

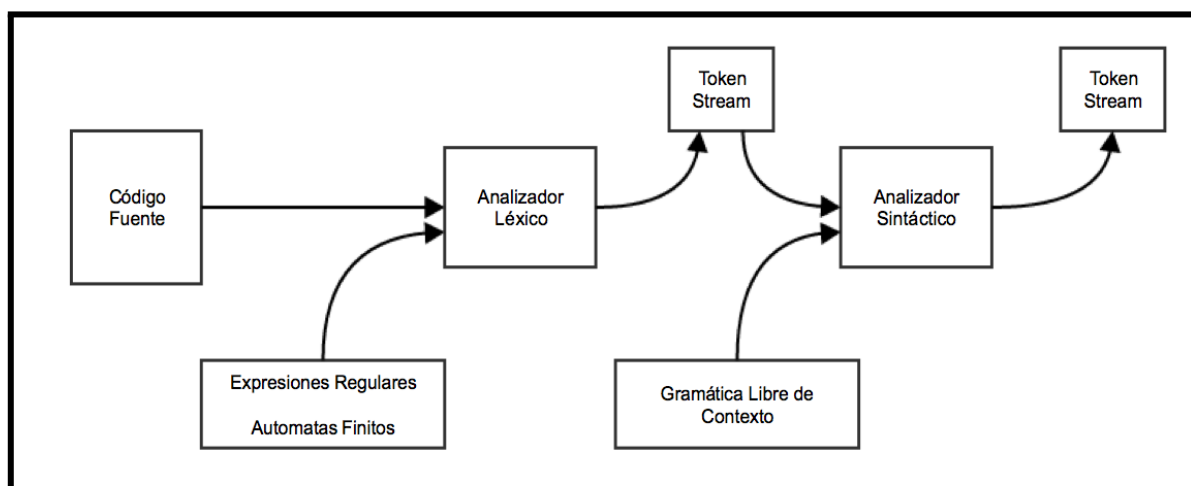


# Actividad 5

## Tema 6

### ***“Presentación y Entrega Final del Proyecto”***

22 de junio del 2022



Repositorio: [https://github.com/pfr36/LYA\\_REPO](https://github.com/pfr36/LYA_REPO)

Video: <https://youtu.be/c1dTHocRf10>

Figueroa Ruiz Pedro (19400568)

Lopez Burgara Marco Antonio (19400608)

Carlos Ismael Orozco Rios (19400609)

## Tabla de contenidos

<a href="#">Tabla de contenidos</a>	<a href="#">2</a>
<a href="#">Abstract</a>	<a href="#">3</a>
<a href="#">Introducción</a>	<a href="#">4</a>
<a href="#">Justificación</a>	<a href="#">8</a>
<a href="#">Metodología</a>	<a href="#">8</a>
<a href="#">Desarrollo del proyecto</a>	<a href="#">11</a>
<a href="#">Manual de Usuario</a>	<a href="#">44</a>
<a href="#">Manual Técnico</a>	<a href="#">51</a>
<a href="#">Correr el compilador de forma directa a través del ejecutable</a>	<a href="#">53</a>
<a href="#">Conclusión</a>	<a href="#">55</a>
<a href="#">Glosario</a>	<a href="#">56</a>

**Abstract**

This document contains the early specifications for the development of the preliminary project “Buena Movilidad Wheelchairs”, with the main objective of improving the general quality of life of physically disabled or afflicted people. It aspires to do so by implementing certain features and capabilities to a wheelchair, which will be activated by voice commands spoken by its user. These features include the ability to autonomously move the wheelchair to a specific place, providing and receiving feedback regarding the user’s surroundings, as well as facilitating interactions with appliances with the aid of wireless technological devices.

## Introducción

Durante el desarrollo de este documento se presentará nuestro proyecto desarrollado ‘Buena Movilidad Wheelchairs’. El principal enfoque de nuestro proyecto consta en la automatización de una silla de ruedas común y corriente, con lo que lograremos conseguir que el usuario de la silla de ruedas tenga una experiencia más cómoda al permitirle ciertas funciones que le facilitarán la realización de necesidades básicas, tales como: movimiento de la silla, activación de freno, activación de alertas para facilitar el traslado del usuario, equipo de iluminación para facilitar la movilidad, envío de mensaje de auxilio a contacto en caso de incidente, desplazamiento a ciertas áreas definidas por el usuario entre otras funciones, todas estas funcionando por medio de comandos de voz, por lo que el usuario solo tiene que dar una orden para poder ejercer una acción.

También se presenta el análisis, diseño y construcción de un analizador léxico y sintáctico para el compilador del anteproyecto de la materia Lenguajes y Automatas I, esto se realizó mediante el lenguaje de programación Python con ciertas librerías llamadas ply.lex las cuales nos serán de mucha utilidad puesto que tienen un parecido al mismo lenguaje Python pero combinado con Java.

## Antecedentes

### Situación actual del problema.

Las sillas de ruedas son un dispositivo que proporciona apoyo y movilidad a una persona con dificultades para caminar o desplazarse. Una de las problemáticas más actuales y comunes son las personas que usan sillas de ruedas no aptas para lo que ellos requieren y que termina influyendo en el potencial de su uso. Y esto se debe en mayor medida a lo difícil que es maniobrar la silla en habitaciones estándar, donde se lidia con puertas muy estrechas, escaleras, etcétera.

Además entre otros impedimentos que se detectó incide directamente en el incremento de dolor, es la deformación de espalda y el sentadero debido al material plástico que se desgasta con el uso y el tiempo.

Otro conflicto comentado por los usuarios es la incompatibilidad que se genera entre la silla de ruedas y la andadera. Lo anterior evidencia que las andaderas simplemente no están diseñadas para usarse de manera complementaria con una silla de ruedas convencional.

«[...] tengo que agarrarme bien de la andadera y hacer más fuerza porque la silla choca con la andadera, no se acerca y es muy incómodo pararse» (A., usuaria mexicana con Artritis Reumatoide, ama de casa).

### Elaboración de los antecedentes.

“Se considera persona con capacidades diferentes a todo ser humano que presente temporal o permanentemente una limitación, pérdida o disminución de sus facultades físicas, intelectuales o sensoriales, para realizar sus actividades connaturales.” (Rodallegas Hinojosa, 2004)

"Se reconoce que las personas con capacidades diferentes sufren marginación y discriminación, no solo por parte de la sociedad, sino también en ocasiones de su familia, lo que les lleva a tener, además de un problema físico, baja autoestima.." (Rodallegas Hinojosa, 2004)

(J Rehabil., 2005) "Independent mobility increases vocational and educational opportunities, reduces dependence on caregivers and family members, and promotes feelings of self-reliance"

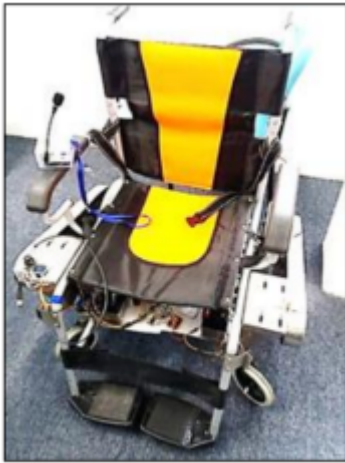


“Una de las primeras decisiones que tuve que tomar fue si quería llevar una silla de ruedas eléctrica o una manual. Escogí la manual porque pensé que me iría bien hacer ejercicio, pero enseguida detecté un problema: en interiores me manejaba bien pero para desplazarme en espacios abiertos no tenía suficiente fuerza en los brazos, la movilidad era casi nula”. Por eso, Bach se puso manos a la obra y en 2004 ya había diseñado un primer prototipo de handbike: “Conseguí independencia, me cambió la vida, y pensé que aquello le podía servir a muchas otras personas”. (Humexe, 2020). En la actualidad existen sillas de ruedas que permiten a los discapacitados poder levantarse, tal es el caso de la “EVO Altus” ganadora de los premios RED DOT en 2020.



“Si vivir de forma independiente es el objetivo final, ser capaz de levantarse es un paso importante. La EVO Altus (EVO-ALT) está especialmente diseñada para ayudar a aquellos que tienen el control corporal muy limitado para lograr este paso. La silla proporciona no sólo una, sino tres maneras diferentes de levantarse: sentarse a pie, reclinado a pie y recostado a pie. Los

modos de parada junto con otras características son configurables para ajustarse a la necesidad del usuario y para ayudar al usuario a alcanzar la independencia en las ruedas” (KARMA Medical, 2018)



Tan Kian Hou, Yagasena y Chelladurai de la Universidad Internacional Quest en Malasia diseñaron un prototipo de una silla de ruedas controlada por comandos de voz, usando una silla convencional y un microcontrolador Arduino. El módulo de reconocimiento de voz es la característica clave de este proyecto que se utiliza para configurar la voz deseada como comando y salida. Consta de tres fases, que son personalización de voz, captura de voz y reconocimiento de voz. La personalización de voz es el proceso de hacer coincidir la voz deseada grabada con la señal de salida deseada.

La captura de voz es la fase que registra el comando de voz de la persona deseada y guarda la voz según la configuración de personalización.

“The advances in speech recognition technology have made it possible to control any electronics based device using voice command. This technology is capitalized for voice controlled wheelchair to assist those with both upper and lower limb disabilities.” (Tan Kian Hou, 2020)



### Justificación

De acuerdo con el Censo de Población y Vivienda 2020, en México hay 6,179,890 personas con algún tipo de discapacidad, lo que representa 4.9 % de la población total del país. Nuestro proyecto, “Buena Movilidad Wheelchairs”, pretende ofrecer los medios para mejorar la calidad de vida de este sector importante de la población.

Los beneficios más importantes que ofrece nuestro proyecto incluyen pero no se limitan a:

- La persona usuaria de la silla es menos dependiente y como consecuencia de esto aumentaría su confianza en sí misma, ya que se dan casos en que las personas se sienten como “cargas” para sus familias o muy limitados a lo que hacen.
- A diferencia de otras sillas de ruedas convencionales, es cómoda, haciendo que no ocurran efectos secundarios al usar la silla tales como dolor de espalda, desnivelación de cadera, etc.
- La persona que utilice esta silla tendrá a su disposición la posibilidad de alertar a las personas cercanas o lejanas a él ante cualquier tipo de emergencia ya que con ella podremos saber su ubicación en tiempo real y con alertas de auxilio mediante sus bocinas.
- A diferencia de otras sillas convencionales la persona contará con una mayor seguridad ante posibles peligros o accidentes que conlleva transportarse en una silla por una ciudad o lugares muy concurridos.



## Metodología

Funcionalidades:

The development of these functionalities is made to facilitate the daily activities of its user, as well as to improve their quality of life.

1) Escribir la lista de funcionalidades que tendrá el proyecto.

Silla de Ruedas:

1. Movilidad automática dirigida por comandos de voz; el usuario podrá especificar la distancia así como los grados.
2. Sensor de proximidad; la silla tendrá un sensor que haga que la silla se detenga si es que detecta algún obstáculo.
3. Encender y apagar luces; estas estarán integradas en la silla, el usuario podrá encenderla/apagarla cuando sea necesario por medio de un comando de voz.
4. Masajeador; cuando el usuario tenga la necesidad puede activar un masajeador que le proporcionará un masaje en la espalda.
5. Mandar mensaje de auxilio; cuando se encuentre el usuario en una situación de emergencia tiene la opción de contactar por medio de un comando de voz a un contacto del usuario.
6. Alerta de paso; en las situaciones de contingencia el usuario puede activar un sonido de alerta para que las personas a su alrededor se percaten de su presencia a través de un comando de voz con ayuda de unas bocinas integradas.
7. Poner música; la silla reproducirá la música que el usuario solicite por medio de voz en las bocinas integradas.
8. Acercamiento; el usuario puede acercarse a ciertos objetos especificados con la ayuda de un bluetooth beacon especificando por voz el nombre del objeto previamente configurado.

9. Luces por proximidad; en las horas donde ya no hay luz, el prescindir de los contactos es de gran ayuda para personas con movilidad limitada por lo que esta función enciende las luces de las habitaciones cuando la silla se acerca a estas por medio de la tecnología bluetooth beacons.

10. Tomar la temperatura del usuario; Nos permitirá realizar un diagnóstico general con respecto al estado en el que se encuentra la persona por medio de la toma de temperatura por medio de un sensor.

