Manual Técnico 2022

Estudiante

Nombre: Andy Ezequiel Sanic Tiul

Carnet: 202006699

INDICE

INTRODUCCIÓN

El siguiente trabajo describe los aspectos técnicos informáticos del sistema de información. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

Información destacada

Este manual es con el fin de orientar y referenciar a la información necesaria para orientar al personal, planteamiento del análisis programación e instalación del sistema.
Este manual es referido a personal con conocimientos en programación avanzada, sobre el uso de autómatas, método árbol y construcción de este.

Objetivos

Instruir el uso correcto del Sistema de Información, para el acceso oportuno y continuamente adecuado, descripción de los archivos relevantes del sistema, con los que se podrá orientas en la configuración y soporte de este

REQUERIMIENTOS

El sistema puede ser instalado en cualquier sistema operativo con los siguientes requerimientos:

- Visual Estudio Code v **1.65.1**

Librerias:

- PrettyTable
- Tkinter
- Webbrowser
- 2 GB RAM

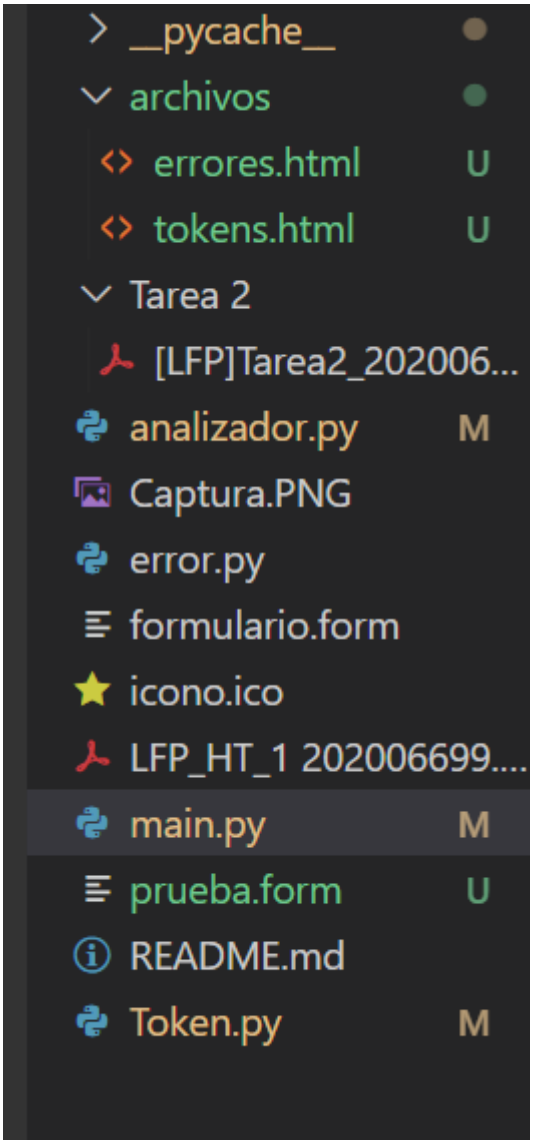
INSTALACIÓN Y CONFIGURACIÓN

Existen muchos métodos de instalación, sin embargo, a continuación, se describe de forma sencilla y segura la instalación, la que genera un link simbólico.

1. Descomprima el rar, para su uso directo.
2. Instale las librerías, especificadas, (PrettyTable, Tkinter, Webbrowser).

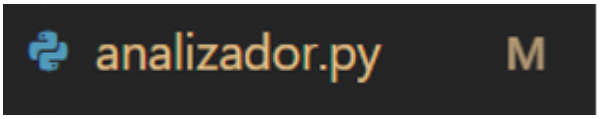
PROGRAMA

Estructura raíz



A continuación, describimos los directorios y archivos más importantes:

