Manual técnico

Constructor principal: el ejecutables esta dentro de App.py, dentro de el se llama a la clase interfaz, para desplegar la interfaz de la app

```
1   from Interfaz import menuPrincipal
2
3   menuPrincipal()
```

Clase menuPrincipal: esta clase se encarga de crear la interfaz, usando tkinter, la función principal es menuPrincipal(), despliega una ventana con esta configuración

```
menu=tk.Tk()
menu.title("Proyecto 2")
menu.geometry("1280x720")
menu.resizable(False, False)
```

Dentro de esta hay 2 botones, un menú desplegable y 2 textBox, el primer botón se llama abrir, el segundo botón se llama, actualizar, el menú desplegable se llama menuDeslizable, el textBox de la izquierda se llama pantalla1, el textBox de la derecha se llama pantalla2

El botón abrir llama a la función abrirArchivo(), que usa tkinter log, para usar el explorador de archivos para capturar la ruta

```
def abrirArchivo():
    global ruta
    archivo = filedialog.askopenfile(filetypes=[("Archivos de texto", "*.txt"),

    if archivo:
        ruta=archivo.name
        txbPantalla1.delete('1.0', tk.END)
        txbPantalla1.insert(tk.END, (str(abrirEntrada(ruta))))
```

El botón analiza llama a la función analizarArchivo(), se encarga de ejecutar la función que analiza lexicamente el archivo

```
def analizarArchivo():
    global ruta
    txbPantalla2.delete('1.0', tk.END)
    txbPantalla2.insert(tk.END, (str(funciones(ruta))))
```

El menú tiene 3 funciones, reporte de errores, reporte de tokenes, árbol de derivación

El seleccionable reporte de errores llama a la función desplegarErrores()

```
def desplegarErrores():
    txbPantalla1.delete('1.0', tk.END)
    txbPantalla1.insert(tk.END, (str(errores())))
```

El seleccionable reporte de tokens llama a la función desplegarTokens()

```
def desplegarTokens():
    global ruta
    txbPantalla2.delete('1.0', tk.END)
    txbPantalla2.insert(tk.END, (str(leerPorSimbolo(ruta))))
```

El seleccionable árbol de desviacion llama a la función desplegarGrafica()

```
def desplegarGrafica():
    global ruta
    grafica()
```

Clase analizadorLexico: se encarga de la función de analizar lexicamente en archivo, dentro de ella se encuentran diferentes funciones, encargadas de las diferentes interpretaciones de archivo

```
Simbolo= namedtuple("Simbolo",["valor", "linea", "columna"])
linea=1
columna=1
simbolos=[]
listaRegistros=[]
listaClaves=[]
listaErrores=[]
listaMensajes=[]
listaMensajesln=[]
```

Función abrirEntrada(ruta): esta función se encarga de abrir el archivo

```
def abrirEntrada(ruta):
    salida="Contenido completo:\n"
    with open(ruta, 'r', encoding='utf-8') as archivo:
        contenido = archivo.read()

    salida+=(contenido)
    return salida
```

Función leerPorSimbolo(ruta): esta función analizar lexicamente cada token y lo guarda en una lista

```
def leerPorSimbolo(ruta):
    global columna,linea
    salida="Analisis lexico:\n"
    with open(ruta, 'r', encoding='utf-8') as archivo:
        for lineas in archivo:
            for caracter in lineas:
                if caracter.isspace():
                    columna=1
                else:
                    simbolo = Simbolo(caracter, linea, columna)
                    simbolos.append(simbolo)
                    columna += 1
            linea+=1

    for simbolo in simbolos:
        salida+=(f"Fila {simbolo.linea}, Columna {simbolo.columna}, Token
    return salida
```

Función leerClaver(): esta función se encargar de leer las claves dentro del archivo, por medio de controles

```
def leerClaves():
    i = 0
    claves = False
    clave = ""
    palabraClave = ""
    sublista = ["C", "l", "a", "v", "e", "s"]
```

Función leerRegistros(): esta función se encargar de leer los registros dentro del archivo, por medio de controles

```
def leerRegistros():
    i = 0
    registros = False
    registro = ""
    palabraregistro = ""
    sublista = ["r", "e", "g", "i", "s", "t", "r", "o", "s"]
```

Función leerImprimir(): esta función se encargar de leer las palabras imprimir dentro del archivo, por medio de controles

```
def leerImprimir():
    i = 0
    imprimir = False
    mensaje = ""
    palabraImprimir = ""
    sublista = ["i", "m", "p", "r", "i", "m", "i", "r","("]
```

Función leerImprimirLn(): esta función se encargar de leer las palabras imprimirln dentro del archivo, por medio de controles

```
def leerImprimirln():
    i = 0
    imprimir = False
    mensaje = ""
    palabraImprimir = ""
    sublista = ["i", "m", "p", "r", "i", "m", "i", "r","l","n"]
```

Clase tablas(): se encarga de hacer la tabla html usando la librería graphviz, desplegando la tabla en el navegador

```
def reporteHtml():
    dot = Digraph(comment='Tabla HTML')

    tabla_html = '''<<TABLE BORDER="1" CELLBORDER="1" CELLSPACING="0">'''
    tabla_html +='''<TR><TD>'''+reporte()+'''</TD></TR>'''

    tabla_html +='''<TR>'''
    for pos in range(len(listaClaves)):
        tabla_html += "<TD>"+str(listaClaves[pos])+ "</TD>"
    tabla_html +='''</TR>'''

    numeroDeFilas=int(((len(listaRegistros))/(len(listaClaves))))
    numeroDeColumnas=int((len(listaClaves)))

    for filas in range(numeroDeFilas):
        tabla_html +='''<TR>'''
        for columna in range(numeroDeColumnas):
            tabla_html += "<TD>"+str(listaRegistros[(filas*numeroDeColumnas)+columna])
        tabla_html +='''</TR>'''

    tabla_html +='''</TABLE>>'''

    dot.node('tabla', label=tabla_html, shape='none')
    dot.render('tabla_tabular', view=True)
```

Clase graficas(): esta función hace el árbol de desviación

```
def grafica():

    dot = Digraph(comment='Árbol Simple')

    dot.node("Claves", "Claves")

    for pos in range(len(listaClaves)):
        dot.node(listaClaves[pos], listaClaves[pos])

    for pos in range(len(listaClaves)):
        dot.edge("Claves", listaClaves[pos])

    dot.node("Registros", "Registros")

    numeroDeFilas=int(((len(listaRegistros))/(len(listaClaves))))
```