## Desarrollo del proyecto

El lenguaje está en base a python y con las librerías de ply.lex, con un parecido al mismo lenguaje de python, combinado con java.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

### B) Definir su tabla de tokens y lexemas.

Token	Descripción	Ejemplo
PalabrasReserva	Son las cadenas que identifican las funciones reservadas	IMPORT, STRING,INT
Operadores	Caracteres +, -, /, *, =	+ ó - ó / ó * ó =
Comparación	Caracteres ==, !=, <, >, >=, <=	== ó != ó < ó > ó >= ó <=
Separadores	Caracteres ., :, ;, ' "	. ó , ó : ó ; ó ' ó "
AbiertoCerrado	Caracteres (, ), {, }, [, ]	( ó ) ó { ó } ó [ ó ]
Pasos	Caracteres ++, -- (incremento decremento)	++ ó --
id	Letra seguida por letras y numeros	Contador1, ResultadoMul
número	Cualquier dígito	1, 4892, 57, 192
SaltoLinea	Caracteres \n indicando el salto de línea	\n
Comentario	Comentario de línea indicado por #	#Comentario, # Ola
Error	Cualquier caracter no valido	ñ, †, ~

### C) Definir las instrucciones (sintaxis) de su lenguaje.

Palabra reservada	Descripción	Ejemplo
IMPORT	Cadena mayus y min I,M,P,O,R,T	IMPORT ó import

DEF	Cadena mayus y min D, E, F	DEF ó def
CLASS	Cadena mayus y min C, L, A, S, S	CLASS ó class
IF	Cadena mayus y min I, F	IF ó if
ELSE	Cadena mayus y min E, L, S, E	ELSE ó else
FOR	Cadena mayus y min F, O, R	FOR ó for
IN	Cadena mayus y min I, N	IN ó in
RANGE	Cadena mayus y min R, A, N, G, E	RANGE ó range
SELF	Cadena mayus y min S, E, L, F	SELF ó self
WHILE	Cadena mayus y min W, H, I, L, E	WHILE ó while
TRY	Cadena mayus y min T, R, Y	TRY ó try
EXCEPT	Cadena mayus y min E, X, C, E, P, T	EXCEPT ó except
RETURN	Cadena mayus y min R, E, T, U, R, N	RETURN ó return
BREAK	Cadena mayus y min B, R, E, A, K	BREAK ó break
NEXT	Cadena mayus y min N, E, X, T	NEXT ó next
INPUT	Cadena mayus y min I, N, P, U, T	INPUT ó input
PRINT	Cadena mayus y min P, R, I, N, T	PRINT ó print
INT	Cadena mayus y min I, N, T	INT ó int
FLOAT	Cadena mayus y min F, L, O, A, T	FLOAT ó float
BOOLEAN	Cadena mayus y min B, O, O, L, E, A, N	BOOLEAN ó boolean
STRING	Cadena mayus y min S, T, R, I, N, G	STRING ó string
POWER	Cadena mayus y min P, O, W, E, R	POWER ó power
SQRT	Cadena mayus y min S, Q, R, T	SQRT ó sqrt
AND	Cadena mayus y min A, N, D	AND ó and
OR	Cadena mayus y min O, R	OR ó or
NOT	Cadena mayus y min N, O, T	NOT ó not

#### D) Definir las reglas del lenguaje.

1. Las palabras reservadas deben ser rigurosamente escritas ó todas en mayúsculas ó todas en minúsculas para considerarse como tal
2. Los caracteres inválidos (No añadidos al alfabeto) son catalogados como caracteres inválidos

3. Los identificadores deben comenzar por una letra mayúscula o minúscula, seguida de letras y/o números.
4. Todo carácter abierto ( (, {, [ ) deberá tener su carácter para cerrar.
5. Los operadores, deben de ser seguidos de números

#### E) Definir los patrones válidos.

A continuacion se presentan varios patrones válidos del lenguaje.

```
Resultado = 5 + 3;
```

```
Resultado++;
```

```
Resultado = 5 + 3; \n Resultado++;
```

```
Numero = INPUT #Comentario
```

```
Resultado=5+3;
```

```
DEF Funcion1: \n RETURN 1
```

#### F) Diseñar los autómatas.

Para generalizar las palabras reservadas de nuestro lenguaje, como ya está definida, para ser considerada como tal, debe ser todas en mayúsculas o todas minúsculas, por ende, serán mostrados dos ejemplos de dichos autómatas, para que se pueda intuir como serán para el resto de palabras reservadas.

Automata theory is a branch of computational theory that studies abstract machines and the problems they are capable of solving. Automata theory is closely related to formal language theory since automata are often classified by the class of formal languages they are able to recognize. They are also very useful in computational complexity theory.

### G) Identificar los Errores Léxicos.

Los errores léxicos son aquellos caracteres que no pertenecen al alfabeto del lenguaje, o aquellos que no coinciden con ningún patrón de los tokens. En este caso tenemos por ejemplo:

ñ	'	~	^
ı	?	☀	◆
♀	\$	%	&

Entre estos y algunos otros caracteres que no están disponibles en el lenguaje definido

**2) Desarrollar (codificar) el Analizador Léxico de su proyecto en el lenguaje de programación elegido en la asignatura.**

```
#Importaciones
import lex
import re
import codecs
import os
import sys
from tkinter import *

#Lista de Tokens que usaran los metodos
    #Basicas
tokens = ['ID', 'NUMERO', 'SUMA', 'ASIGNACION', 'RESTA', 'DIVISION', 'MULTIPLICACION',
    #Operadores

'IGUAL', 'DIFERENTE', 'MAYORQUE', 'MENORQUE', 'MAYORIGUALQUE', 'MENORIGUALQUE',
    #Separadores

'PUNTO', 'COMA', 'DOSPUNTOS', 'PUNTOCOMA', 'COMILLASIMPLE', 'COMILLADOBLE', 'PARENTESIS
ABIERTO',

'PARENTESISCERRADO', 'LLAVEABIERTO', 'LLAVECERRADO', 'CORCHETEABIERTO', 'CORCHETECERR
ADO', 'ESPACIO',

    #Pasos
    'INCREMENTO', 'DECREMENTO']

#Diccionario de palabras reservadas
reservadas = {
    'import': 'IMPORT',
    'def': 'DEF',
    'class': 'CLASS',
    'if': 'IF',
    'else': 'ELSE',
    'for': 'FOR',
    'in': 'IN',
    'range': 'RANGE',
    'self': 'SELF',
    'while': 'WHILE',
    'try': 'TRY',
    'except': 'EXCEPT',
```

```
'return': 'RETURN',
'break': 'BREAK',
'next': 'NEXT',

'input': 'INPUT',
'print': 'PRINT',
'int': 'INT',
'float': 'FLOAT',
'boolean': 'BOOLEAN',
'string': 'STRING',

'powew': 'POWER',
'sqrt': 'SQRT',
'and': 'AND',
'or': 'OR',
'not': 'NOT'
}

tokens = tokens + list(reservadas.values())

t_ignore = '\t'

#Operadores matematicos
t_SUMA = r'\+'
t_ASIGNACION = r'='
t_RESTA = r'\-'
t_DIVISION = r'/'
t_MULTIPLICACION = r'\*'

#Operadores
t_IGUAL = r'=='
t_DIFERENTE = r'!='
t_MAYORQUE = r'>'
t_MENORQUE = r'<'
t_MAYORIGUALQUE = r'>='
t_MENORIGUALQUE = r'<='

t_PUNTO = r'\.'
t_COMA = r','
t_DOSPUNTOS = r':'
t_PUNTOCOMA = r';'
```



```
t_COMILLASIMPLE = r'\''
t_COMILLADOBLE = r'\"'
t_PARENTESISABIERTO = r'\('
t_PARENTESISCERRADO = r'\)'
t_LLAVEABIERTO = r'\{'
t_LLAVECERRADO = r'\}'
t_CORCHETEABIERTO = r'\['
t_CORCHETECERRADO = r'\]'
t_ESPACIO = r'\s'

t_INCREMENTO = r'\++'
t_DECREMENTO = r'\--'

def t_ID(t):
    r'[a-zA-Z_][a-zA-Z0-9_]*'
    if t.value.upper() in reservadas:
        t.value = t.value.upper()
        t.type = t.value
    return t

def t_NUMERO(t):
    r'\d+'
    t.value = int(t.value)
    return t

def t_SALTOLINEA(t):
    r'\n+'
    t.lexer.lineno += t.value.count("\n")

def t_COMENTARIO(t):
    r'\#.*'
    pass

def t_error(t):
    t.lexer.skip(1)
    return "Caracter ilegal"

a = []

def analiza(cadena):
    analizador = lex.lex()
```

```
analizador.input(cadena)
a.clear
while True:
    tok = analizador.token()
    if not tok : break
    a.append(str(tok))
return a

#Aqui va el texto a analizar
cadena1 = "Resultado = 5 + 3; \n Resultado++; \n Numero1 = INPUT #Comentario \n
DEF Funcion1: \n RETURN 1"
analiza (cadena1)

print("Cadena analizada: \n\n" + cadena1)

print("\n\n====Tokens====")
for i in a:
    print(i)
    print("")
```

## Resultados

```
Cadena analizada: Cadena = 5 + 3
Cadena++;
Tokens
LexToken(ID, 'Cadena', 1, 0)
LexToken(ESPACIO, ' ', 1, 6)
LexToken(ASIGNACION, '=', 1, 7)
LexToken(ESPACIO, ' ', 1, 8)
LexToken(NUMERO, 5, 1, 9)
LexToken(ESPACIO, ' ', 1, 10)
LexToken(SUMA, '+', 1, 11)
LexToken(ESPACIO, ' ', 1, 12)
LexToken(NUMERO, 3, 1, 13)
LexToken(ESPACIO, ' ', 1, 14)
LexToken(ESPACIO, ' ', 2, 16)
LexToken(ID, 'Cadena', 2, 17)
LexToken(INCREMENTO, '++', 2, 23)
LexToken(PUNTOCOMA, ';', 2, 25)

Cadena analizada: a = b + 40 $
Tokens
LexToken(ID, 'a', 1, 0)
LexToken(ESPACIO, ' ', 1, 1)
LexToken(ASIGNACION, '=', 1, 2)
LexToken(ESPACIO, ' ', 1, 3)
LexToken(ID, 'b', 1, 4)
LexToken(ESPACIO, ' ', 1, 5)
LexToken(SUMA, '+', 1, 6)
LexToken(ESPACIO, ' ', 1, 7)
LexToken(NUMERO, 40, 1, 8)
LexToken(ESPACIO, ' ', 1, 10)
Caracter ilegal
```

```
Resultado = 5 + 3;
Resultado++;
Numero1 = INPUT #Comentario
DEF Funcion1:
RETURN 1
=====Tokens=====
LexToken(ID, 'Resultado', 1, 0)
LexToken(ESPACIO, ' ', 1, 9)
LexToken(ASIGNACION, '=', 1, 10)
LexToken(ESPACIO, ' ', 1, 11)
LexToken(NUMERO, 5, 1, 12)
LexToken(ESPACIO, ' ', 1, 13)
LexToken(SUMA, '+', 1, 14)
LexToken(ESPACIO, ' ', 1, 15)
LexToken(NUMERO, 3, 1, 16)
LexToken(PUNTOCOMA, ';', 1, 17)
LexToken(ESPACIO, ' ', 1, 18)
LexToken(ESPACIO, ' ', 2, 20)
LexToken(ID, 'Resultado', 2, 21)
LexToken(INCREMENTO, '++', 2, 30)
LexToken(PUNTOCOMA, ';', 2, 32)
LexToken(ESPACIO, ' ', 2, 33)
LexToken(ESPACIO, ' ', 3, 35)
LexToken(ID, 'Numero1', 3, 36)
LexToken(ESPACIO, ' ', 3, 44)
LexToken(ASIGNACION, '=', 3, 45)
LexToken(ESPACIO, ' ', 3, 46)
LexToken(ID, 'INPUT', 3, 47)
LexToken(ESPACIO, ' ', 3, 52)
LexToken(ESPACIO, ' ', 4, 66)
LexToken(ID, 'DEF', 4, 67)
LexToken(ESPACIO, ' ', 4, 70)
LexToken(ID, 'Funcion1', 4, 71)
LexToken(DOSPUNTOS, ':', 4, 79)
LexToken(ESPACIO, ' ', 4, 80)
LexToken(ESPACIO, ' ', 5, 82)
LexToken(ID, 'RETURN', 5, 83)
LexToken(ESPACIO, ' ', 5, 89)
LexToken(NUMERO, 1, 5, 90)
PS C:\Users\solda\OneDrive\Documentos\Visual Studio Code\AnalizadorLexico>
```

## Análisis Sintáctico:

```
#Arreglo donde se almacenan los errores
Errores = []

#===== Buscar entre los tokens =====
def BuscarS(tokens):
    global llavesFaltantes
    llavesFaltantes = 0
    #Limpiar el arreglo de errores
    Errores.clear()

    i=0
    while i<len(tokens):
        if tokens[i].type == "INT":
            gINT(tokens,i)
        if tokens[i].type == "FLOAT":
            gFLOAT(tokens,i)
        if tokens[i].type == "BOOLEAN":
            gBOOLEAN(tokens,i)
        if tokens[i].type == "STRING":
            gSTRING(tokens,i)
        if tokens[i].type == "IF":
            gIF(tokens,i)
        if tokens[i].type == "WHILE":
            gWHILE(tokens,i)
        if tokens[i].type == "FOR":
            gFOR(tokens,i)
        if tokens[i].type == "DEF":
            gDEF(tokens,i)
        if tokens[i].type == "CLASS":
            gCLASS(tokens,i)
        if tokens[i].type == "POWER":
            gPOWER(tokens,i)
        if tokens[i].type == "SQRT":
            gSQRT(tokens,i)
        if tokens[i].type == "MULTIPLICACION":
            gAritmeticos(tokens,i)
        if tokens[i].type == "DIVISION":
            gAritmeticos(tokens,i)
        if tokens[i].type == "SUMA":
```

```
        gAritmeticos(tokens,i)
    if tokens[i].type == "RESTA":
        gAritmeticos(tokens,i)

    if tokens[i].type == "LLAVECERRADO":
        llavesFaltantes-=1
    i+=1

#despues del ciclo, revisa que no falten o sobren llaves
if llavesFaltantes>0 and (not Errores):
    Errores.append('Faltaron '+str(llavesFaltantes)+' llaves de cerradura.')
if llavesFaltantes<0 and (not Errores):
    Errores.append('Sobran '+str(abs(llavesFaltantes))+' llaves de
cerradura.')
```

