USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

MANUAL TECNICO

PROYECTO No. 1
Lenguajes Formales y de Programación

SEPTIEMBRE 2021

SECCION: B+

Ivan Alessandro Hilario Chacón
201902888

TABLA DE CONTENIDO

1. INTRODUCCIÓN

El presente manual técnico contiene la información precisa para describir el funcionamiento de la aplicación designada para el proyecto No.1 del laboratorio del curso Lenguajes Formales y de Programación. Aplicación en la cual se hizo uso del lenguaje de programación Python así mismo implementando autómatas definido y expresiones regulares. Python resulta siendo uno de los lenguajes más entendibles por su sintaxis, que, si se sabe lo básico del idioma inglés, en ocasiones, se puede entender de manera relativamente fácil las porciones de código que estén hechas con este lenguaje.

Sin embargo, en el transcurso del presente manual se explica el funcionamiento de las funciones principales, los algoritmos implementados y el paradigma que se utilizó para concluir esta práctica.

2. PROCESOS

PROCESOS DE ENTRADA

Cargar archivo con extensión lfp
Ejemplo: ⟶

```
Introduccion_a_la_Programación = {
  < "Gerrie Morot" ; 96 >,
  < "Brander De Cristofalo" ; 56 >,
  < "Kiersten Bowld" ; 55 >,
  < "Tanner Caudelier" ; 64 >,
  < "Dag Bernardini" ; 24 >,
  < "Rey Tweedle"  ; 58 >
} REP,APR,AVG,MIN
```

PROCESOS DE SALIDA

Despliegues de Imágenes
Despliegues de Reporte de Errores y Tokens
Despliegue de Reporte HTML

3. REQUERIMIENTOS DEL SISTEMA

REQUERIMIENTOS DE HARDWARE

Equipo, teclado, mouse,
monitor. Memoria RAM 2GB.
Procesador 1.4 GHz.

REQUERIMIENTOS DE SOFTWARE

Sistema operativo (Windows 7 en
adelante). Python 3.0

4. HERRAMIENTAS UTILIZADAS

- Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

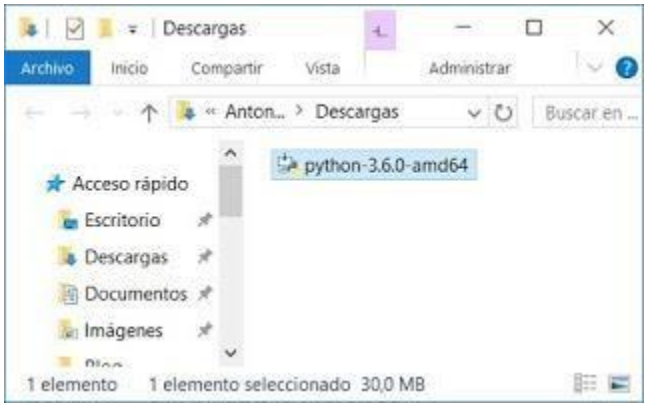
- Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