Analizador.py



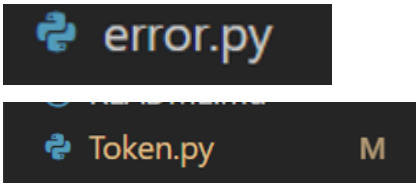
Aquí se hace el autómata, y se desarrolla el árbol generando la estructura del contenido a programar.

```python
from error import Error
from Token import constructor
from prettytable import PrettyTable      You, ayer • …
import os
from tkinter import messagebox
You, hace 28 minutos | 1 author (You)
class AnalizadorLexico():
    def __init__(self):
        self.ListaTokens = []
        self.ListaErrores = []
        self.linea = 1
        self.columna= 0
        self.lexema = "" # strings
        self.estado = 1  #
        self.i = 0
        #tipo, lexema, linea y columna
    # while=?
    def agregar_token(self, caracter, tipo, linea, columna ):
        self.ListaTokens.append(constructor(caracter, linea, columna, tipo))
        self.lexema = ""
    def error_append(self, caracter, linea, columna,  tipo):
        self.ListaErrores.append(Error("caracter: ", caracter, "linea: ", linea, "columna: ", columna, "tipo", tipo))
    #!=====================================================
    #? Entrada de excepciones, simbolos, caracteres, opciones, etc
    def Estado0(self, caracter):…
```

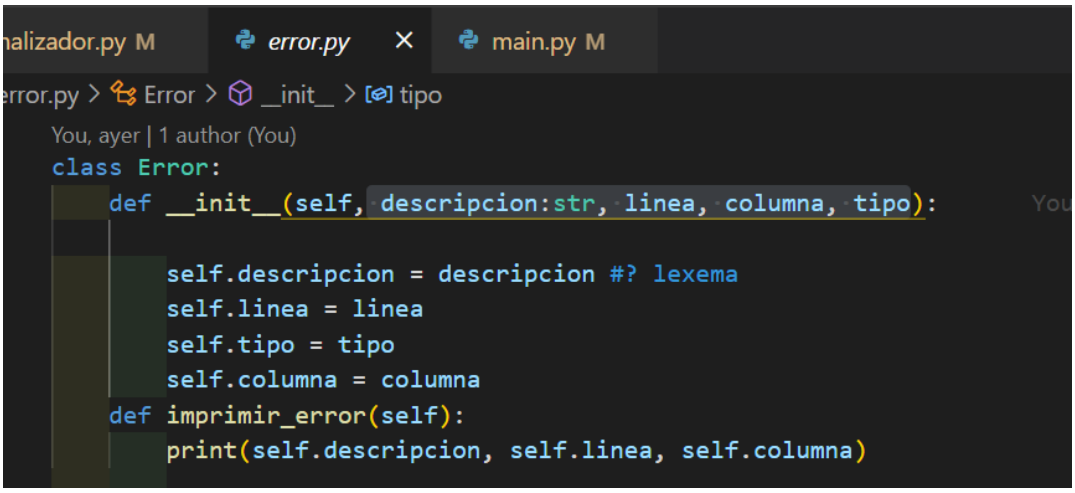
```python
#!=============================================================
#? Entrada de excepciones, simbolos, caracteres, opciones, etc
def Estado0(self, caracter):…

#!=============================================================
#? Estados para ingreso de cadenas, indicadores y reservadas, excepciones de cadenas
def Estado1(self, caracter :str):…

#!=============================================================
#? Estados de digitos enteros
def Estado2(self, caracter :str):…

#!=============================================================
#? Estado para creación de cadenas
def Estado3(self, caracter : str):…


def Estado7(self, caracter: str):
    if caracter == '"' or caracter == "'":
        self.estado = 7
        self.lexema += caracter
    else:
        self.agregar_token(self.lexema, "Cadena", self.linea, self.columna)
        self.estado = 0
        self.i -= 1
#!=============================================================
#? Estado para creación de simbolos
def Estado4 (self, caracter : str):…
```