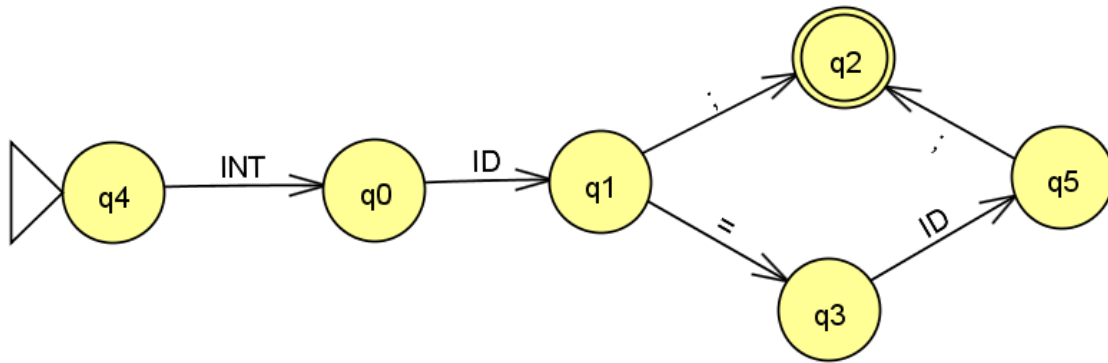
```
#===== Autómatas =====
#===== INT =====
def gINT(t,i):
    #Cantidad de elementos len(t)
    if(len(t)>i+1):
        if(t[i+1].type != 'ID'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ID | Linea:'+str(t[i+1].lineno) + '
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataINT.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID |
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataINT.png')
            return
    #Se cumplio la primera condicion ->
    if(len(t)>i+2):
        if(t[i+2].type == 'ASIGNACION'):
            gAsign(t,i+2)
        elif(t[i+2].type != 'PUNTOCOMA'):
            Errores.append('Token Inesperado:'+str(t[i+2].value)+', se esperaba
```

```
otro tipo de token: ; o = | Linea:'+str(t[i+2].lineno) +'
Columna:'+str(t[i+2].lexpos))
    Errores.append('AutomataINT.png')
else:
    Errores.append('Error: se espera un tipo de token: ; o = |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
    Errores.append('AutomataINT.png')
# int a; 3
# int a = 3; 5

#===== Asignacion =====
def gAsign(t,i):
    if(len(t)>i+1):
        if((t[i+1].type != 'ID') and (t[i+1].type != 'NUMERO')):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'
'+str(t[i+1].type)+', se esperaba otro tipo de token: ID o NUMERO |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataINT.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID o NUMERO |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataINT.png')
            return

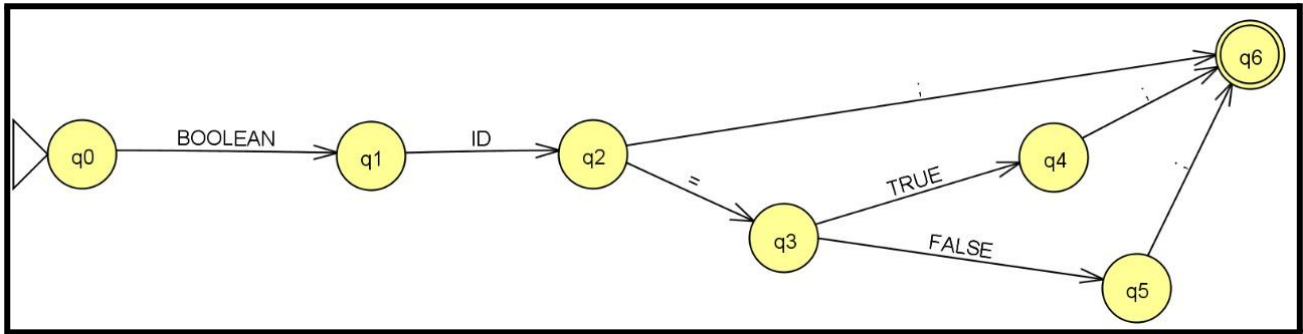
    if(len(t)>i+2):
        if(t[i+2].type != 'PUNTOCOMA'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ; | Linea:'+str(t[i+2].lineno) +'
Columna:'+str(t[i+2].lexpos))
            Errores.append('AutomataINT.png')
        else:
            Errores.append('Error: se espera un tipo de token: ; |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataINT.png')
```

**Autómata INT:**



```
#===== Declarar BOOLEAN =====
def gBOOLEAN(t,i):
    #Cantidad de elementos Len(t)
    if(len(t)>i+1):
        if(t[i+1].type != 'ID'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ID | Línea:'+str(t[i+1].lineno) + '
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataBOOLEAN.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID |
Línea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataBOOLEAN.png')
            return
    #Se cumplio la primera condicion ->
    if(len(t)>i+2):
        if(t[i+2].type == 'ASIGNACION'):
            gAsignBOOLEAN(t,i+2)
        elif(t[i+2].type != 'PUNTOCOMA'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ; o = | Línea:'+str(t[i+2].lineno) + '
Columna:'+str(t[i+2].lexpos))
            Errores.append('AutomataBOOLEAN.png')
        else:
            Errores.append('Error: se espera un tipo de token: ; o = |
Línea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataBOOLEAN.png')
```

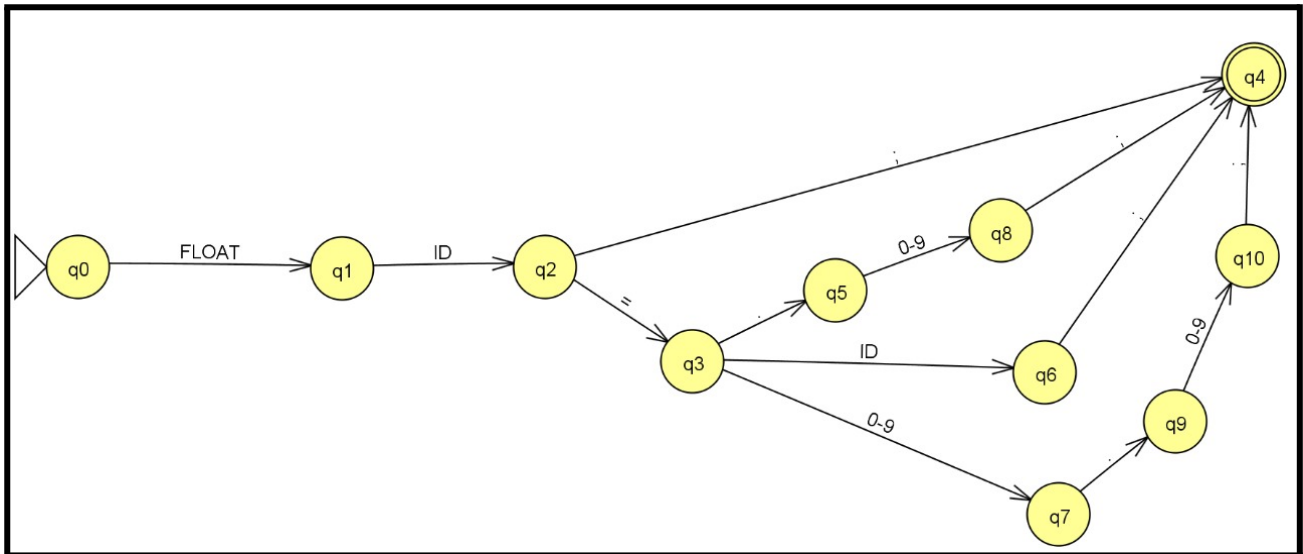
### Autómata BOOLEAN:



```
#===== Declarar FLOAT =====
def gFLOAT(t,i):
    #Cantidad de elementos len(t)
    if(len(t)>i+1):
        if(t[i+1].type != 'ID'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ID | Linea:'+str(t[i+1].lineno) + '
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataFLOAT.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID |
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataFLOAT.png')
            return
    #Se cumplio la primera condicion ->
    if(len(t)>i+2):
        if(t[i+2].type == 'ASIGNACION'):
            gAssignFLOAT(t,i+2)
        elif(t[i+2].type != 'PUNTOCOMA'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ; o = | Linea:'+str(t[i+2].lineno) + '
Columna:'+str(t[i+2].lexpos))
            Errores.append('AutomataFLOAT.png')
        else:
            Errores.append('Error: se espera un tipo de token: ; o = |
Linea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataFLOAT.png')
```



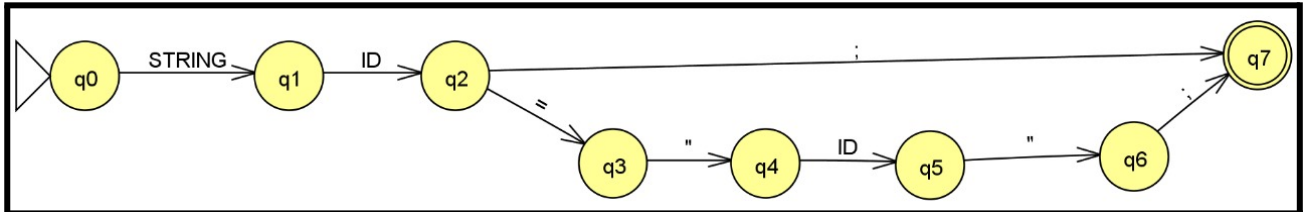
### Autómata FLOAT:



```
#===== Declarar STRING =====
def gSTRING(t,i):
    #Cantidad de elementos len(t)
    if(len(t)>i+1):
        if(t[i+1].type != 'ID'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ID | Linea:'+str(t[i+1].lineno) + '
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataSTRING.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID |
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataSTRING.png')
            return
    #Se cumplio la primera condicion ->
    if(len(t)>i+2):
        if(t[i+2].type == 'ASIGNACION'):
            gAssignSTRING(t,i+2)
        elif(t[i+2].type != 'PUNTOCOMA'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ; o = | Linea:'+str(t[i+2].lineno) + '
Columna:'+str(t[i+2].lexpos))
            Errores.append('AutomataSTRING.png')
        else:
```

```
Errores.append('Error: se espera un tipo de token: ; o = |  
Línea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))  
Errores.append('AutomataSTRING.png')  
#STRING A = "AAGRIA";
```