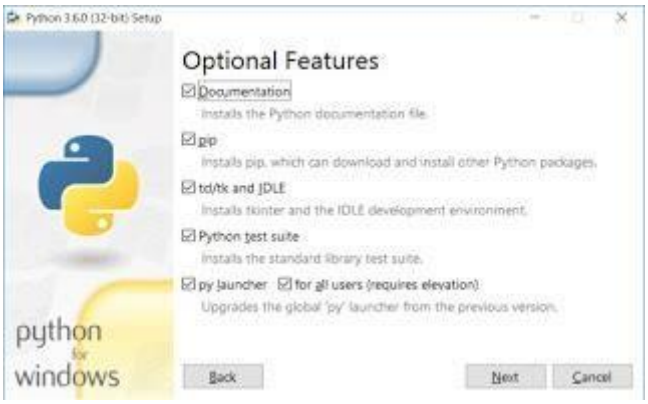
5. INSTALACIÓN DE APLICACIÓNES

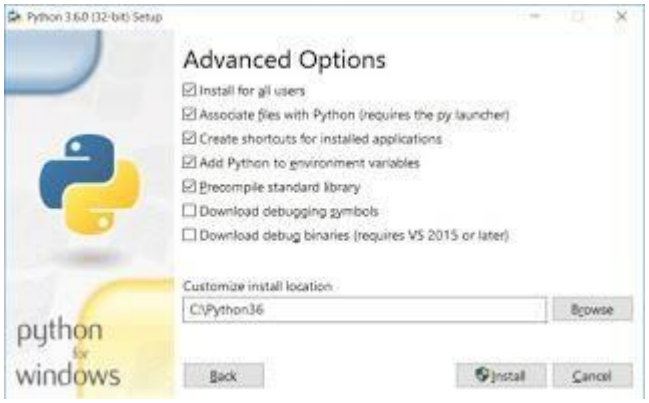
Para descargar Python en Windows:
https://www.python.org/downloads/

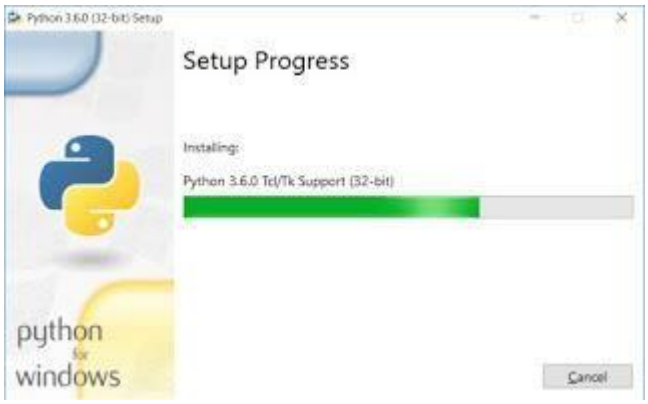
Abrir el ejecutable



Iniciar la instalación





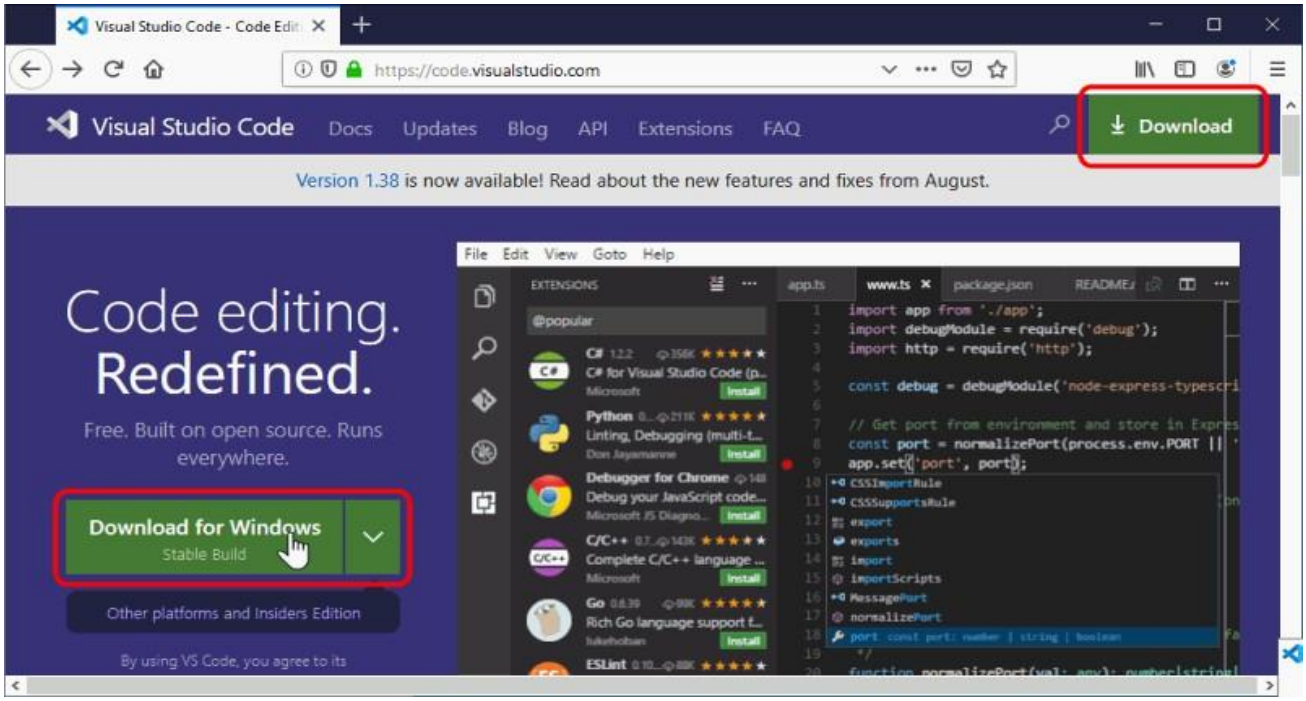


Se concluye con la Instalación:

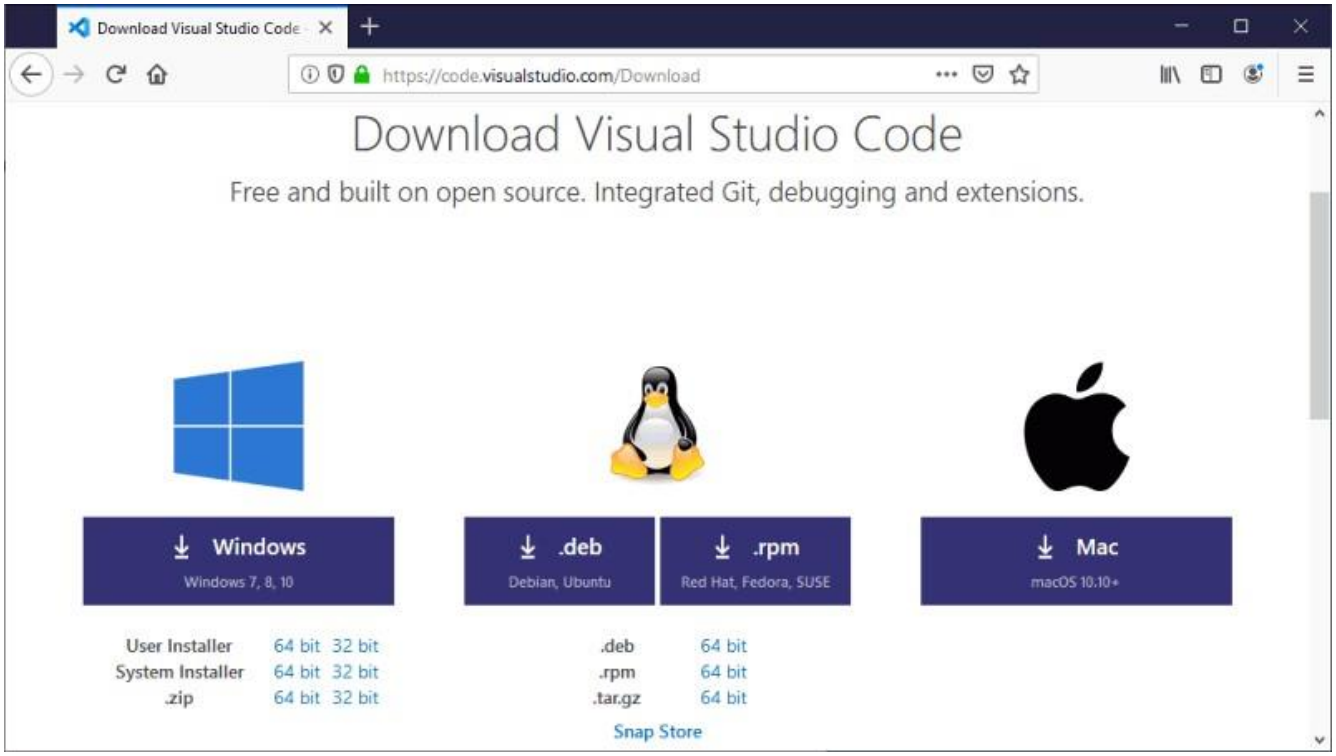
Para descargar e instalar Visual Studio Code en Windows:

Página oficial para la descarga:

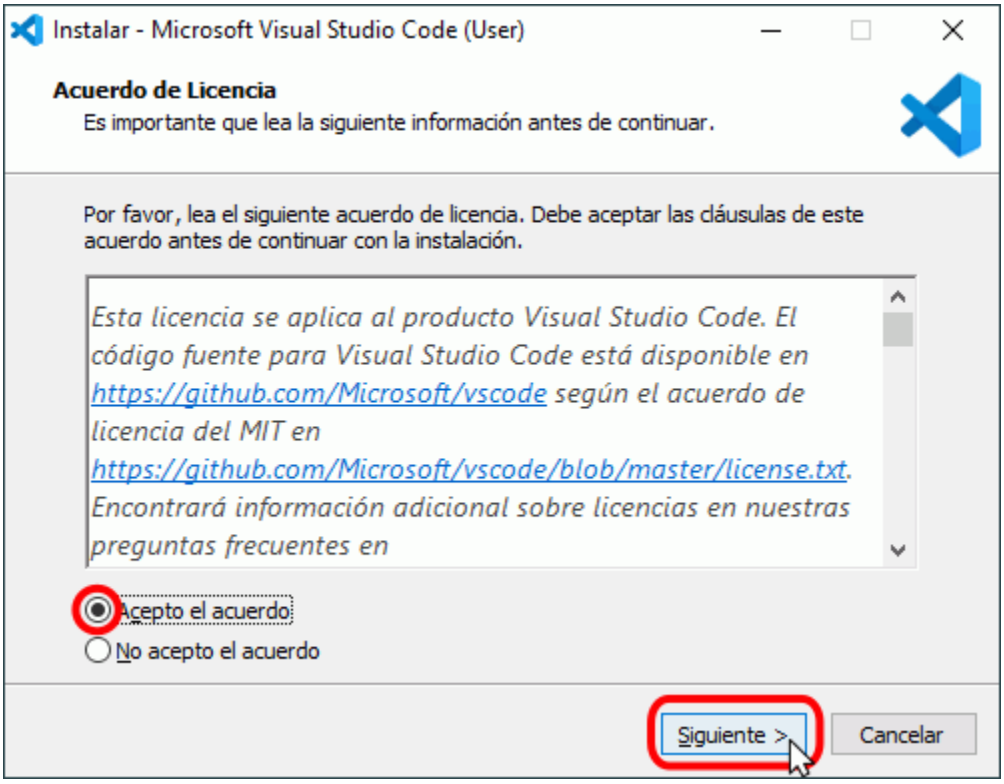
https://code.visualstudio.com/

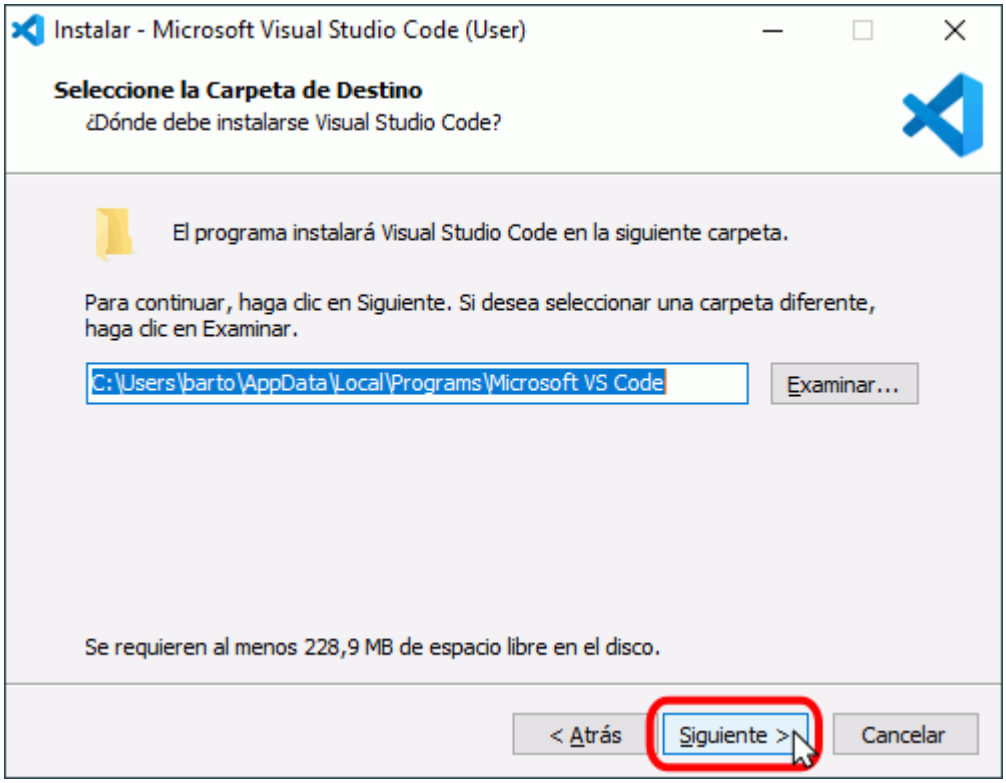