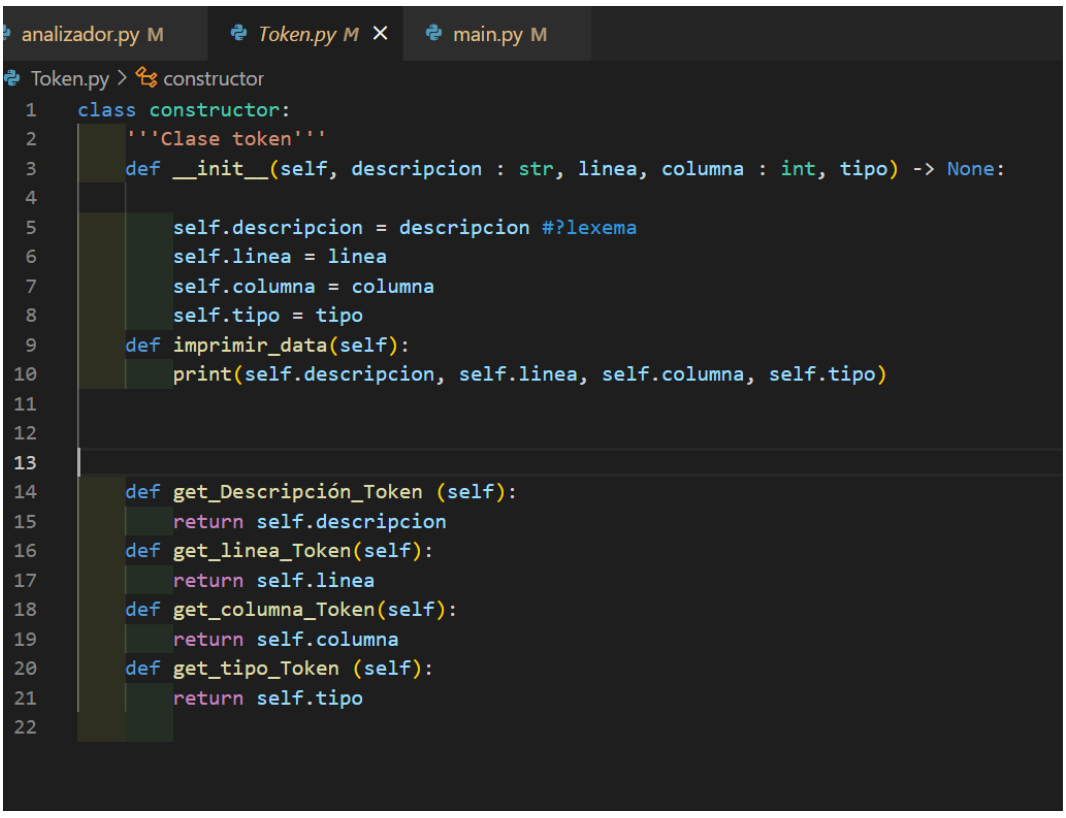
```python
    #? Estados para creación de decimales
    def Estado6(self, caracter: str):…
    #!=====================================================
    #? Estados para creación de decimales




    def nofunciona(self):…

    def Analizar(self, caracter):…

    def imprimirTokens(self):…

    def imprimirErrores(self):…



    def iniciar(self):…
    def guardar(self, name: str, cadena: str, abrir: bool = True ):  #? es una libreria por defecto, sirve para manej
    def busqueda(self, codigo ):…

    def Tabla_tokens(self, cadena, nombre_tab):…
```

Error.py



Se genera nuestras estructuras, constructor y clases correspondientes.