### Autómata STRING:



Asignación de los distintos tipos de datos: En caso asignar un valor a la variable declarada, se utiliza el camino alterno '=' el cual implica el uso de estas funciones.

```
#===== AsignacionBOOLEAN  
=====  
def gAssignBOOLEAN(t,i):  
    if(len(t)>i+1):  
        if((t[i+1].type != 'TRUE') and (t[i+1].type != 'FALSE')):  
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'  
'+str(t[i+1].type)+'', se esperaba otro tipo de token: TRUE o FALSE |  
Línea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))  
            Errores.append('AutomataBOOLEAN.jpg')  
            return  
        else:  
            Errores.append('Error: se espera un tipo de token: TRUE o FALSE |  
Línea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))  
            Errores.append('AutomataBOOLEAN.png')  
            return  
    if(len(t)>i+2):  
        if(t[i+2].type != 'PUNTOCOMA'):  
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'', se esperaba  
otro tipo de token: ; | Línea:'+str(t[i+2].lineno) + '  
Columna:'+str(t[i+2].lexpos))  
            Errores.append('AutomataBOOLEAN.png')  
        else:  
            Errores.append('Error: se espera un tipo de token: ; |  
Línea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))  
            Errores.append('AutomataBOOLEAN.png')
```

```
#===== AsignacionSTRING
=====
def gAssignSTRING(t,i):
    if(len(t)>i+1):
        if((t[i+1].type != 'COMILLADOBLE')):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'
'+str(t[i+1].type)+'', se esperaba otro tipo de token: COMILLADOBLE |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataSTRING.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: COMILLADOBLE |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataSTRING.png')
            return

    if(len(t)>i+2):
        if(t[i+2].type != 'ID'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'', se esperaba
otro tipo de token: ID | Linea:'+str(t[i+2].lineno) +'
Columna:'+str(t[i+2].lexpos))
            Errores.append('AutomataSTRING.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataSTRING.png')
            return

    if(len(t)>i+3):
        if(t[i+3].type != 'COMILLADOBLE'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'', se esperaba
otro tipo de token: COMILLADOBLE | Linea:'+str(t[i+3].lineno) +'
Columna:'+str(t[i+3].lexpos))
            Errores.append('AutomataSTRING.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: COMILLADOBLE |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataSTRING.png')
```

```
        return

    if(len(t)>i+4):
        if(t[i+4].type != 'PUNTOCOMA'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
otro tipo de token: ; | Linea:'+str(t[i+4].lineno) + '
Columna:'+str(t[i+4].lexpos))
            Errores.append('AutomataSTRING.png')
        else:
            Errores.append('Error: se espera un tipo de token: ; |
Linea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataSTRING.png')

#===== AsignacionFLOAT =====
def gAssignFLOAT(t,i):
    if(len(t)>i+1):
        if((t[i+1].type != 'ID') and (t[i+1].type != 'NUMERO') and (t[i+1].type
!= 'PUNTO')):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'
'+str(t[i+1].type)+', se esperaba otro tipo de token: ID o NUMERO o PUNTO |
Linea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataFLOAT.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID o NUMERO o PUNTO |
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataFLOAT.png')
            return

    #El caso si el primer token es un PUNTO
    if(t[i+1].type == 'PUNTO'):
        if(len(t)>i+2):
            if(t[i+2].type != 'NUMERO'):
                Errores.append('Token Inesperado:'+str(t[i+1].value)+', se
esperaba otro tipo de token: NUMERO | Linea:'+str(t[i+2].lineno) + '
Columna:'+str(t[i+2].lexpos))
                Errores.append('AutomataFLOAT.png')
                return
            else:
                return
```

```
        Errores.append('Error: se espera un tipo de token: NUMERO |  
Linea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))  
        Errores.append('AutomataFLOAT.png')  
        return  
  
    if(len(t)>i+3):  
        if(t[i+3].type != 'PUNTOCOMA'):  
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se  
esperaba otro tipo de token: ; o NUMERO | Linea:'+str(t[i+3].lineno) + '  
Columna:'+str(t[i+3].lexpos))  
            Errores.append('AutomataFLOAT.png')  
        else:  
            Errores.append('Error: se espera un tipo de token: ; o NUMERO |  
Linea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))  
            Errores.append('AutomataFLOAT.png')  
            #El caso si el primer token es un ID  
            if(t[i+1].type == 'ID'):  
                if(len(t)>i+2):  
                    if(t[i+2].type != 'PUNTOCOMA'):  
                        Errores.append('Token Inesperado:'+str(t[i+1].value)+', se  
esperaba otro tipo de token: ; | Linea:'+str(t[i+2].lineno) + '  
Columna:'+str(t[i+2].lexpos))  
                        Errores.append('AutomataFLOAT.png')  
                    else:  
                        Errores.append('Error: se espera un tipo de token: ; |  
Linea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))  
                        Errores.append('AutomataFLOAT.png')  
                    #El caso si el primer token es un NUMERO  
                    if(t[i+1].type == 'NUMERO'):  
                        if(len(t)>i+2):  
                            if(t[i+2].type != 'PUNTO'):  
                                Errores.append('Token Inesperado:'+str(t[i+1].value)+', se  
esperaba otro tipo de token: PUNTO | Linea:'+str(t[i+2].lineno) + '  
Columna:'+str(t[i+2].lexpos))  
                                Errores.append('AutomataFLOAT.png')  
                                return  
                            else:  
                                Errores.append('Error: se espera un tipo de token: PUNTO |  
Linea:'+str(t[i+1].lineno) + ' Columna:'+str(t[i+1].lexpos))  
                                Errores.append('AutomataFLOAT.png')  
                                return
```

```
        if(len(t)>i+3):
            if(t[i+3].type != 'NUMERO'):
                Errores.append('Token Inesperado:'+str(t[i+1].value)+', se
esperaba otro tipo de token: NUMERO | Linea:'+str(t[i+3].lineno) +'
Columna:'+str(t[i+3].lexpos))
                Errores.append('AutomataFLOAT.png')
                return
            else:
                Errores.append('Error: se espera un tipo de token: NUMERO |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
                Errores.append('AutomataFLOAT.png')
                return

        if(len(t)>i+4):
            if(t[i+4].type != 'PUNTOCOMA'):
                Errores.append('Token Inesperado:'+str(t[i+1].value)+', se
esperaba otro tipo de token: ; o NUMERO | Linea:'+str(t[i+4].lineno) +'
Columna:'+str(t[i+4].lexpos))
                Errores.append('AutomataFLOAT.png')
            else:
                Errores.append('Error: se espera un tipo de token: ; o NUMERO |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
                Errores.append('AutomataFLOAT.png')
```

```
#=====condicion if=====
def gIF(t,i):
    if(len(t)>i+1):
        if(t[i+1].type != 'PARENTESISABIERTO'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
un tipo de token: PARENTESISABIERTO | Linea:'+str(t[i+1].lineno) +'
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataIF.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: PARENTESISABIERTO "("
| Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataIF.png')
            return
```

```
i = Condicion(t,i+1)
if(i == 0):
    return

if(len(t)>i):
    if(t[i].type != 'PARENTESISCERRADO'):
        Errores.append('Token Inesperado:'+str(t[i].value)+', se esperaba un
tipo de token: PARENTESISCERRADO | Linea:'+str(t[i].lineno) +'
Columna:'+str(t[i+1].lexpos))
        Errores.append('AutomataIF.png')
        return
    else:
        Errores.append('Error: se espera un tipo de token: ) PARENTESISCERRADO |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
        Errores.append('AutomataIF.png')
        if(len(t)>i+1):
            if(t[i+1].type != 'LLAVEABIERTO'):
                Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
un tipo de token: LLAVEABIERTO | Linea:'+str(t[i+1].lineno) +'
Columna:'+str(t[i+1].lexpos))
                Errores.append('AutomataIF.png')
                return
            else:
                Errores.append('Error: se espera un tipo de token: LLAVEABIERTO |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
                Errores.append('AutomataIF.png')

global llavesFaltantes
llavesFaltantes+=1

#funcion reutilizable para cualquier tipo de condicion la cual es recursiva y
retorna un 0 si la sintaxis de la condicion esta mal
# y si esta bien la sintaxis retorn el valos de la pisicion en donde termino la
condicion del arreglo de tokens
def Condicion(t,i):
    i2=0
    if(len(t)>i+1):
        if((t[i+1].type != 'ID') and (t[i+1].type != 'NUMERO')):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'
'+str(t[i+1].type)+', se esperaba un tipo de token: ID o NUMERO |
```

```
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
    Errores.append('AutomataCONDICION.png')
    return 0
else:
    Errores.append('Error: se espera un tipo de token: ID o NUMERO |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
    Errores.append('AutomataCONDICION.png')
    return 0

if(len(t)>i+2):
    if((t[i+2].type != 'IGUAL') and (t[i+2].type != 'DIFERENTE') and
(t[i+2].type != 'MAYORQUE') and (t[i+2].type != 'MENORQUE') and (t[i+2].type !=
'MAYORIGUALQUE') and (t[i+2].type != 'MENORIGUALQUE') and (t[i+2].type !=
'IGUAL'))):
        Errores.append('Token Inesperado:'+str(t[i+2].value)+', se esperaba
un operador de comparacion: | Linea:'+str(t[i+2].lineno) +'
Columna:'+str(t[i+2].lexpos))
        Errores.append('AutomataCONDICION.png')
        return 0
    else:
        Errores.append('Error: se espera un tipo de token: de COMPARACION
(<,>,<=,>=,!=,==,!=,!=) | Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
        Errores.append('AutomataCONDICION.png')
        return 0

if(len(t)>i+3):
    if((t[i+3].type != 'ID') and (t[i+3].type != 'NUMERO') and (t[i+3].type
!= 'FALSE') and (t[i+3].type != 'TRUE')):
        Errores.append('Token Inesperado:'+str(t[i+3].value)+'
'+str(t[i+3].type)+', se esperaba un tipo de token: ID, NUMERO o VALOR BOOLEANO |
Linea:'+str(t[i+3].lineno) +' Columna:'+str(t[i+3].lexpos))
        Errores.append('AutomataCONDICION.png')
        return 0
    else:
        Errores.append('Error: se espera un tipo de token: ID, NUMERO o VALOR
BOOLEANO | Linea:'+str(t[i+2].lineno) +' Columna:'+str(t[i+2].lexpos))
        Errores.append('AutomataCONDICION.png')
        return 0

if(len(t)>i+4):
    if(t[i+4].type == 'AND' or t[i+4].type == 'OR' or t[i+4].type == 'NOT'):
```



```

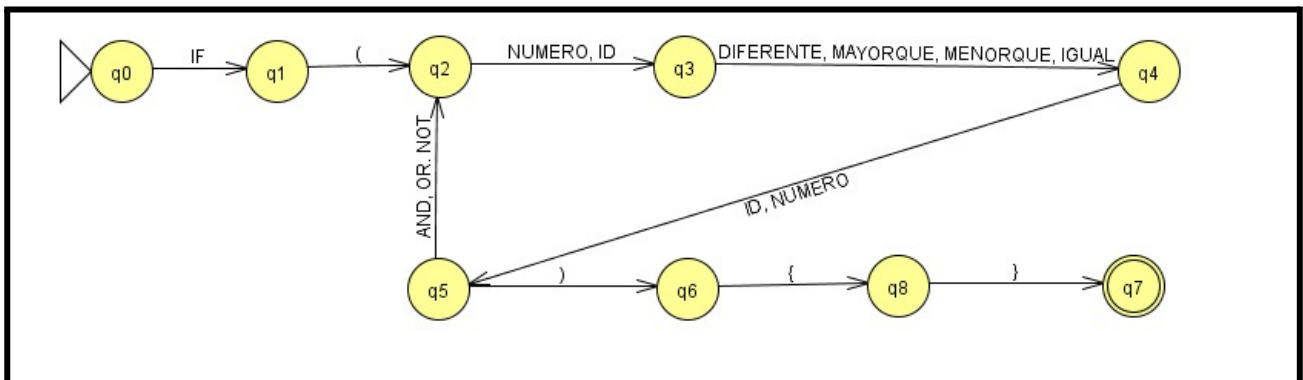
        i2 = Condicion(t,i+4)
        return i2
    ...

    if(t[i+4].type != 'PARENTESISCERRADO'):
        Errores.append('Token Inesperado:'+str(t[i+4].value)+', se esperaba
un tipo de token: PARENTESISCERRADO | Linea:'+str(t[i+4].lineno) +'
Columna:'+str(t[i+4].lexpos))
        return 0'''

    return i+4
else:
    Errores.append('Error: se espera un tipo de token: PARENTESISCERRADO ")"
u OPERADOR LOGICO (AND, OR, NOT)| Linea:'+str(t[i+3].lineno) +'
Columna:'+str(t[i+3].lexpos))
    Errores.append('AutomataCONDICION.png')
    return 0

```

### Autómata IF:



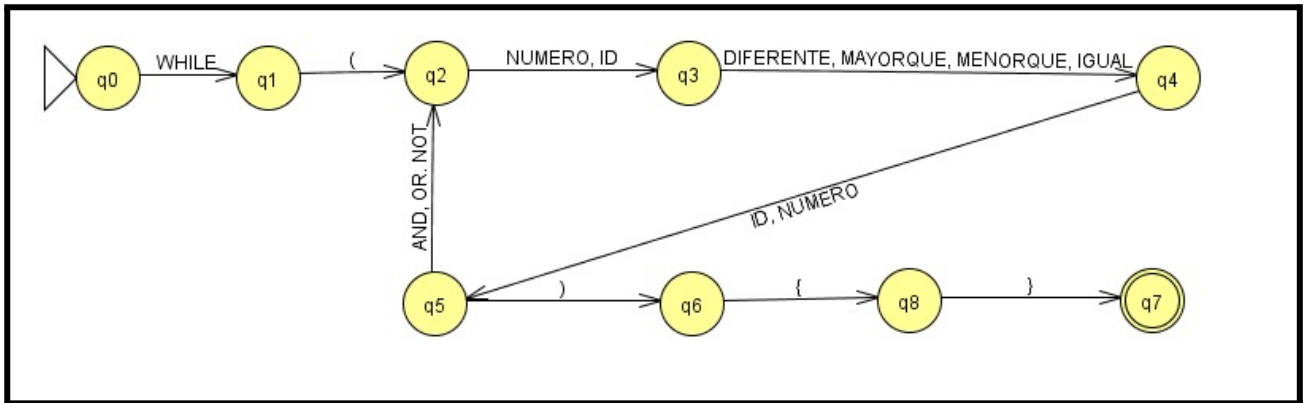
```

#=====CICLO DE REPETICION
WHILE=====
def gWHILE(t,i):
    if(len(t)>i+1):
        if(t[i+1].type != 'PARENTESISABIERTO'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
un tipo de token: PARENTESISABIERTO | Linea:'+str(t[i+1].lineno) +'
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataWHILE.png')
            return
        else:

```

```
    Errores.append('Error: se espera un tipo de token: PARENTESISABIERTO "("  
| Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))  
    Errores.append('AutomataWHILE.png')  
    return  
  
    i = Condicion(t,i+1)  
    if(i == 0):  
        return  
  
    if(len(t)>i):  
        if(t[i].type != 'PARENTESISCERRADO'):  
            Errores.append('Token Inesperado:'+str(t[i].value)+', se esperaba un  
tipo de token: PARENTESISCERRADO | Linea:'+str(t[i].lineno) +'  
Columna:'+str(t[i+1].lexpos))  
            Errores.append('AutomataWHILE.png')  
            return  
        else:  
            Errores.append('Error: se espera un tipo de token: ) PARENTESISCERRADO |  
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))  
            Errores.append('AutomataWHILE.png')  
            if(len(t)>i+1):  
                if(t[i+1].type != 'LLAVEABIERTO'):  
                    Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba  
un tipo de token: LLAVEABIERTO | Linea:'+str(t[i+1].lineno) +'  
Columna:'+str(t[i+1].lexpos))  
                    Errores.append('AutomataWHILE.png')  
                    return  
                else:  
                    Errores.append('Error: se espera un tipo de token: LLAVEABIERTO |  
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))  
                    Errores.append('AutomataWHILE.png')  
            global llavesFaltantes  
            llavesFaltantes+=1
```

***Autómata WHILE:***



```
#===== DEF (declaracion funciones) =====
def gDEF(t,i):
    if(len(t)>i+1):
        if(t[i+1].type != 'ID'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
un tipo de token: ID | Linea:'+str(t[i+1].lineno) + '
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataDEF.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID |
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataDEF.png')
            return

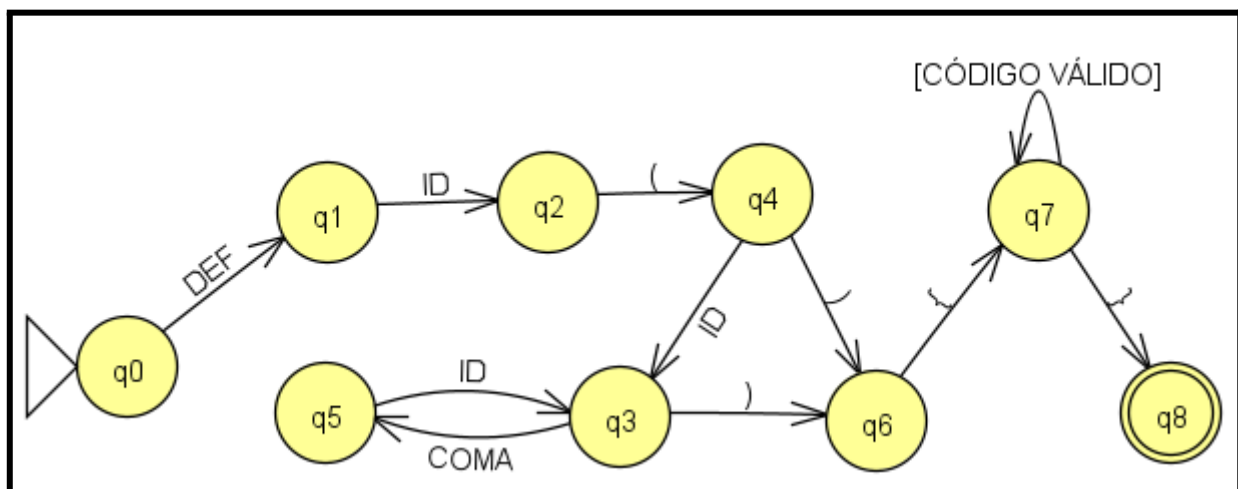
    if(len(t)>i+2):
        if(t[i+2].type != 'PARENTESISABIERTO'):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
un tipo de token: PARENTESISABIERTO | Linea:'+str(t[i+1].lineno) + '
Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataDEF.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: PARENTESISABIERTO "("
| Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataDEF.png')
            return

    i = Parametro(t,i+2)
    if(i == 0):
```

```
return

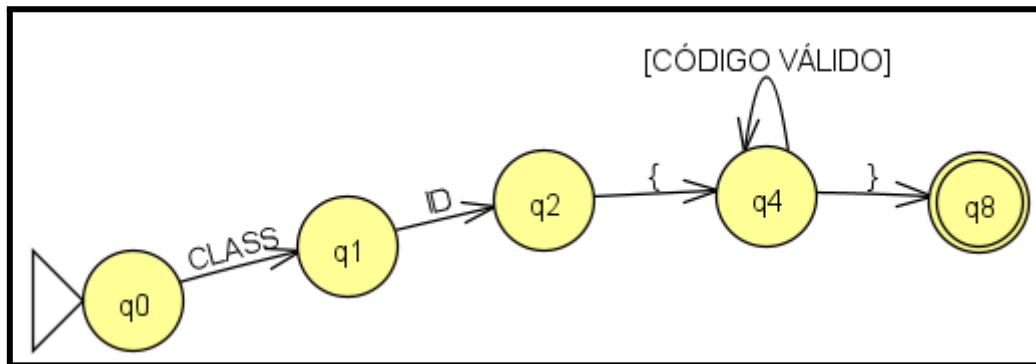
if(len(t)>i):
    if(t[i].type != 'PARENTESISCERRADO'):
        Errores.append('Token Inesperado:'+str(t[i].value)+', se esperaba un
tipo de token: PARENTESISCERRADO | Línea:'+str(t[i].lineno) +'
Columna:'+str(t[i+1].lexpos))
        Errores.append('AutomataDEF.png')
        return
    else:
        Errores.append('Error: se espera un tipo de token: ) PARENTESISCERRADO |
Línea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
        Errores.append('AutomataDEF.png')
        if(len(t)>i+1):
            if(t[i+1].type != 'LLAVEABIERTO'):
                Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba
un tipo de token: LLAVEABIERTO | Línea:'+str(t[i+1].lineno) +'
Columna:'+str(t[i+1].lexpos))
                Errores.append('AutomataDEF.png')
                return
            else:
                Errores.append('Error: se espera un tipo de token: LLAVEABIERTO |
Línea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
                Errores.append('AutomataDEF.png')
                global llavesFaltantes
                llavesFaltantes+=1
```

### Autómata DEF:



```
#===== CLASS =====  
def gCLASS(t,i):  
    if(len(t)>i+1):  
        if(t[i+1].type != 'ID'):  
            Errores.append('Token Inesperado:'+str(t[i+1].value)+', se esperaba  
un tipo de token: ID | Linea:'+str(t[i+1].lineno) +'  
Columna:'+str(t[i+1].lexpos))  
            Errores.append('AutomataCLASS.png')  
            return  
        else:  
            Errores.append('Error: se espera un tipo de token: ID |  
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))  
            Errores.append('AutomataCLASS.png')  
            return  
  
    if(len(t)>i+2):  
        if(t[i+2].type != 'LLAVEABIERTO'):  
            Errores.append('Token Inesperado:'+str(t[i+2].value)+', se esperaba  
un tipo de token: LLAVEABIERTO | Linea:'+str(t[i+2].lineno) +'  
Columna:'+str(t[i+2].lexpos))  
            Errores.append('AutomataCLASS.png')  
            return  
        else:  
            Errores.append('Error: se espera un tipo de token: LLAVEABIERTO |  
Linea:'+str(t[i].lineno) + ' Columna:'+str(t[i].lexpos))  
            Errores.append('AutomataCLASS.png')  
    global llavesFaltantes  
    llavesFaltantes+=1
```

***Autómata CLASS:***

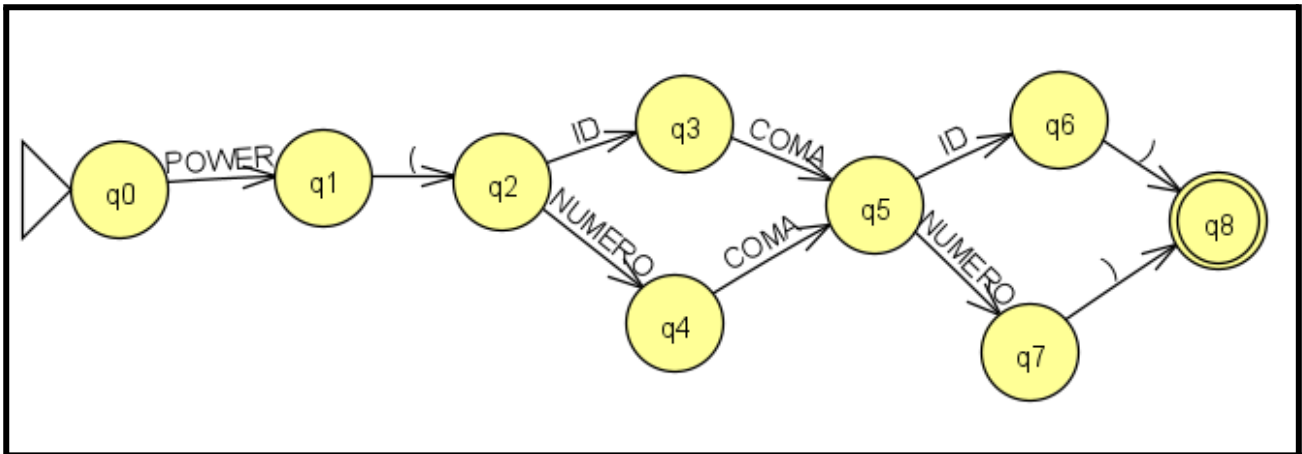


```
#===== POWER (POTENCIA X^Y) =====
def gPOWER(t,i):
    if(len(t)>i+1):
        if((t[i+1].type != 'PARENTESISABIERTO')):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'
'+str(t[i+1].type)+', se esperaba otro tipo de token: PARENTESISABIERTO "(" |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataPOWER.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: PARENTESISABIERTO "("
| Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataPOWER.png')
            return
    if(len(t)>i+2):
        if((t[i+2].type != 'ID') and (t[i+2].type != 'NUMERO')):
            Errores.append('Token Inesperado:'+str(t[i+2].value)+'
'+str(t[i+2].type)+', se esperaba otro tipo de token: ID o NUMERO |
Linea:'+str(t[i+2].lineno) +' Columna:'+str(t[i+2].lexpos))
            Errores.append('AutomataPOWER.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID o NUMERO |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataPOWER.png')
            return
    if(len(t)>i+3):
        if((t[i+3].type != 'COMA')):
            Errores.append('Token Inesperado:'+str(t[i+3].value)+'
```

```
'+str(t[i+3].type)+', se esperaba otro tipo de token: COMA |
Linea:'+str(t[i+3].lineno) +' Columna:'+str(t[i+3].lexpos))
    Errores.append('AutomataPOWER.png')
    return
else:
    Errores.append('Error: se espera un tipo de token: COMA |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
    Errores.append('AutomataPOWER.png')
    return
if(len(t)>i+4):
    if((t[i+4].type != 'ID') and (t[i+4].type != 'NUMERO')):
        Errores.append('Token Inesperado:'+str(t[i+4].value)+'
'+str(t[i+4].type)+', se esperaba otro tipo de token: ID o NUMERO |
Linea:'+str(t[i+4].lineno) +' Columna:'+str(t[i+4].lexpos))
        Errores.append('AutomataPOWER.png')
        return
    else:
        Errores.append('Error: se espera un tipo de token: ID o NUMERO |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
        Errores.append('AutomataPOWER.png')
        return

if(len(t)>i+5):
    if((t[i+5].type != 'PARENTESISCERRADO')):
        Errores.append('Token Inesperado:'+str(t[i+5].value)+'
'+str(t[i+5].type)+', se esperaba otro tipo de token: PARENTESISCERRADO ")" |
Linea:'+str(t[i+5].lineno) +' Columna:'+str(t[i+5].lexpos))
        Errores.append('AutomataPOWER.png')
        return
    else:
        Errores.append('Error: se espera un tipo de token: PARENTESISCERRADO ")"
| Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
        Errores.append('AutomataPOWER.png')
```

### ***Autómata POWER:***

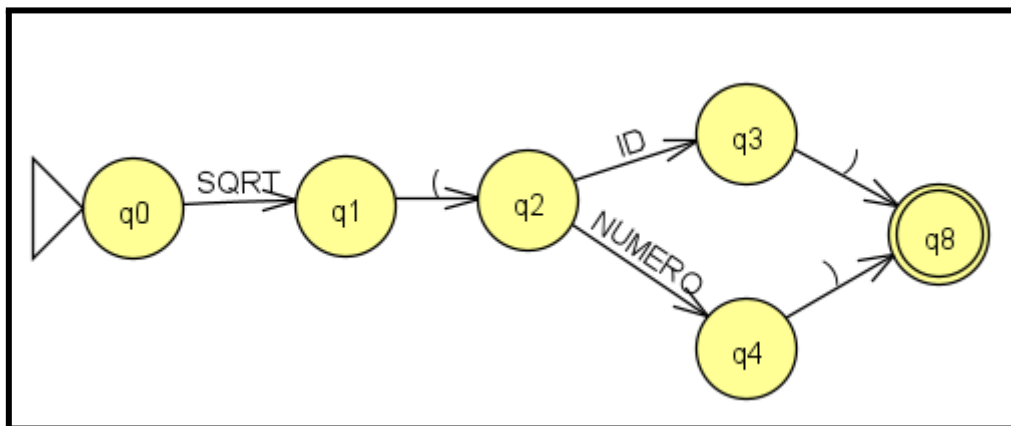


```
#===== raiz cuadrada =====
def gSQRT(t,i):
    if(len(t)>i+1):
        if((t[i+1].type != 'PARENTESISABIERTO')):
            Errores.append('Token Inesperado:'+str(t[i+1].value)+'
'+str(t[i+1].type)+', se esperaba otro tipo de token: PARENTESISABIERTO "(" |
Linea:'+str(t[i+1].lineno) +' Columna:'+str(t[i+1].lexpos))
            Errores.append('AutomataSQRT.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: PARENTESISABIERTO "("
| Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataSQRT.png')
            return
    if(len(t)>i+2):
        if((t[i+2].type != 'ID') and (t[i+2].type != 'NUMERO')):
            Errores.append('Token Inesperado:'+str(t[i+2].value)+'
'+str(t[i+2].type)+', se esperaba otro tipo de token: ID o NUMERO |
Linea:'+str(t[i+2].lineno) +' Columna:'+str(t[i+2].lexpos))
            Errores.append('AutomataSQRT.png')
            return
        else:
            Errores.append('Error: se espera un tipo de token: ID o NUMERO |
Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))
            Errores.append('AutomataSQRT.png')
            return
    if(len(t)>i+3):
```



```
if((t[i+3].type != 'PARENTESISCERRADO')):  
    Errores.append('Token Inesperado:'+str(t[i+3].value)+'  
' +str(t[i+3].type)+', se esperaba otro tipo de token: PARENTESISCERRADO ")" |  
Linea:'+str(t[i+3].lineno) +' Columna:'+str(t[i+3].lexpos))  
    Errores.append('AutomataSQRT.png')  
    return  
else:  
    Errores.append('Error: se espera un tipo de token: PARENTESISCERRADO ")")  
| Linea:'+str(t[i].lineno) +' Columna:'+str(t[i].lexpos))  
    Errores.append('AutomataSQRT.png')
```