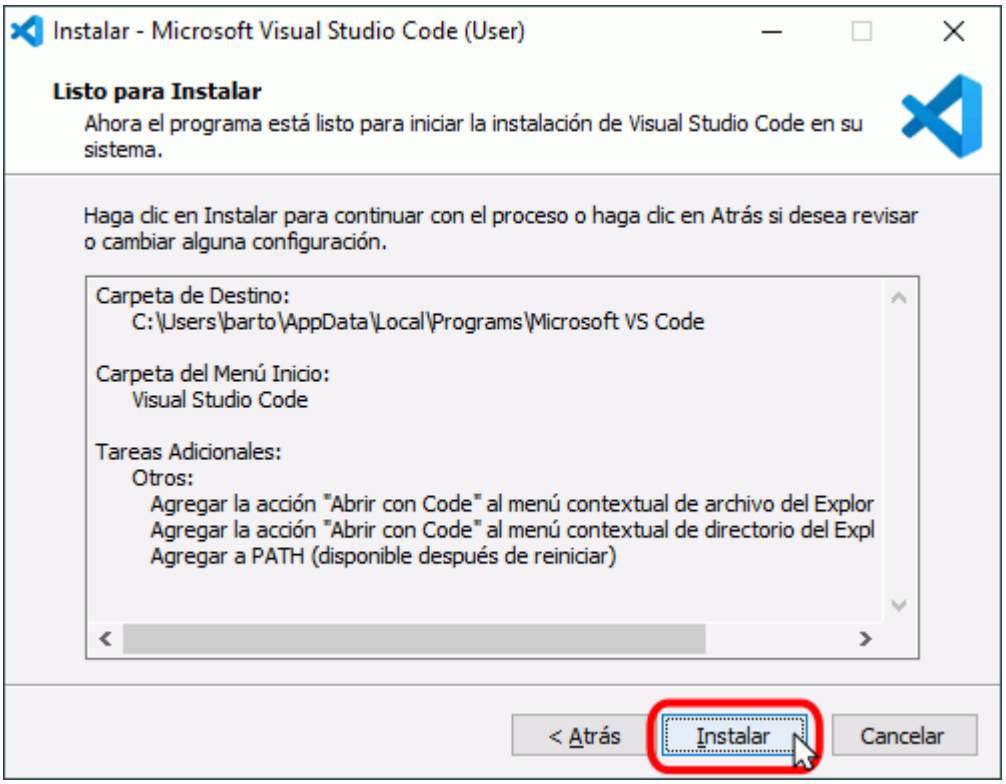
Seleccionamos para Windows



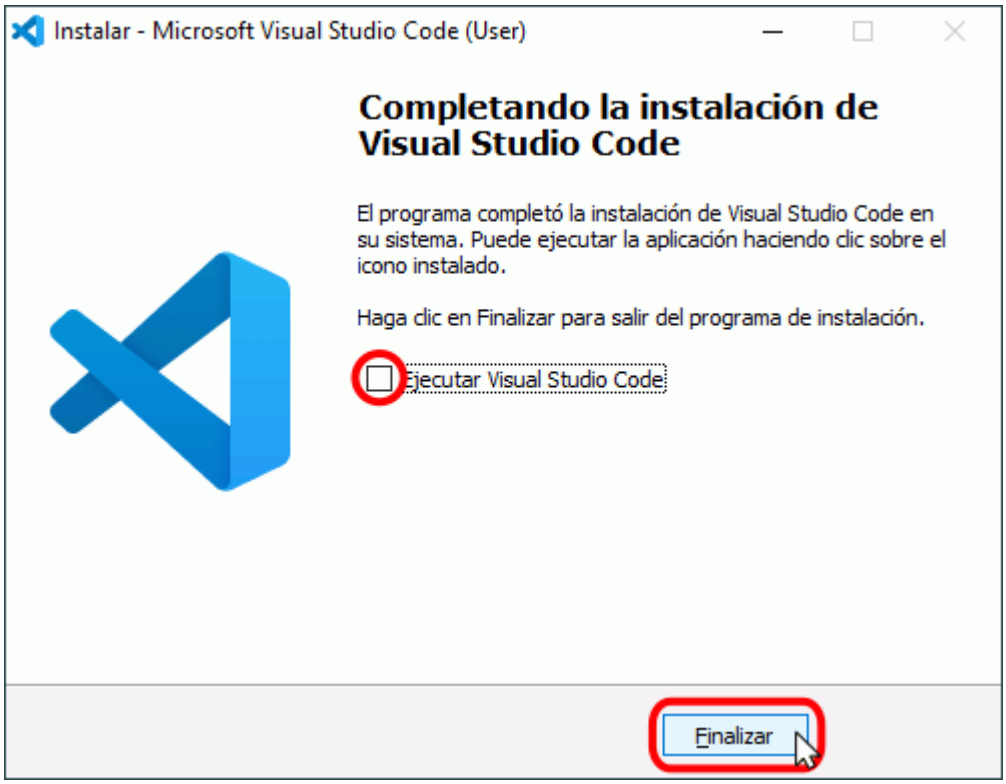
Y se siguen los siguientes pasos:







Y se finalizará de instalar VS Code



6. FUNCIONES

- Variables: creación de las variables para el manejo de información dentro del programa

```
global listaIMAGEN
    HtmlGeneral = ""
    HTMLImgagen =""
    titulo = ""
    ancho = 0
    alto = 0
    filasY = 0
    columnasX = 0
    filtros = ""
    yfila = 0
    xcolumna = 0
    confirmacion = []
    altoCelda = 0
    anchoCelda =  0
```

- Abrir Reportes: Función implementada para el uso de abrir automáticamente reportes

```
def abrirReportes():#ABRIR REPORTE
    webbrowser.open_new("Reportes.html")
    webbrowser.open_new("ReportesErrores.html")
    webbrowser.open_new("Imagenes.html")
```

- Separa: dicha función recibe un contenido y carácter retorna un lista spliteda por el carácter enviada a dicha función

```
def splitear(cadena, caracter):#FUNCION QUE SEPARA CONTENIDO POR CARACTER
    temporal = ""
    listaTemporal = []
    for i in cadena:
        if i == caracter:
            listaTemporal.append(temporal.strip())
            temporal = ""
        else:
            temporal += i
    if temporal.strip() != "":
        listaTemporal.append(temporal.strip())
    return listaTemporal
```

- CrearArchivo : función que recibe el contenido escrito por html y ruta para así poder crear dicho reporte dentro del programa

```
def CrearArchivo(ruta, contenido):#ESCRITURA DE ARCHIVO PARA REPORTES
    archivo = open(ruta, 'w')
    archivo.write(contenido)
    archivo.close
```

- Clase Celdas: Clase creada para el manejo de información que contendrá posiciones y código hexagesimal

```
class Celdas(object):
    def __init__(self,x,y,estado,codigo):
        self.x = x
        self.y = y
        self.estado = estado
        self.codigo = codigo
```

- Class Analizador: clase creada para el manejo de lista de token y errores

```
class AnalizadorLexico:
    def __init__(self):
        self.listaTokens = []
        self.listaErrores = []

    def analizar(self, codigo_fuente):
        self.listaTokens = []
        self.listaErrores = []
```

- BuscarArchivo: abre una ventana emergente para poder seleccionar dicho archivo a leer y así poder retornar el contenido de dicho archivo para su manejo

```
def Buscar_archivo():#FUNCION QUE REALIZAR LA SELLECION DEL ARCHIVO A LEER Y RETORNA
    Tk().withdraw()
    archivo = filedialog.askopenfile(
        title = "Seleccionar un archivo LFP",
        initialdir = "./",
        filetypes = (
            ("Archivos lfp", "*.lfp"),))
    if archivo is None:
        print('No se seleccionó ningun archivo\n')
        return None
    else:
        texto = archivo.read()
        archivo.close()
        print('Lectura exitosa\n')
        return texto
```

- Clase Token: Clase creada para el manejo de token ingresado

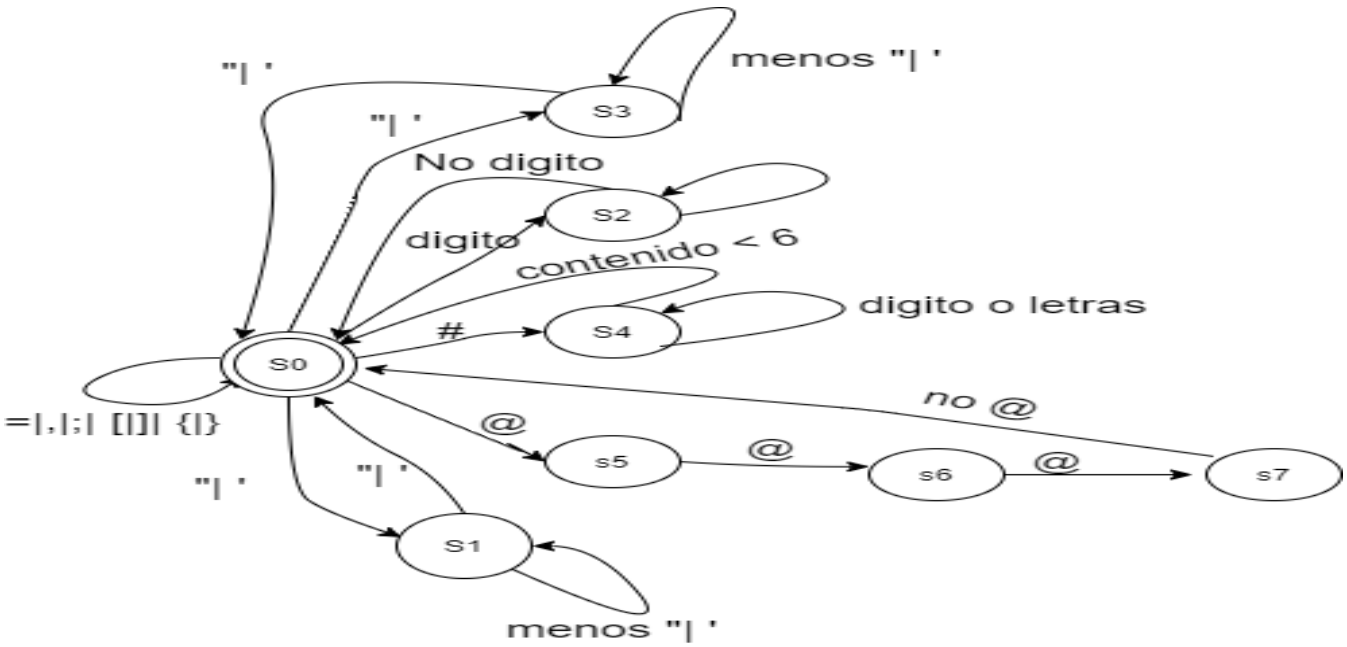
```
class Token:
    def __init__(self, lexema, tipo,  linea, columna):
        self.lexema = lexema
        self.tipo = tipo
        self.columna = columna
        self.linea = linea
```

7. PARADIGMA DE PROMACIÓN

Para el desarrollo de esta aplicación se utilizó el paradigma de programación orientado a objetos, por su fácil manejo, su versatilidad y su reducción de líneas de código.

Este paradigma está basado en varias técnicas del sexenio: herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

8. AUTOMATA DEFINIDO



9. TABLA DE TOKENS

TOKEN	PATRON
TITULO	palabra TITULO
ANCHO	palabra ANCHO
ALTO	palabra ALTO
FILAS	palabra FILAS
COLUMNAS	palabra COLUMNAS
CELDAS	palabra CELDAS
FILTROS	palabra FILTROS
MIRRORX	palabra MIRRORX
MIRRORY	palabra MIRRORY
DOUBLEMIRROR	palabra DOUBLEMIRROR
igual	signo igual
puntocoma	caracter punto y coma
llavea	caracter llave que abre
llavec	caracter llave que cierra
corchetea	caracter de corchete de abertura
corchetec	caracter de corchete de cierre
coma	caracter coma
arroba	caracter de arroba separación
cadena	inicia y termina con " o '
entero	secuencia de digitos
codigo	codigo color hexagesimal
error	cualquier cosa no sea lenguaje