main.py

Aquí importamos nuestras funciones, para ejecutar nuestra aplicación.

```python
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from tkinter.filedialog import askopenfile
from tkinter import filedialog, Tk
import sys
from tkinter.scrolledtext import ScrolledText
from analizador import AnalizadorLexico
import webbrowser       You, hace 32 minutos • Uncommitted changes
lexico = AnalizadorLexico ()

def ventana_creacion():
    #!ttk son los componentes
    #? creamo nuestro objeto
    ventana = tk.Tk()
    #? modificamos el tamaño de la ventana
    ventana.geometry('650x550')
    #? titulo de la ventana
    ventana.title("Interfaz Gráfica")
    #? configuramos nuestro icono de nuestra app
    ventana.iconbitmap("icono.ico")
    ventana.resizable(0,0) #? evitamos que cambie de tamaño
    ventana.config(background= "#6898FD")
```

```
#=====================================================
entrada = Label(text = "Menu",  font = ("Cambria", 14), bg= "#FFFFFF", width = "500", height = "2")
#!           Label
ventana.rowconfigure(0, weight=1)

# Ingr_entrada = tkinter.Label(ventana, text = "             Bienvenido            ").pack()


#! ========================================
#? creación de nuestra ventana


boton_analizar = Button(ventana, text  = "Analizar", command = analizar, width="30", height="2", bg = "#00FF58" )
boton_analizar.place(x=22, y =450)

boton_enter = Button(ventana, text = "Enter", command = enter_entrada, width="5", height="1", bg ="#00FF58")
boton_enter.place( x = 500, y = 150)

boton_cargar = Button(ventana, text = "Cargar", command= abrir, width= "30", height= "2", bg = "#00FF58" )
boton_cargar.place(x= 350, y = 450)
global opcion_escogida
opcion_escogida = tk.StringVar()
global combo
combo = ttk.Combobox( textvariable= opcion_escogida)
combo = ttk.Combobox( values=[
                        "Reporte de Tokens",
                        "Reporte de errores",
                        "Manual",
                        "Manual Técnico"])
```

```
  # combo.bind("<<ComboboxSelected>>", op_uso())

  combo.place(x=500, y=75)
  #!=====================================================
  global caja_texto

  caja_texto = ScrolledText(ventana, width = "50", height= "20")
  caja_texto.place(x=40, y = 60)
  #? utilizamos nuestro pack layout manager
  #? uso del boton de la ventana
  #boton_1.grid(row = 1, column=0, sticky="SE")
  #! advertencia: este metodo debe ser el último
  entrada.pack()
  ventana.mainloop()
def llamadas(event):
    if combo.get() == "Reporte de Tokens":
        lexico.imprimirTokens()
    # if opcion != None:
    #     messagebox.showinfo("Error", "Carga tu información")

  # if opcion_escogida.get() == "Reporte de Tokens":
  #     combo.set()
  #     if formato != None
  #     analizar()
def tenemos(codigo: str):
    if len (codigo) > 0:
       global datos_analic
       datos_analic = AnalizadorLexico()
       datos_analic.busqueda(codigo)
```

```
def analizar():
    #!traemos nuestros datos para manipulación y realización de tablas
    a = caja_texto.get("1.0", tk.END)
    lexico.iniciar()
    lexico.Analizar(a)
#     lexico.imprimirTokens()
#     lexico.imprimirErrores()



    # modificamos el boton al tocarlo
    #boton_1.config(text = "Ingresando")
    #!creación de botones
#boton_1 = ttk.Button(ventana, text = "valor", command= evento_click)
def enter_entrada():
    #!================================================
    if combo.get() == "Reporte de Tokens":
        combo.set("")
        if lexico.ListaTokens != None:
            lexico.imprimirTokens()
        else:
            print("No tenemos información")

    elif combo.get() == "Reporte de errores":
        combo.set("")
        if lexico.ListaErrores != None:
            lexico.imprimirErrores()
```

```
def enter_entrada():
    #!================================================
    if combo.get() == "Reporte de Tokens":
        combo.set("")
        if lexico.ListaTokens != None:
            lexico.imprimirTokens()
        else:
            print("No tenemos información")

    elif combo.get() == "Reporte de errores":
        combo.set("")
        if lexico.ListaErrores != None:
            lexico.imprimirErrores()
        else:
            print("No tenemos información")
    elif combo.get() == "Manual":
        combo.set("")
        webbrowser.open_new(r"C:\Users\andye\OneDrive\Documentos\repositorios\LFP_PY1_202006699\Tarea 2\[LFP]Tarea2_2
        abrir()

    # selection = combo.get()
    # #? creacion de mensajes
    # messagebox.showinfo(
```

```
def abrir():
    Tk().withdraw()
    archivo = filedialog.askopenfile(
        title = "Seleccionar un archivo",
        initialdir = "./",
        filetypes = [
            #Definimos los tipo de archivo
            ("archivos .form", "*.form"),
            ("todos los archivos",  "*.*")
        ]
    )
    #si no se seleccióno ningun archivo
    if archivo is None:
        print('No se selecciono ningun archivo\n')
        return None
    else:
        archivo_2 = open(archivo.name, "r", encoding = "utf-8")
        #Leer el texto
        texto = archivo_2.read()
        archivo_2.close()
        caja_texto.delete("1.0", "end")
        caja_texto.insert(INSERT, texto)
        return texto #retorna nuestro texto
```

```
def salir():
    sys.exit()

if __name__ == '__main__':

    archivo_prueba = '''Hola mundo ~>> [···
    ventana_creacion()
    # lexico.Analizar(txt)
    # for token in lexico.ListaTokens:
    #     print(len(token.))
    # for token in lexico.ListaErrores:
    #     print(len(lexico.error_append()))

    lexico.Tabla_tokens()
    print(len(lexico.ListaTokens))
    print(len(lexico.ListaErrores))
```

Variable	Descripción	Paquete	Clase

`ventana = tk.Tk()`
`entrada`
`boton_analizar`
`boton_enter`
`boton_cargar`
`combo`
`caja_texto`
	Creación de la ventana	Tkinter	Ttk
			Filedialog
			Scrolledtext