### *Autómata SQRT:*



## Herramientas:

El presente analizador léxico y sintáctico se realizó en el lenguaje de programación de alto nivel Python, esto debido a su alta compatibilidad con las herramientas necesarias para cumplir con los objetivos del presente proyecto, que se logra gracias a su extenso catálogo de módulos integrados y librerías adicionales; además, facilita el proceso que realiza el equipo de desarrollo, ya que al ser necesario que todos trabajen con el mismo lenguaje de programación, su suave curva de aprendizaje permite que todos trabajen en el mismo nivel rápidamente.

De la misma manera, permite que la documentación presentada en el presente reporte sea fácil de comprender hacia el lector, gracias a la limpieza y legibilidad del código generado, como ejemplo, se presenta el código para imprimir “Hola mundo” en los lenguajes considerados al principio en el siguiente collage:



```
class HolaMundo {  
    public static void main( String args[] ) {  
        System.out.println( "Hola Mundo!" );  
    }  
}
```



```
#include <iostream.h>  
#include <stdlib.h>  
  
int main ()  
{  
    cout<<"Hola Mundo\n";  
    system("PAUSE");  
    return 0;  
}
```



```
print("¡Hola, Mundo!")
```

Cabe mencionar que Python está desarrollado bajo una licencia que permite usar y distribuirlo bajo cualquier circunstancia, incluso para poner a disposición código y programas que se generan con razones comerciales.

También existen ventajas para el usuario final del compilador, entre estas la alta compatibilidad de los programas generados en cualquier sistema operativo, y aunque Python no sea conocido por su optimización de programas grandes, el compilador consume muy poca memoria y cumple su función de manera rápida y eficiente.

El equipo de desarrollo trabajó en el entorno de trabajo proporcionado por el IDE de Visual Studio Code, que facilitó en gran medida el desarrollo en conjunto del programa gracias a su integración con Git, que fue la plataforma especificada para manejar, desarrollar y publicar el proyecto.

Este es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Todas estas características nos fueron útiles a lo largo del desarrollo del proyecto.

Para hacer y mostrar los componentes gráficos del editor, se utilizó TkInter, que es la interfaz por defecto de Python para el kit de herramientas de GUI Tk.

Para la funcionalidad que permite al usuario ingresar código en el editor por medio de comandos de voz, se utilizó la librería de SpeechRecognition, que permite capturar la voz del usuario y realizar las acciones deseadas de acuerdo con la información captada.

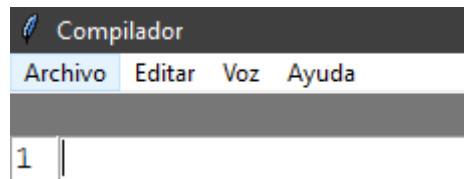
También nos apoyamos del programa Lex, integrado a Python por David Beazley en la librería de PLY, para la generación de nuestro analizador léxico. Este nos permite identificar y clasificar los tokens utilizados para este proceso, especificando las tuplas de token y tipo de token para analizar el texto proporcionado y separarlo en una lista de tokens.

PLY is a zero-dependency Python implementation of the traditional parsing tools lex and yacc. It uses the same LALR(1) parsing algorithm as yacc and has most of its core features. It is compatible with all modern versions of Python.

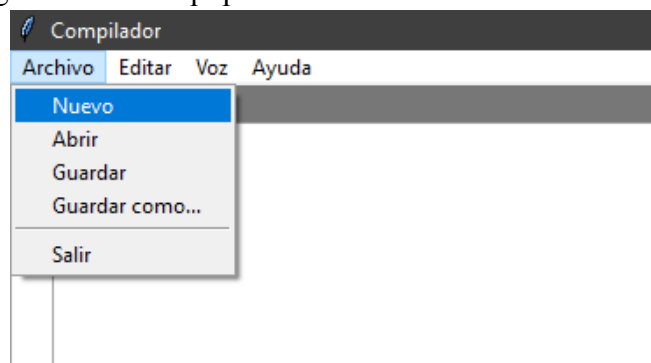
PLY was originally created in 2001 to support an Introduction to Compilers course at the University of Chicago. As such, it has almost no features other than the core LALR(1) parsing algorithm. However, from a more practical point of view, there is a lot of flexibility in terms of how you decide to use it. You can use PLY to build Abstract Syntax Trees (ASTs), simple one-pass compilers, protocol decoders, or even a more advanced parsing framework.

## Manual de Usuario

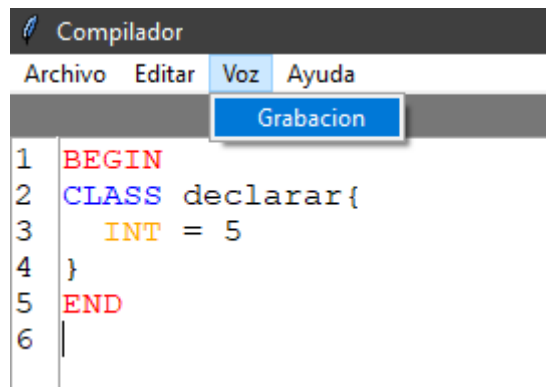
Menú:



Dentro de archivo tenemos la opción de nuevo, abrir, guardar, guardar como y salir. La opción de nuevo nos deja el editor en blanco y nos pregunta si queremos guardar el contenido que modificamos. La opción de Abrir nos permite abrir un archivo de texto almacenado en el equipo, la opción de guardar nos deja guardar un archivo sobre el que estamos trabajando y el guardar como nos deja guardar en el equipo un archivo de texto con el nombre que especifiquemos.

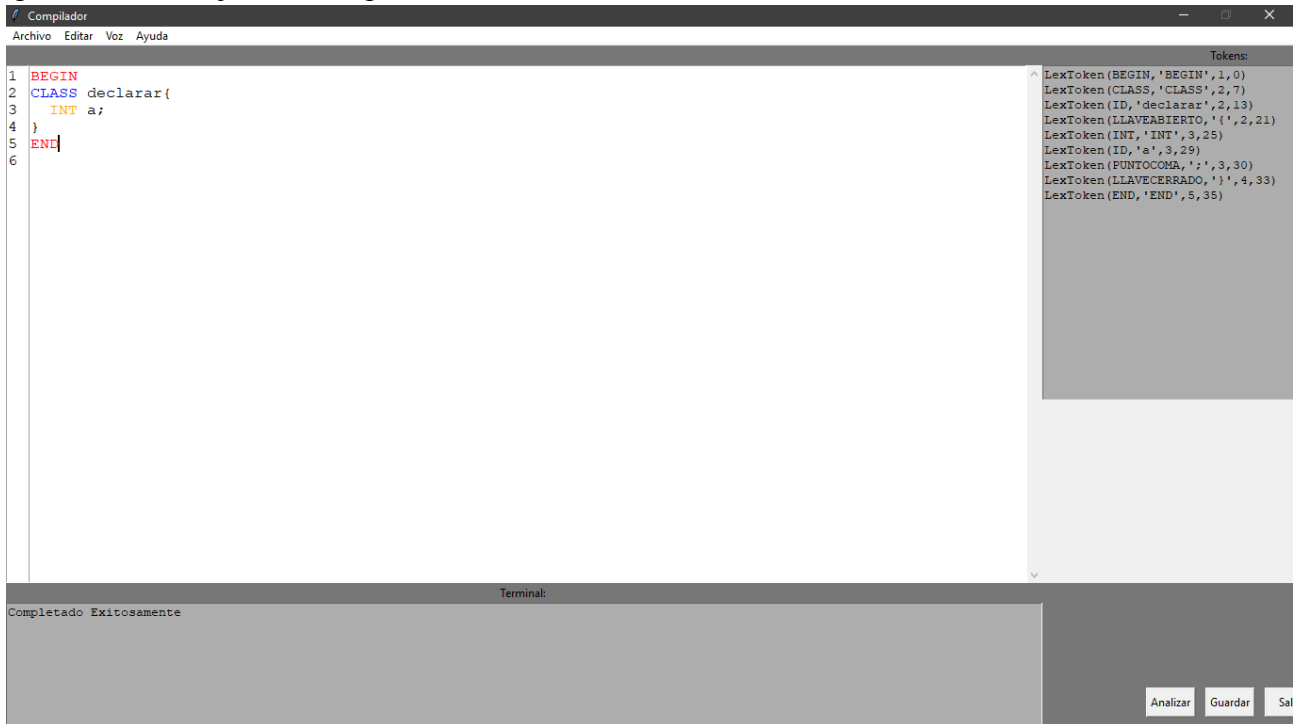


Dentro del menú Voz tenemos grabación el cual comenzará a grabar nuestra voz, la grabación se detiene después de escuchar un silencio y todo lo que dijimos antes se va a plasmar en el editor de texto.



**Botón Analizar:** Después de presionar el botón analizar, podremos observar en la esquina superior izquierda la lista de los tokens que se encontraron.

Después se van a listar los errores sintácticos encontrados en el texto y se van a apilar uno sobre otro en la parte inferior. En el caso de la siguiente imagen como no hubo errores sintácticos nos aparece el mensaje de “Completado Exitosamente”



En el caso que tengamos errores vemos en la siguiente imagen como se muestran dentro de la terminal. Lo que nos permite es poder dar click izquierdo y después clic derecho para mostrar la imagen. Esto se hace de esta manera ya que el clic izquierdo lo que realiza es setear la imagen que se muestra y el clic derecho lo que hace es mostrarla dentro del editor.



### Ejemplos de funciones en el lenguaje:

A continuación se presenta una explicación de cómo se aplica el lenguaje en algunas de las funciones propias de la silla de ruedas:

#### Mover silla:

```
begin
  class moverHacia{
    def mover(grados,distancia){
      int destinoX;
      int destinoY;
      destinoX = posX + cos(x) * distancia;
      destinoY = posY + sin(x) * distancia;
      girar(grados);
      while(posX!=destinoX AND posY!=destinoY){
        if(detectarObstaculo==FALSE){
          avanzar();
        }
        if(detectarObstaculo==TRUE){
          girar(10);
        }
      }
    }
  }
end
```

Dentro de los delimitadores de BEGIN y END, tenemos la clase moverHacia, que contiene la función para mover la silla tomando como parámetros los grados relativos a la orientación actual de la silla, y la distancia en línea recta a recorrer hasta llegar al destino.

Se definen los valores de posición del destino en forma de coordenadas y se obtienen utilizando las funciones trigonométricas necesarias con los parámetros que se tienen.

Después, la silla gira los grados especificados y comienza a avanzar. Se inicia un ciclo que especifica que en caso de que encuentre un obstáculo, la silla gira 10 grados. Cuando la silla llega a su destino, el ciclo se termina y la silla se detiene.

#### Encender luces:

```
begin
  int intensidad = 5;
  int encendido = 1;          #0->apagado, 1->encendido
  string patron = "estandar";
  int tiempo = 100;
  def luces(intensidad,encendido,patron,tiempo){
  begin
    boolean state = true;
    int iluminacion;
    int c = 1;
    if(encendido == 0){
      iluminacion = 0;
    }

    if(patron == estandar and encendido == 1){
      iluminacion = intensidad;
    }
  }
}
```



```

}
if(patron == fade and encendido == 1){
    while(state == true){
        while(intensidad >= 0){
            intensidad--;
            c++;
        }
        while(intensidad <= 10){
            intensidad++;
            c++;
        }
        if(c == tiempo){
            state = false;
        }
    }#while
}
if(patron == alternado and encendido == 1){
    int z = 0;
    c = 0;
    while(state == true){
        if(c == 0){
            intensidad = 10;
        }
        while(c < 10){
            c++;
            z++;
        }
        if(c == 10){
            intensidad = 0;
        }
        while(c > 0){
            c--;
            z++;
        }
        if(z == tiempo){
            state = false;
        }
    }
}
}
end

```

Se toman los valores iniciales, estos como predeterminados son de encendido, intensidad nivel 5, patrón estándar y 100 segundos de duración. Posteriormente se definen los patrones de luces. Se presentan 3 modos: estándar, fade y alternado. En el primero, la intensidad de las luces se mantiene constante, en el segundo cambia lentamente, y en el tercero alterna entre el nivel más alto y el más bajo en periodos de 10 segundos. El programa permite tomar parámetros que permiten alternar entre estos patrones, además de definir la duración e intensidad de estos.

