Manual de uso:

El código está programado en el lenguaje de Python y toda la implementación de los autómatas está pensado para reconocer código de Java

La forma de utilizar el programa es la siguiente:

1. Descargar el código y guardarlo en una carpeta en específico.
2. En la primera línea del código, donde aparece la variable **archivo**, cambiar el “automataV.txt” por el nombre del archivo de texto que se quiera evaluar, es importante recalcar que ambos archivos, tanto el de texto como el código deben de estar en la misma carpeta
3. Ejecutar en el respectivo IDE o consola de comandos

Autómata validación de escritura:

Símbolos de entrada: { A-Z / $ / _ , 0-9 , Espacio , **,** , “” , ‘’ ,= , + - * / , & | , > < , ; **}**

- **Estados = {**
  - **S:** Hilera vacía
  - **A:** Primer carácter valido del nombre de una variable
  - **B:** Espacio o espacios después del nombre de la variable
  - **C:** Coma después del nombre de la variable o de los espacios siguientes a la misma
  - **D:** Nombre de variable o constante después del igual
  - **E:** Signos, letras o números que vayan dentro de comillas dobles
  - **F:** Carácter después del cierre de comillas dobles
  - **G:** Carácter después de la aparición de un operador aritmético en la parte derecha del igual
  - **H:** Primer carácter valido del nombre de una variable al lado derecho del igual
  - **I:** Espacio o espacios después de un nombre de variable, de cierre de comillas simples, comillas dobles o constantes
  - **J:** Primer digito de una constante al lado derecho del igual
  - **K:** Carácter después del carácter “&” o “ | ”
  - **L:** Carácter después de ingresados la combinación “&&” o “| |”
  - **M:** Próximo carácter después de “=” o “!”

- **O:** Próximo carácter después de “<” o “>”
- **P:** Próximo carácter después de ingresados “+” o “-” seguidos del nombre de la variable
- **Q:** Carácter siguiente al ingreso del operador de incremento o decremento
- **R:** Apertura de comillas simples
- **T:** Signo, letra o número que vaya dentro de comillas simples
- **U:** Carácter después del cierre de comillas simples
- **X:** Identificacion del comienzo de un “if”
- **Z:** Elemento despues del cierre de los paréntesis de la condición del if
- **Z2:** Identificacion del comienzo de un “else”
- **¬:** Carácter después de ingresados el “ ; ” o “¬

}

Estado inicial = S

Estado de aceptación = ¬

Tabla de transiciones:

	A-Z / $ / _	0-9	ESPACIO	,	"	=	+/-/*/	&/|	!	>/<	'	;	(	)	{	}	if	else
S	A		S												Ap		X	Z2
A	A	A	B	C		D	P					¬						
B			B	C		D	P					¬						
C	A		C															
D	H	J	D		E						R		Ap	Des				
E	E	E	E	E	F	E	E	E	E	E	E	E	E	E	E	E	E	E
F			I	C		M	G	K	M	O		¬		Des				
G	H	J	G		E						R		Ap					
H	H	H	I	S		M	G	K	M	O		¬		Des				
I			I	S		M	G	K	M	O		¬		Des				
J		J	I	S		M	G	K	M	O		¬		Des				
K						D		L										
L	D	D	D		E						R		Ap					
M						D												
O	H	J	D			D							Ap					
P						D	Q											
Q			Q									¬						
R	T	T	T	T	T	T	T	T	T	T	U	T	T	T	T	T	T	T
T											U							
U			I	C		M	G	K	N	O		¬						
X			X										D/ap				X	
Z			Z												¬			
Z2															¬			Z2
¬			¬									¬				¬		Z2

Particularidades:

- No reconoce que los paréntesis estén equilibrados, por lo que los ignora, es decir, todas las apariciones de los signos de paréntesis en una línea de código serán ignoradas por lo que líneas como:

- a = (5 + 3) ; serán válidas, pero a su vez
- b = (2323 (+ 87 (((( ) ((; también serán validas

❖ No reconoce números decimales

❖ El reconocimiento de “While” no lo hace este autómata, por lo que tiene la pequeña particularidad que toda línea que empiece por “while” es ignorada por este autómata para evitar que la reconozca como errónea.

❖ Reconoce casos particulares como:

- x += 4; pero no hace una distinción de una asignación convencional cuando aparece una coma, por lo que líneas como
- x += 2, 878, 8798; son validas

❖ Tiene una particularidad la cual que reconoce while e if pero con la condición de que el “{“ debe ir al final de la línea donde se invoca, de modo que:

- If (a < b) { ← Es valido
- If (a < b) ← Es invalido

Lista de errores:

ERROR1: El nombre de la variable empieza con un carácter invalido

ERROR2: Uso de signos prohibidos en el nombramiento de la variable

ERROR3: Uso de espacios en el nombramiento de la variable

ERROR4: Uso de signos prohibidos

ERROR5: Uso de signos prohibidos después del cierre de comillas

ERROR6: Uso de signos prohibidos después de un operador aritmético

ERROR7: Uso indebido de comillas

ERROR8: Uso de signos que no son el respectivo operador lógico

ERROR9: Uso de signos que no son el respecto '='

ERROR10: Signos inválidos después de los operadores de incremento o decremento '++' / '--'

ERROR11: Cantidad de caracteres mayor de la permitida

ERROR12: Línea vacía

ERROR13: Falta el operador de cierre ' ; '

ERROR14: Uso de palabra reservada if

ERROR15: Línea desbalanceada

ERROR16: Lista desbalanceada / Falta de '{' o '}'

ERROR17: Comillas sin cerrar

Autómata validación de cada token:

Palabras reservadas o tipos de variables

Símbolos de entrada: { b, o, l, i, c, d, f, n, r, u, e, t, g, a, h**}**

- **Estados = {**
  - **S**
  - **B**
  - **O1**
  - **L**
  - **O2**
  - **U**

- **N1**
- **R**
- **E1E2**
- **T**
- **G**
- **N2**
- **I**
- **C**
- **F1**
- **F2**
- **H**
- **D**
- **E**
- **AE**
- **N2E**
- **RE**
- **F**

Explicación método de balanceo paréntesis:

1. Se crea una lista
2. Al momento de analizar carácter por carácter la línea correspondiente, cuando se encuentre un ‘(‘, agregar un elemento cualquiera a la lista(Apilar)
3. Al momento de que se encuentre un ‘)’, eliminar un elemento de la lista(Desapilar)
4. Al final del proceso preguntarse si la lista quedo vacía, si no lo está, significa que algún paréntesis quedo sobrando

5. También asegurarse al momento de desapilar que la lista no este vacía, ya que se estaría eliminando un elemento de una lista que esta vacía(contradicción), este caso también es motivo de error

Estado inicial = S

Estado de aceptación = R, E1E2, T, G, N2, F2, N2E, RE

Tabla de transiciones:

	b	o	l	i	c	d	f	n	r	u	e	t	g	a	h
S	B		L	I	C	D	F1								
B															
O1		O2	L					N1	RE	U					
L		O1									E1E2				
O2			L												
U	B														
N1												T	G		
R															
E1E2														AE	
T															
G															
N2															
I							F2	N1							
C															H
F1		O1													
F2															
H														AE	
D		O1													
E															
AE								N2E	RE						
N2E															
RE															
F															

Nota: El estado en blanco representa el estado de error F.

Separadores

Símbolos de entrada: { ,, ;}

- **Estados = {**
  - **SE0: Hilera vacía**
  - **SE1: Introdujo un separador válido**
  - **SE2: Introdujo un carácter inválido o múltiples “,”**

  }

Estado inicial = SE0

Estado de aceptación = SE1

Tabla de transiciones:

	,	;	
SE0	SE1	SE1	0
SE1	SE2	SE2	1
SE2	SE2	SE2	0

Operadores

Símbolos de entrada: { +, -, *, =, !, <, >, &, ^, |, ~, /, %}

- **Estados = {**
  - **OP0**
  - **OP1**
  - **OP2**
  - **OP3**
  - **OP5**
  - **OP6**
  - **OP7**
  - **OP8**
  - **OP9**
  - **OP10**
  - **OP11**
  - **OP14**
  - **OP16**
  - **OP18**

  }

Estado inicial = OP0

Estado de aceptación = OP1, OP2, OP3, OP6, OP7, OP8, OP9, OP10, OP11, OP14, OP16

Tabla de transiciones:

	+	-	*	=	!	<	>	&	^	|	~	/	%	
OP0	OP1	OP2	OP3	OP3	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP3	OP3	0
OP1	OP14			OP14										1
OP2		OP14		OP14										1
OP3				OP14										1
OP5				OP14										0
OP6				OP14		OP3								1
OP7				OP14			OP16							1
OP8				OP14				OP14						1
OP9				OP14					OP14					1
OP10				OP14						OP14				1
OP11														1
OP14														1
OP16				OP14			OP3							1
OP18														0

Nota: El estado en blanco representa el estado de error OP18.

Booleanos

Símbolos de entrada: { t, r, u, e, f, a, l, s}

- **Estados = {**
  - **BO0**
  - **BO1**
  - **BO2**
  - **BO3**
  - **BO4**
  - **BO5**
  - **BO6**
  - **BO7**
  - **BO8**

  }

Estado inicial = BO0

Estado de aceptación = BO4

Tabla de transiciones:

	t	r	u	e	f	a	l	s	
BO0	BO1				BO5				0
BO1		BO2							0
BO2			BO3						0
BO3				BO4					0
BO4									1
BO5						BO6			0
BO6							BO7		0
BO7								BO4	0
BO8									0

Nota: El estado en blanco representa el estado de error BO8.

Constantes

Símbolos de entrada: { A-Z / $ / _ , 0-9 , Espacio , **,** , “” , ‘’ ,= , + - * / , & | , > < , ; **}**

Estados = {

- **C0**
- **C1**
- **C2**
- **C3**
- **C6**
- **C7**
- **C8**
- **C9**
- **C11**
- **C12**
- **C13**
- **C14**
- **C15**
- **C16**

}

Estado inicial = C0

Estado de aceptación = C2, C3, C7, C13, C14, C15

Tabla de transiciones:

	num	.	+ -	"	'	alpha	other	
C0	2	8	1	10	11			0
C1	2	8						0
C2	2	3						1
C3	3							1
C6	7							0
C7	7							1
C8	3							0
C9								0
C11	14	14	14	14	15	14	14	0
C12	12	12	12	13	12	12	12	0
C13								1
C14					15			1
C15								1
C16								0

Nota: El estado en blanco representa el estado de error C16.

Variable y tipo; o variable

Símbolos de entrada: { A-Z / $ / _ , 0-9 , Espacio , **,** , “” , ‘’ ,= , + - * / , & | , > < , ; **}**

- **Estados = {**
  - **TV1**
  - **TV2**
  - **TV3**
  - **TV4**

  }

Estado inicial = TV1

Estado de aceptación = TV2

Tabla de transiciones:

	TIPO	VARIABLE	OTHER	
TV1	TV4	TV3	TV3	0
TV2	TV2	TV2	TV2	1
TV3	TV3	TV3	TV3	0
TV4	TV1	TV2	TV1	0

TV2 representa el estado cuando la línea de código posee las clases **tipo** y **variable** al inicio de la línea, mientras que TV3 detecta que la línea de código posee al menos la clase **variable**.