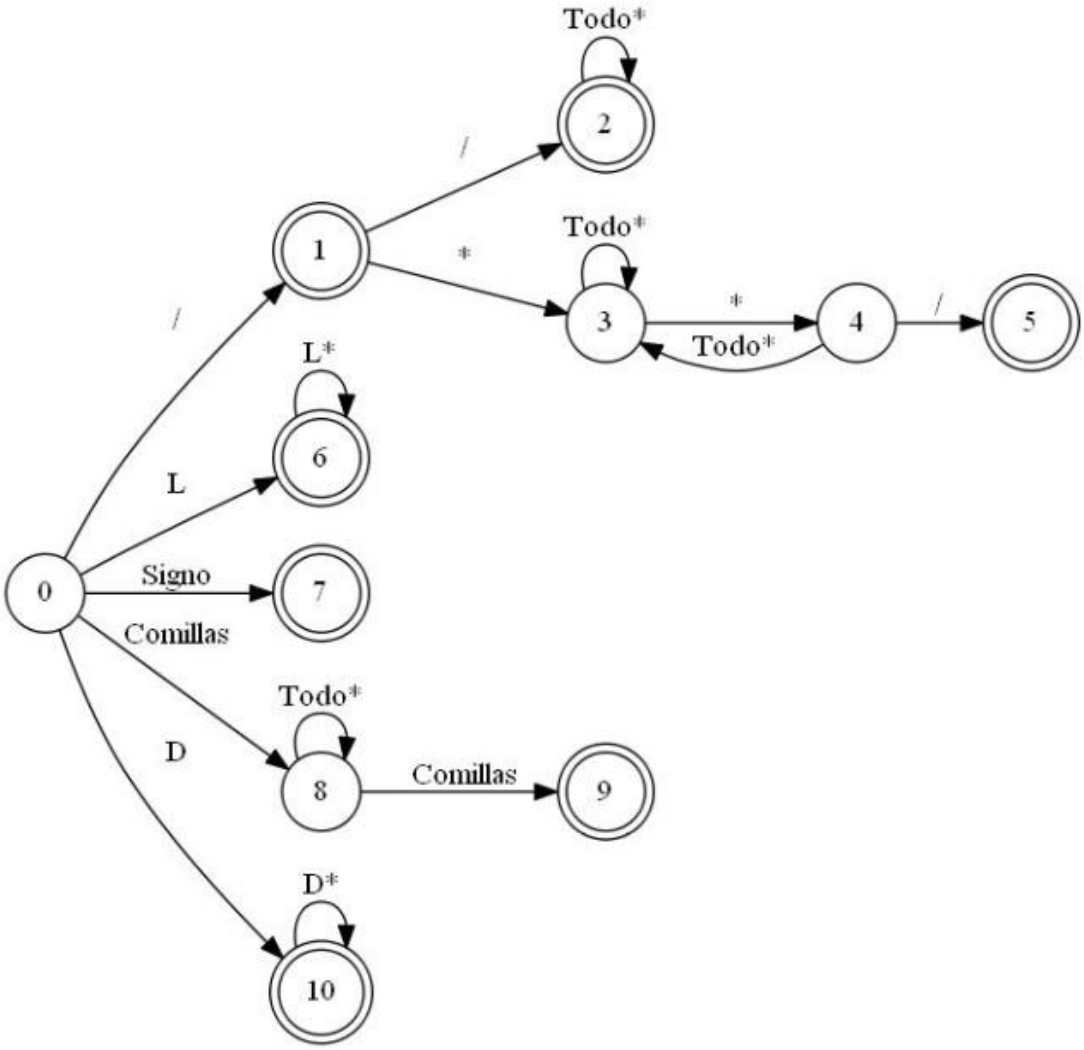
```
def Path(self, texto):
    linea = texto.split("\n")
    salida = linea[0].split("output")
    print(salida[1])
    Tkn.ReporteToken(self, salida[1])
    ERROR.ReporteError(self, salida[1])
```

AUTOMATAS

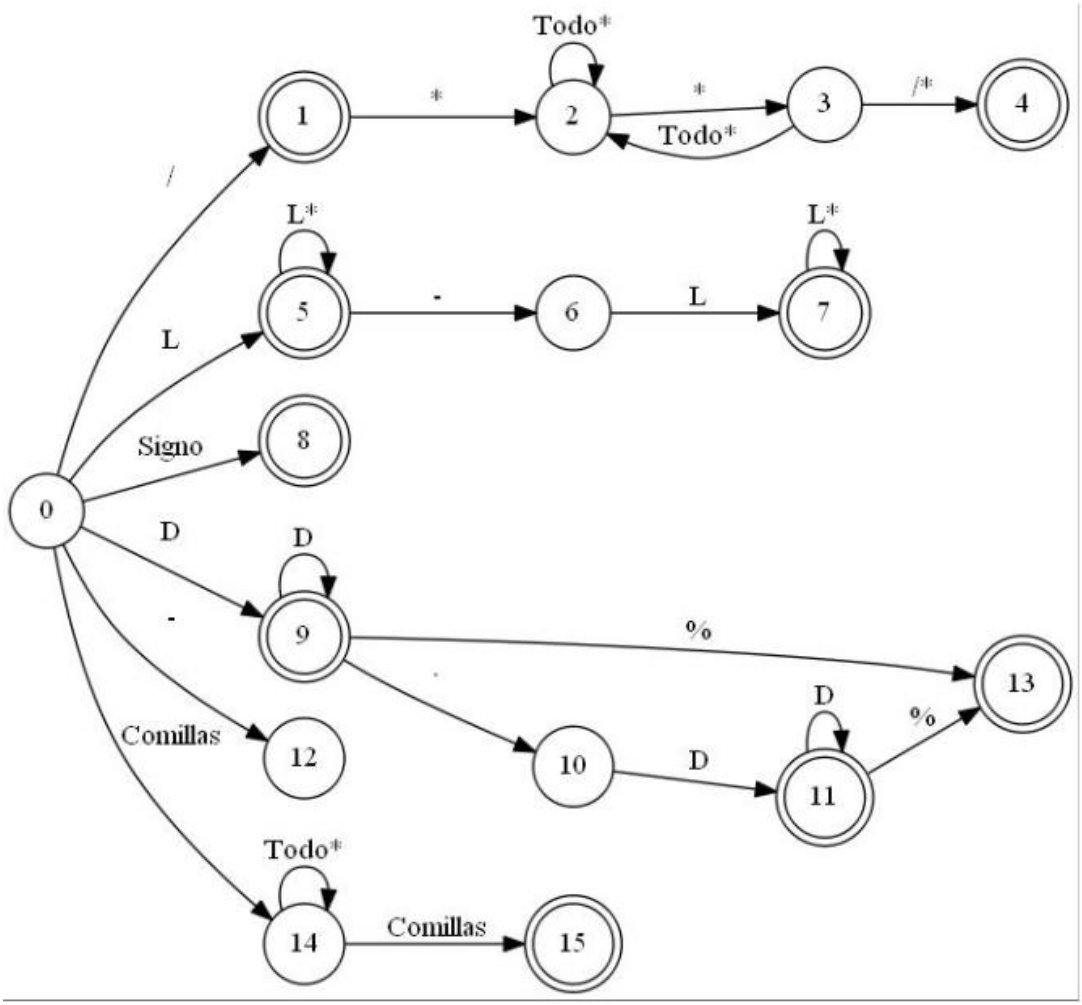
PARA HTML



ANALIZADOR JS



ANALIZADOR PARA CSS



GRAFO PARA RMT