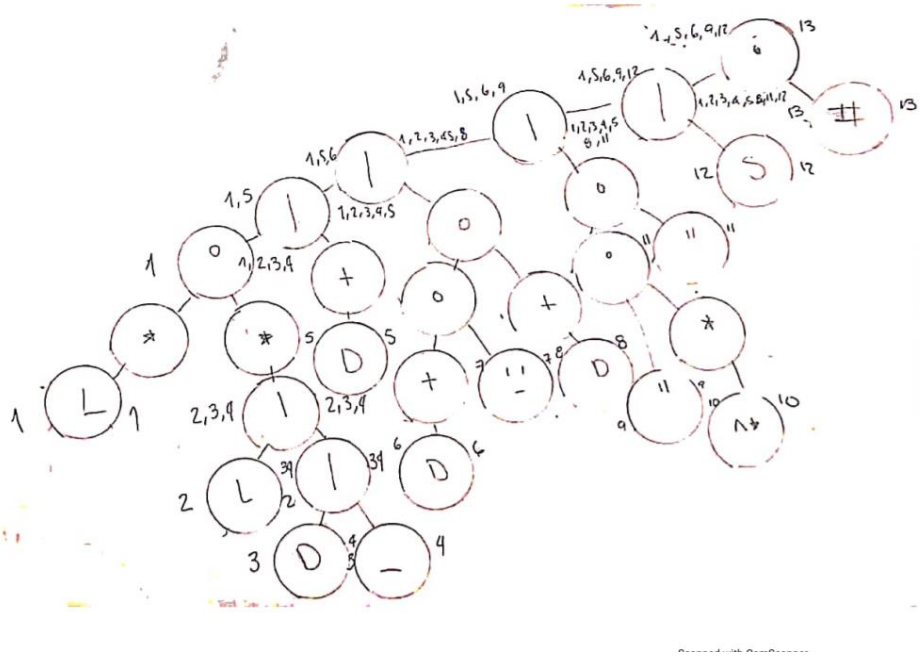
`self.ListaTokens = []`
`self.ListaErrores = []`
`self.linea = 1`
`self.columna= 0`
`self.lexema = "" # strings`
`self.estado = 1 #`
`self.i = 0`
	Creación de analizador, aquí iremos guardando nuestros datos e ingresando nuestros registros.	Python	__init__

`self.descripcion = descripcion`
`#?lexema`
`self.linea = linea`
`self.columna = columna`
`self.tipo = tipo`
	Creación de nuestros tokens, estas variables serán de uso para nuestro constructor	Python	__init__

`self.descripcion = descripcion #?`
`lexema`
`self.linea = linea`
`self.tipo = tipo`
`self.columna = columna`
	Creación de nuestros tokens, estás variables serán de uso para nuestros errores	Python	

ANEXOS

Creación de árbol



Token	Patron	ER	Ejemplo
Identificadores	etiqueta, texto, grupo-radio, grupo-option, otón, EVENTO	(L(L|D|”_”|*)+L	asadon2, asado_, asadon23_
Palabras reservadas	formulario, tipo, valor, fondo, nombre, valores, evento, seguida de letras minusculas	L(L|D|_)*	asadon2, asado_, asadon23_
Simbolos	. - <> , “ ‘ : _	((“)(^”)*(“))	. ,] [
cadenas	**Nombre, Ingrese Nombre, Guatemala, El salvador, Honduras, Evento**	(L(L|D|”_"|*)+L	asadon2, asado_, asadon23_
Cadenas	L, D	(D+)|(D+'.'D+)|S	91231.3

Valor	Hoja	Siguientes
L	1	2,3,4,13
L	2	2,3,4,13
D	3	2,3,4,13
_'	4	2,3,4,13
D	5	5
D	6	6,7
,'	7	8
D	8	8
"	9	10,11
^"	10	10,11
"	11	13
S	12	13
#	13	_____

Estados

	ESTADOS	L	D	_'	.'	"	^"	S
0	S0	S1	S2	-	-	S3		S4
$	S1	S1	S1	S1	-	-	-	-
$	S2	-	S2	-	S5	-	-	-
	S3	-	-	-	-	S4	S3	-
$	S4	-	-	-	-	-	-	-
	S5	S6	-	-	-	-	-	-
$	S6	S6	-	-	-	-	-	-

Autómata