**Masajeador:**

```

int nivel = 8;           #intensidad (8/10)
int encendido = 1;      #0->apagado, 1->encendido
string patron = "estandar"; #estandar por default
int tiempo = 200;
string zona = "lumbar";

def masajeador(nivel,encendido,patron,tiempo,zona){
begin
    boolean state = true;
    int potencia;
    int a;
    int c = 1;

    if(encendido == 0){
        potencia = 0;
    }

    if(zona == lumbar){
        a = 1;
    }

    if(zona == dorsal){
        a = 2;
    }

    if(zona == alta){
        a = 3;
    }

    if(patron == estandar and encendido == 1){
        potencia = nivel;
    }
    if(patron == fade and encendido == 1){
        while(state == true){
            while(nivel >= 0){
                nivel--;
                c++;
            }
            while(nivel <= 10){
                nivel++;
                c++;
            }
            #temporizador
            if(c == tiempo){
                state = false;
            }
        }#while
    }#if principal
}

```





```

if(patron == alternado and encendido == 1){
    int z = 0;
    c = 0;
    while(state == true){
        if(c == 0){
            nivel = 10;
        }
        #Crear un temporizador para alternar
        while(c < 5){
            c++;
            z++;
        }
        if(c == 5){
            nivel = 0;
        }
        while(c > 0){
            c--;
            z++;
        }
        if(z == tiempo){
            state = false;
        }
    }#while
}
#endif
end
}#def masajeador

```

El programa inicia definiendo los valores iniciales, donde se definen el nivel de intensidad, el patrón por defecto a seguir, el tiempo de duración y la zona corporal a la que se aplica el masaje. Después se define la variable de acuerdo con la zona especificada, y después el programa escoge el patrón de masaje de acuerdo con el parámetro ingresado. Similarmente, se disponen de tres patrones, uno constante, otro que cambia lentamente y otro que alterna entre dos estados cada cierto tiempo.

### Luces por proximidad a beacon:

```

BEGIN
class lucesBeacon{

    int proximidad;
    boolean luz;

    if(proximidad >= 0 and proximidad <= 1){
        luz = false;
    }else
        luz = true;
}
END

```

El programa especifica que al estar a cierta distancia del beacon, la luz se enciende, y al estar alejado a este, se apaga. Esto se logra gracias a la tecnología de los beacons Bluetooth, por lo que gran parte de la funcionalidad se basa en sensores externos a la silla.

**Alerta de paso:**

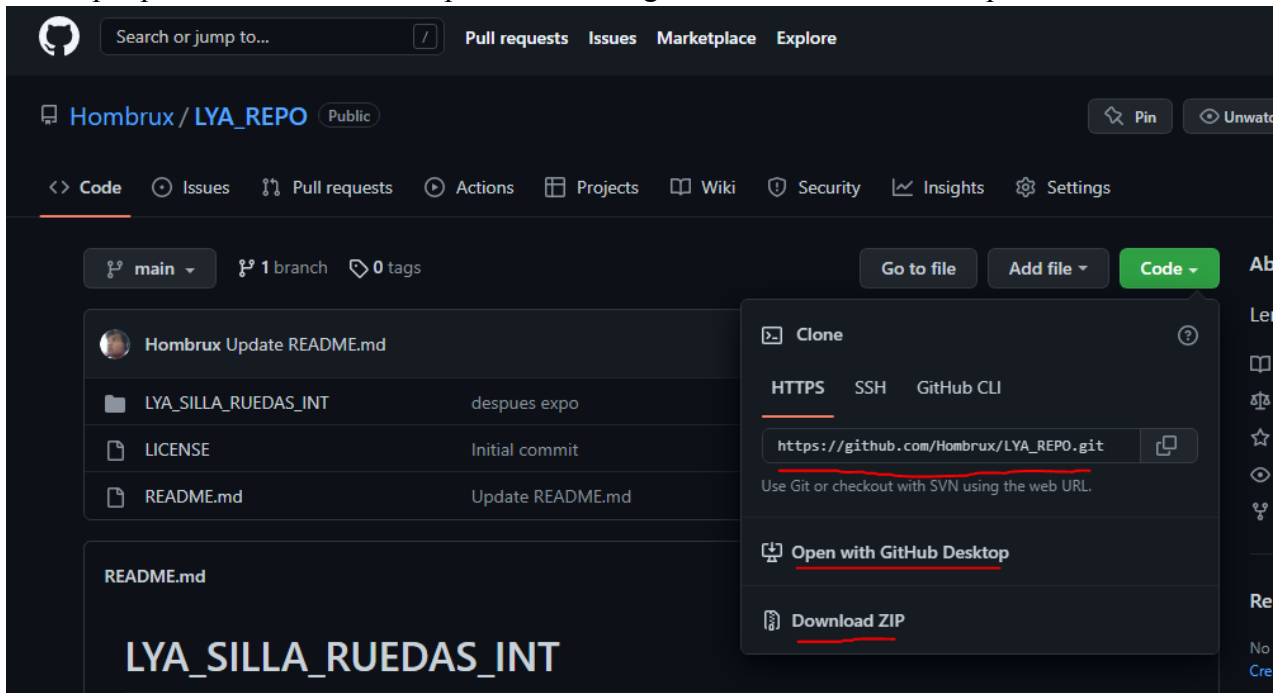
```
BEGIN
class AlertaPaso{
    boolean alerta = false;
    int volumen;
    float duracion;
    string comandoVoz;

    if(comandoVoz == Encender_alerta){
        alerta = true;
        volumen = 85;
        duracion = 15;
    }
}
END
```

Este programa define los valores predeterminados para hacer sonar la alerta para hacer saber a otras personas que el usuario va pasando o aproximándose. Podemos observar que en caso de que el usuario realice explícitamente el comando de voz para encender la alerta, esta se enciende y se establecen los valores de volumen y duración de la alerta. Estos pueden ser cambiados al establecer parámetros especificándolos en el comando de voz.

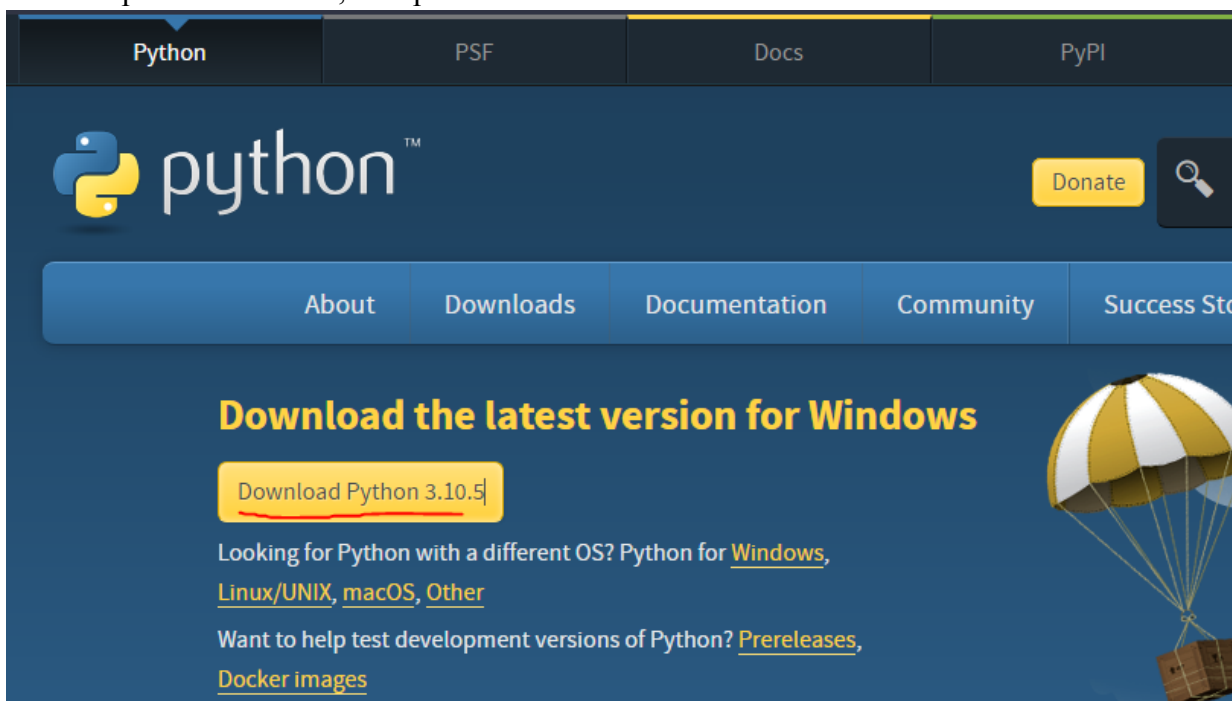
## Manual Tecnico

Para comenzar vamos al repositorio: [https://github.com/Hombrux/LYA\\_REPO](https://github.com/Hombrux/LYA_REPO) que es el mismo que se encuentra en el la portada, una vez dentro del proyecto podemos clonarlo utilizando la URL que se nos proporciona como se ve en la imagen, también podemos abrir el proyecto con GitHub Desktop o por ultimo tenemos la opción de descargar todos los archivos comprimidos en un ZIP.

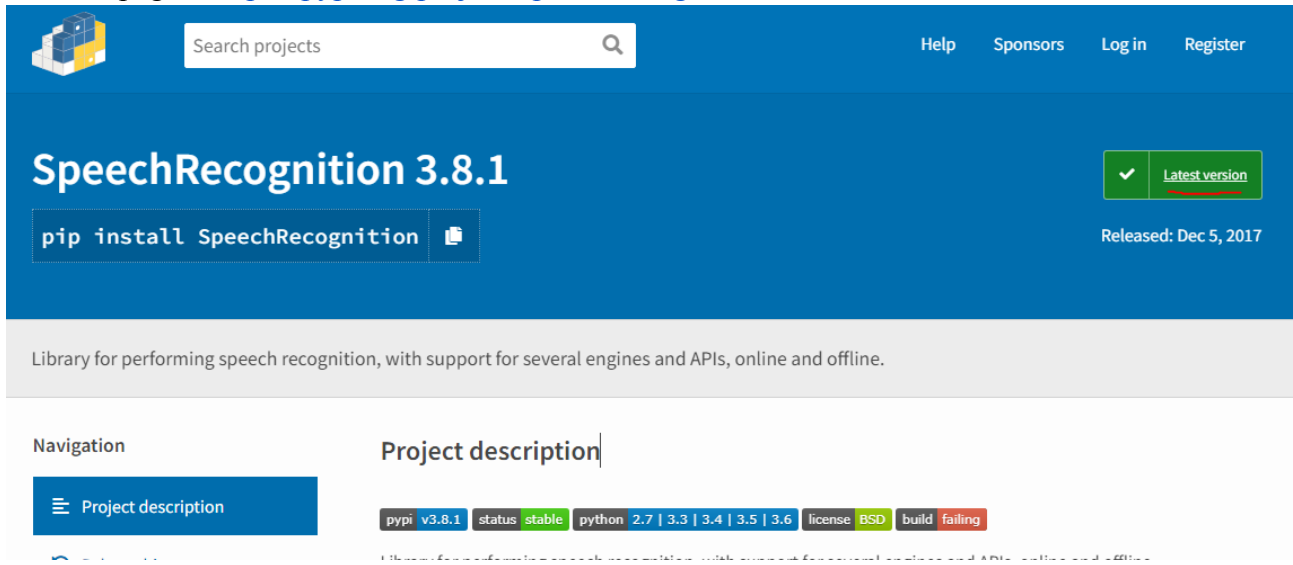


### Instalación de herramientas de desarrollo:

**Python.** - Vamos a la pagina de <https://www.python.org/downloads/> y descargamos python para el sistema operativo deseado, esto para



**Speech Recognition.** - Esta es una librería para el reconocimiento de voz la cual se puede descargar desde su pagina <https://pypi.org/project/SpeechRecognition/>



The screenshot shows the PyPI project page for SpeechRecognition 3.8.1. The header is blue with a search bar and links for Help, Sponsors, Log in, and Register. The main section features the project name 'SpeechRecognition 3.8.1' in large white text, a green 'Latest version' badge, and a button to 'pip install SpeechRecognition'. Below this, a description states: 'Library for performing speech recognition, with support for several engines and APIs, online and offline.' The bottom section has a 'Navigation' sidebar with 'Project description' selected, and a 'Project description' tab showing details like pypi v3.8.1, status stable, python 2.7 | 3.3 | 3.4 | 3.5 | 3.6, license BSD, and build failing.

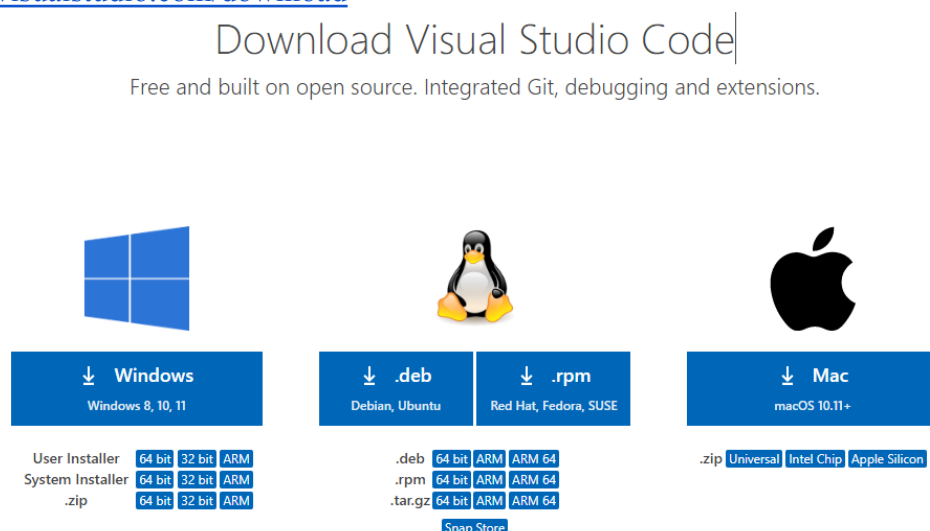
**PyAudio.** - Esta es una librería de la que hace uso speech recognition, para instalarla recomendamos seguir los pasos de este video tutorial: <https://www.youtube.com/watch?v=BaMmiv3sxbQ> ya que en versiones recientes de python no es posible instalar esta librería

**Pillow.** - Esta es una librería que nos permite manipular los distintos tipos de formato de imagen, para instalarla se puede hacer desde la terminal con el comando "pip install Pillow" desde el CMD en Windows o desde la terminal en MAC

```
C:\Users\hombr>pip install Pillow
Requirement already satisfied: Pillow in c:\users\hombr\appdata\local\programs
```

### Ambiente de Desarrollo:

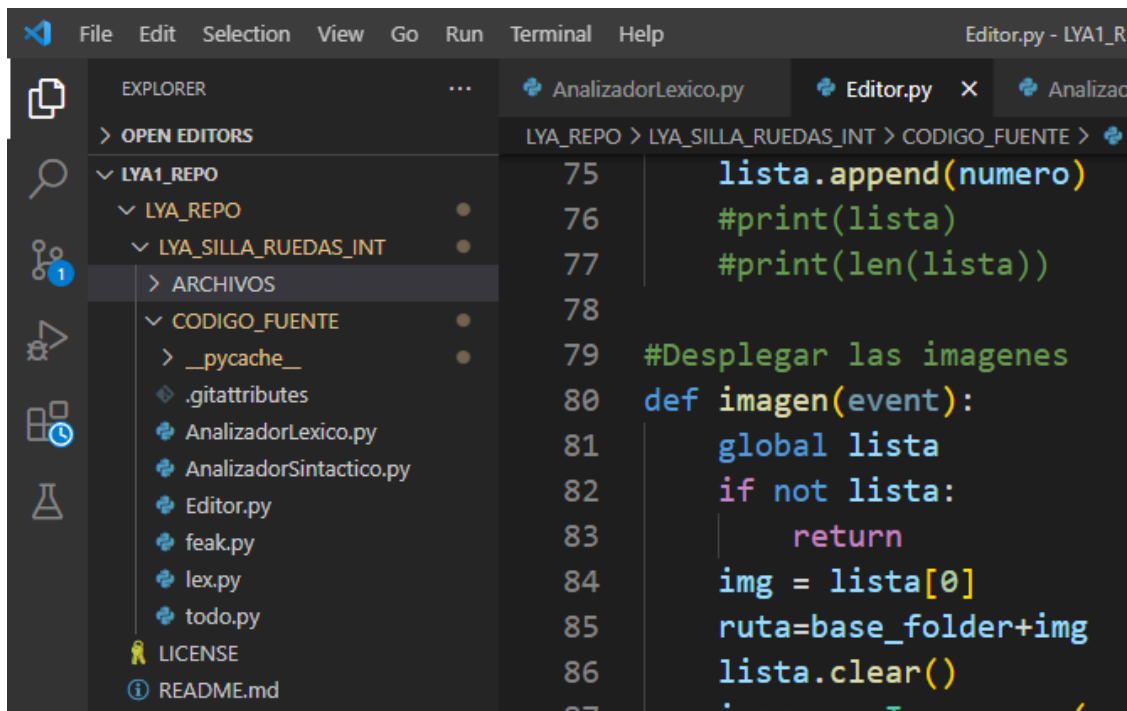
Recomendamos utilizar Visual Studio Code, disponible desde la pagina de Microsoft <https://code.visualstudio.com/download>



The image shows the Visual Studio Code download page. It features the text 'Download Visual Studio Code' and 'Free and built on open source. Integrated Git, debugging and extensions.' Below this, there are three main sections for different operating systems: Windows (with a Windows logo), Linux (with a penguin logo), and Mac (with an Apple logo). Each section lists available download methods and file formats. For Windows, options include User Installer, System Installer, and .zip. For Linux, options include .deb, .rpm, and .tar.gz. For Mac, options include .zip, Universal, Intel Chip, and Apple Silicon. A 'Snap Store' button is also present at the bottom.

Este fue el editor en común usado por los miembros del equipo de desarrollo ya que es fácil de manejar.

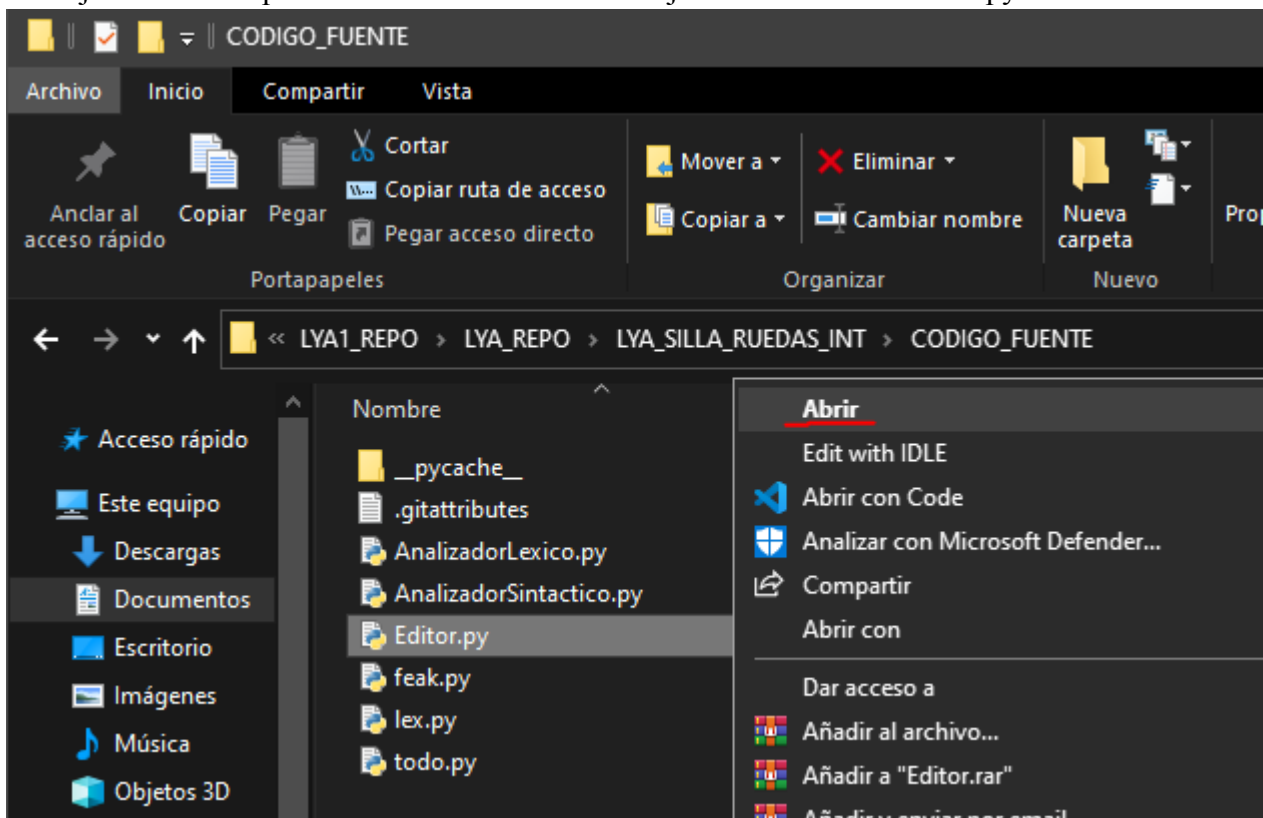
Este editor te permite clonar directamente el repositorio a tu espacio de trabajo lo que facilita el trabajo en equipo.



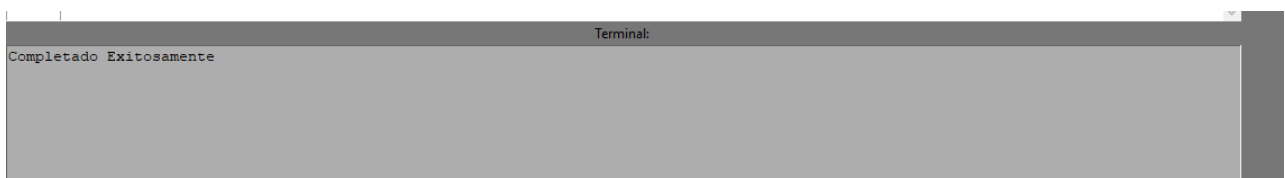
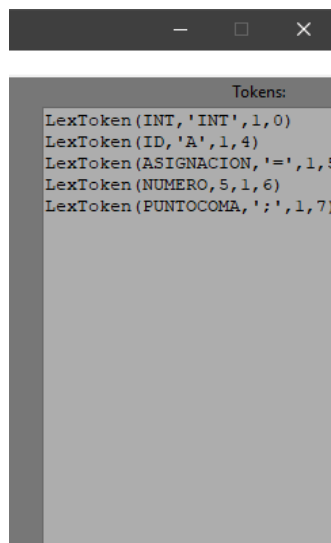
Para ejecutar el proyecto, ejecutamos el archivo editor.py donde podemos ver desde la interfaz gráfica el efecto de los cambios efectuados dentro de los archivos Editor.py, AnalizadorLexico.py y AnalizadorSintactico.py

### Correr el compilador de forma directa a través del ejecutable

Para ejecutar el compilador tan solo hace falta con ejecutar el archivo Editor.py



## Interfaz del usuario



## Conclusión

Después de esta información presentada podemos concluir que fue importante la realización de este trabajo, puesto que la construcción de un analizador léxico y sintáctico para un compilador es de las partes más importantes ya que es casi un buen porcentaje de lo que un compilador es, este análisis asegura que este bien escrito y tengan coherencia los programas escritos en determinado lenguaje de programación.

También esta actividad nos enseñó a como realizar estos analizadores que aunque parezca irónico lo que se tiene que hacer para construirlos es analizar, letra por letra y después palabra por palabra de manera que obtenga un sentido dado por nosotros los programadores.

Asimismo, el desarrollo del proyecto fue un proceso complicado debido a la falta de disponibilidad y falta de tiempo para trabajar en conjunto, pero se logró que el equipo pudiera trabajar eficientemente fuera del calendario escolar ordinario.

## Glosario

- **Analizador léxico:** es la 1ra. fase de un compilador, su principal función es leer los caracteres del código fuente y formarlos en unidades lógicas para que lo aborden las siguientes partes del compilador.
- **Compilador:** es un programa que traduce código fuente escrito en un lenguaje de alto nivel como Java, a un lenguaje legible por la máquina llamado código objeto, lenguaje de destino o incluso lenguaje ensamblador. Por lo tanto, un compilador podría llamarse traductor, pero sus tareas son más amplias porque, como parte de la compilación del programa, también informa de errores al leer el código.
- **Errores léxicos:** se traduce en la generación de un error de sintaxis que será detectado más tarde por el analizador sintáctico cuando el analizador léxico devuelve un componente léxico que el analizador sintáctico no espera en esa posición.
- **Interfaz:** la conexión física y funcional que se establece entre dos aparatos, dispositivos o sistemas que funcionan independientemente uno del otro. En este sentido, la comunicación entre un ser humano y una computadora se realiza por medio de una interfaz.
- **Léxico:** Conjunto de palabras conocidas de un idioma, es decir, su vocabulario, lo que recogen los diccionarios de dicha lengua. Es un conjunto de palabras y de significados asociados, que sin embargo funciona en diferentes niveles: el de la lengua formal.
- **Python:** es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina.
- **Token:** También llamado componente léxico es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación.