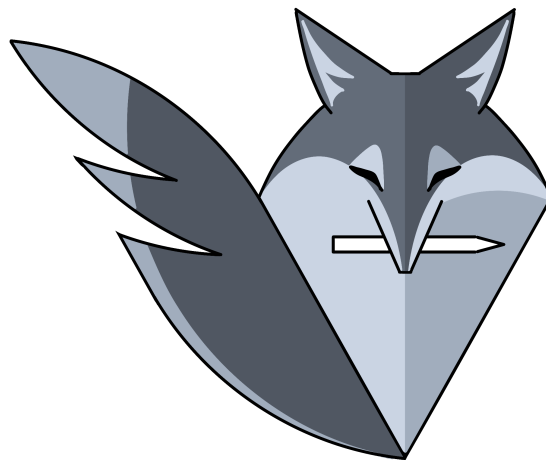




TC2037 - Implementación de métodos computacionales - Gpo 820

Evidencia #1: Diseño e implementación básica de un DSL para enseñar a programar a niños






LOBO

Karen Priscila Navarro Arroyo | A01641532

20/03/2023

[1] Elementos del léxico

Código de lexema	Lexema	Expresión regular (Python)	Descripción
Números			
100	<i>int</i>	<code>r'^[\d]*\$'</code>	Se refiere a cualquier valor numérico. Debe ser un número entero mayor a 0.
Movimiento			
101	<i>av</i> <i>avanza</i>	<code>r'^av\$ ^avanza\$'</code>	Avanza hacia el frente en el valor especificado.
102	<i>rv</i> <i>reves</i>	<code>r'^rv\$ ^reves\$'</code>	Avanza en sentido contrario a la dirección actual en el valor especificado.
103	<i>gd</i> <i>girod</i>	<code>r'^gd\$ ^girod\$'</code>	Cambia la dirección a la derecha en el ángulo especificado (Ej. <i>gd 90</i> hace un giro de 90 grados a la derecha ).
104	<i>gi</i> <i>giroi</i>	<code>r'^gi\$ ^giroi\$'</code>	Cambia la dirección a la izquierda en el ángulo especificado.
Pluma y posición			
105	<i>ct</i> <i>centro</i>	<code>r'^ct\$ ^centro\$'</code>	Regresa al centro, la posición original. (Nota: Puede dibujar una línea desde su posición actual hasta el centro o no, dependiendo del modo de la pluma). Posición original: 
106	<i>br</i> <i>borrar</i>	<code>r'^br\$ ^borrar\$'</code>	Reinicia el <i>canvas</i> a su estado inicial, borrando cualquier dibujo que hubiera previamente y regresando a la posición original: 
107	<i>nd</i>	<code>r'^nd\$'</code>	No dibujar. Aún se puede seguir moviendo por el canvas pero sin dibujar ninguna línea.
108	<i>sd</i>	<code>r'^sd\$'</code>	Sí dibujar. Permite volver a dibujar en el canvas. Éste es el estado de dibujo por defecto de la pluma.

Color			
109	color	r'(^color\$)'	Cambia el color de la pluma a cualquiera de sus 15 colores disponibles (201-215).
Colores			
201	tinto	r'(^tinto\$)'	HEX: #770606
202	rojo	r'(^rojo\$)'	HEX: #db0e0e
203	naranja	r'(^naranja\$)'	HEX: #eb5d0b
204	amarillo	r'(^amarillo\$)'	HEX: #e7ca0e
205	verde	r'(^verde\$)'	HEX: #26820f
206	lima	r'(^lima\$)'	HEX: #0ee91d
207	turquesa	r'(^turquesa\$)'	HEX: #12ad79
208	azul	r'(^azul\$)'	HEX: #1319c4
209	morado	r'(^morado\$)'	HEX: #730fad
210	lila	r'(^lila\$)'	HEX: #c876ff
211	rosa	r'(^rosa\$)'	HEX: #ed55ba
212	cafe	r'(^cafe\$)'	HEX: #53220b
213	negro	r'(^negro\$)'	HEX: #000000
214	gris	r'(^gris\$)'	HEX: #888888
215	blanco	r'(^blanco\$)'	HEX: #ffffff
Repetir			
110	repetir	r'(^repetir\$)'	Repite lo que esté entre corchetes ([]) la cantidad de veces indicada. (Ej. repetir 4[av 100 gd 90] dibujará un cuadrado).
301	[r'(^\[\$)'	Marca el inicio de las indicaciones que se repetirán en el ciclo.
302]	r'(^\[\$)'	Marca el fin de las indicaciones que se repetirán en el ciclo.

[2] Autómata determinístico de proceso de análisis del léxico

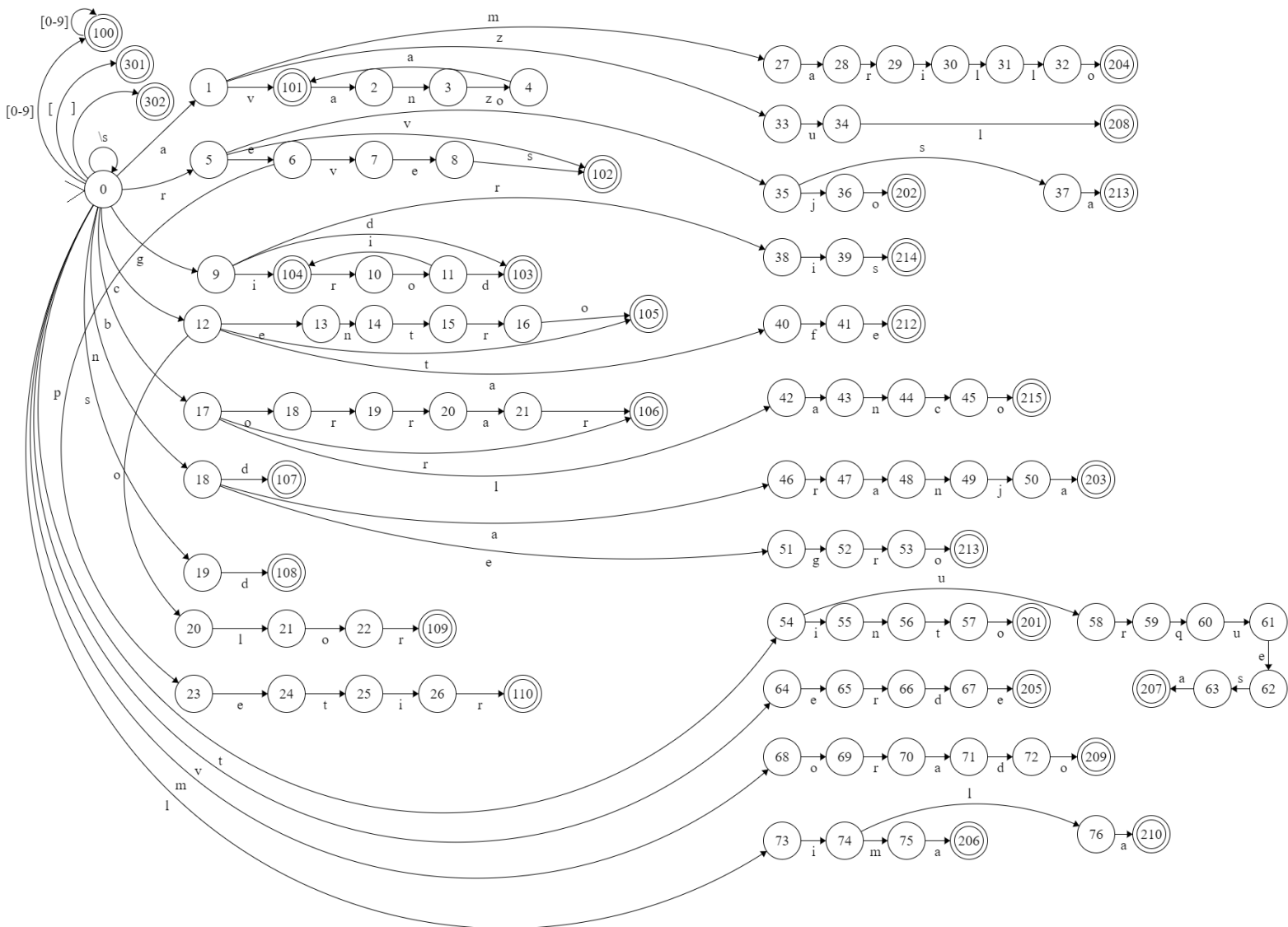


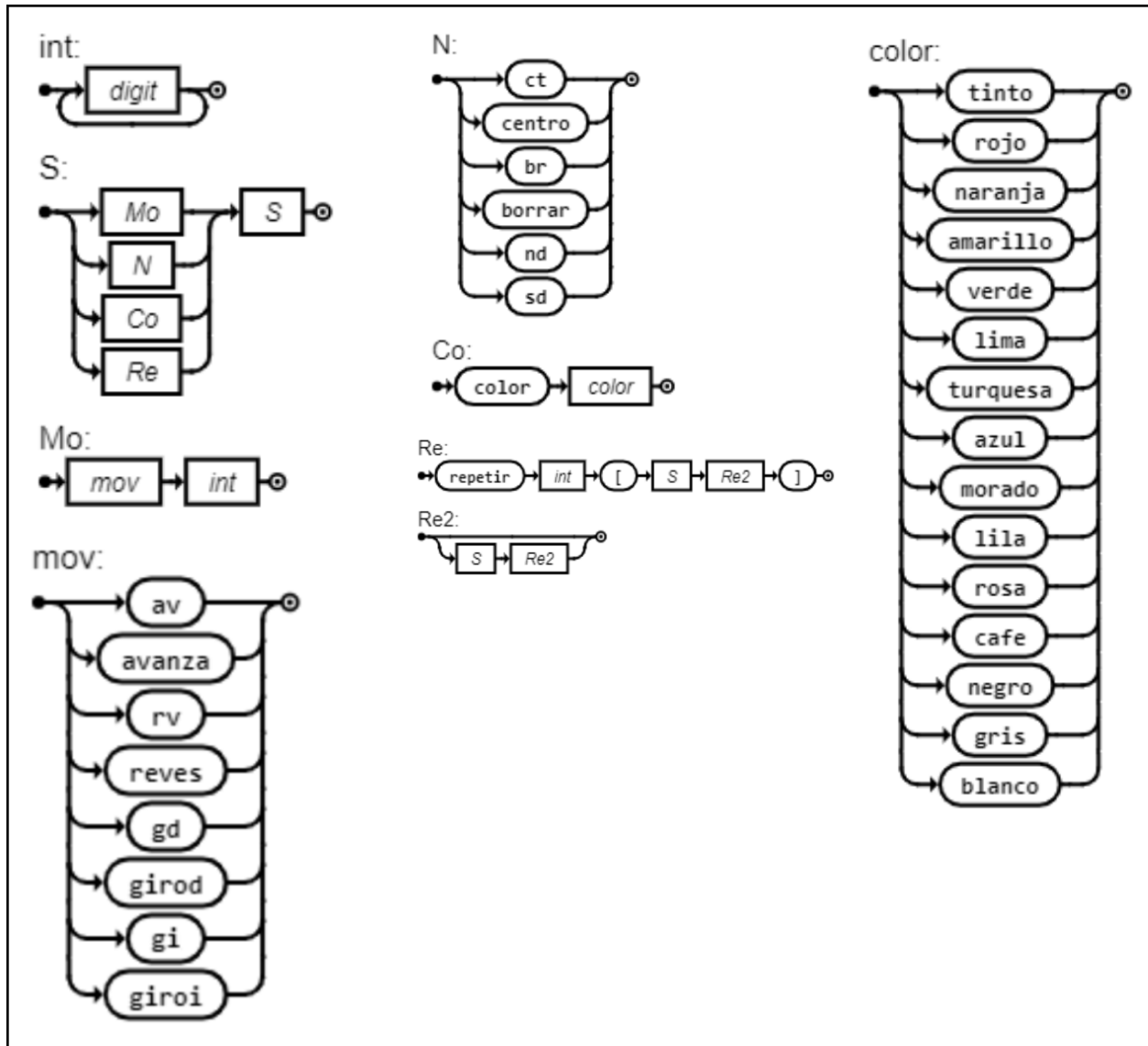
Imagen con mejor calidad:

<https://drive.google.com/file/d/1EcFOcTgNWYRbMPtdiNhIbHbShyGN-ms0/view?usp=sharing>

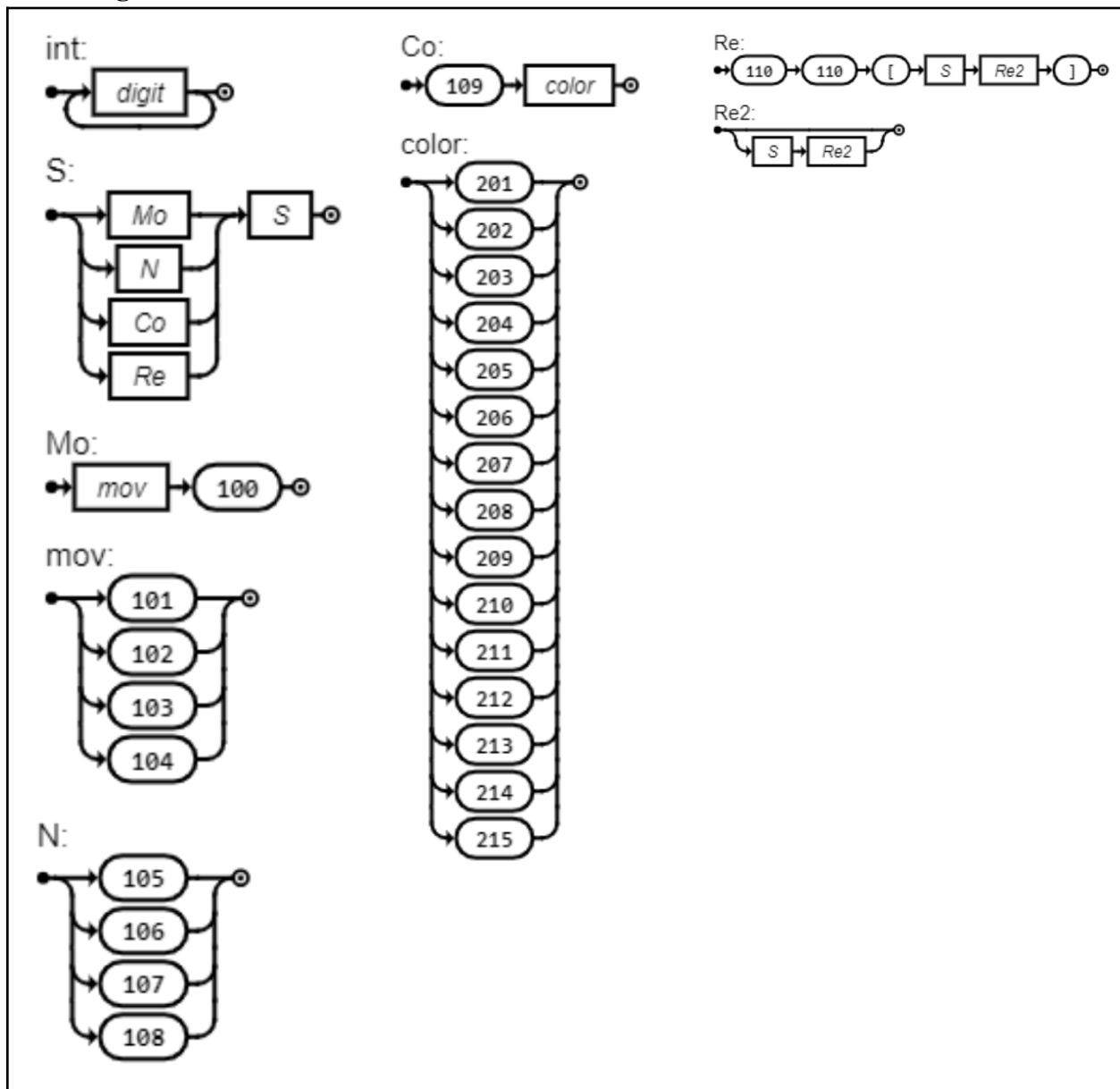
[3] Sintaxis

[3.1] Diagramas de sintaxis

Con lexemas:



Con códigos de lexema:



[3.2] Gramática BNF

Código	Nombre	Pertenece a <BNF>		
101	av, avanza	<Mo>	<S> ::= (<Mo> <N> <Co> <Re>) (<S> ε) <Mo> ::= <mov> 100 <mov> ::= 101 102 103 104	
102	rv, reves			
103	gd, girod			
104	gi, giroi			
105	ct, centro	<N>	<N> ::= 105 106 107 108	
106	br, borrar			
107	nd			
108	sd			
109	color	<Co>	<Co> ::= 109 <color> <color> ::= 201 202 209 203 210 204 211 205 212 206 213 207 214 208 215	
110	repetir	<Re>		<Re> ::= 110 100 '[' <S><Re2> ']' <Re2> ::= <S><Re2> ε
301	[
302]			
100	int	-		
*Durante el análisis de léxico, los lexemas se traducen a los códigos de 100-302, con los que después se realiza el análisis de sintaxis como aparece en el diagrama.			*En <Re2>, ε es cuando el contador deja de ser mayor a 0 (n>0) y el ciclo repetir se termina (ver código .py).	

Gramática con lexemas:

$\langle S \rangle ::= (\langle Mo \rangle | \langle N \rangle | \langle Co \rangle | \langle Re \rangle) (\langle S \rangle | \epsilon)$

$\langle Mo \rangle ::= \langle mov \rangle \text{ int}$

$\langle mov \rangle ::= 'av'$

| 'avanza'

| 'rv'

| 'reves'

| 'br'

| 'borrar'

$\langle N \rangle ::= 'ct'$

| 'centro'

| 'br'

| 'borrar'

| 'nd'

| 'sd'

$\langle Co \rangle ::= 'color' \langle color \rangle$

$\langle nombre \rangle ::= 'tinto'$

| 'rojo'

| 'naranja'

| 'amarillo'

| 'verde'

| 'lima'

| 'turquesa'

| 'azul'

| 'morado'

| 'lila'

| 'rosa'

| 'cafe'

| 'negro'

| 'gris'

| 'blanco'

$\langle Re \rangle ::= 'repetir' \text{ int } '[' \langle S \rangle \langle Re2 \rangle ']'$

$\langle Re2 \rangle ::= \langle S \rangle \langle Re2 \rangle$

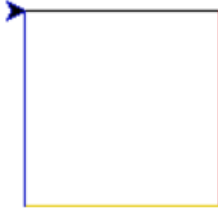
| ϵ

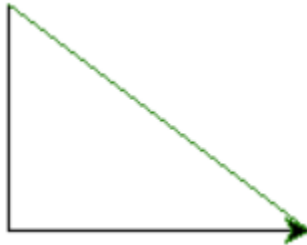
[4] Casos de prueba

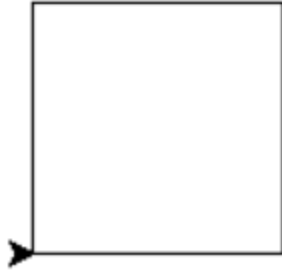
Las entradas (inputs) con las que se probó el programa, describiendo si funcionaron o no.

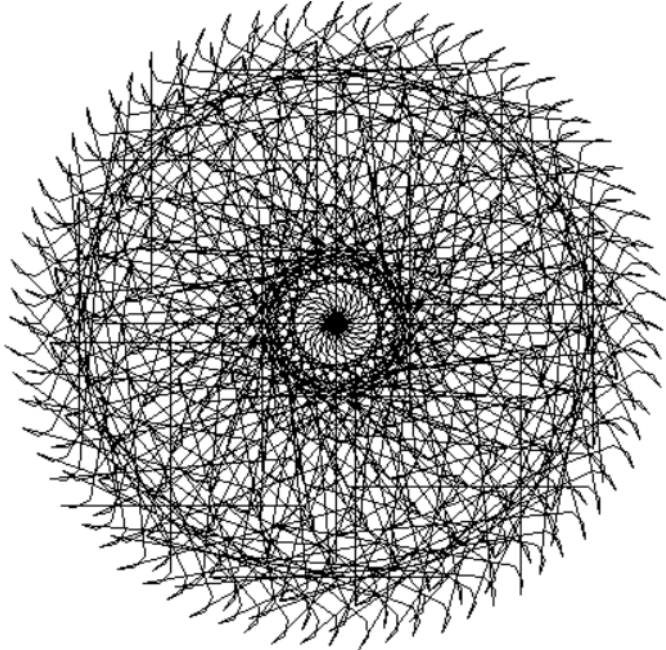
Input es el código LOBO del archivo de texto. La salida esperada dirá “Dibujo” si en el Output debería aparecer algo dibujado en el canvas, de lo contrario el error que debería aparecer estará en Salida esperada y el Output deberá mostrar los mismo.

El primer renglón de Output corresponde a la terminal del programa y la segunda al canvas.

[1]	Input	av 100 gd 90 color rojo av 100 gd 90 color amarillo av 100 gd 90 color azul av 100 gd 90
	Salida esperada	Dibujo: Cuadrado con cada lado de diferente color.
	Output	<div>El código se ha ejecutado con éxito Enter para cerrar</div> <div></div>

[2]	Input	rv 80 gi 90 av 60 color verde ct
	Salida esperada	Dibujo: Triángulo de hipotenusa verde.
	Output	<div>El código se ha ejecutado con éxito Enter para cerrar</div> <div></div>

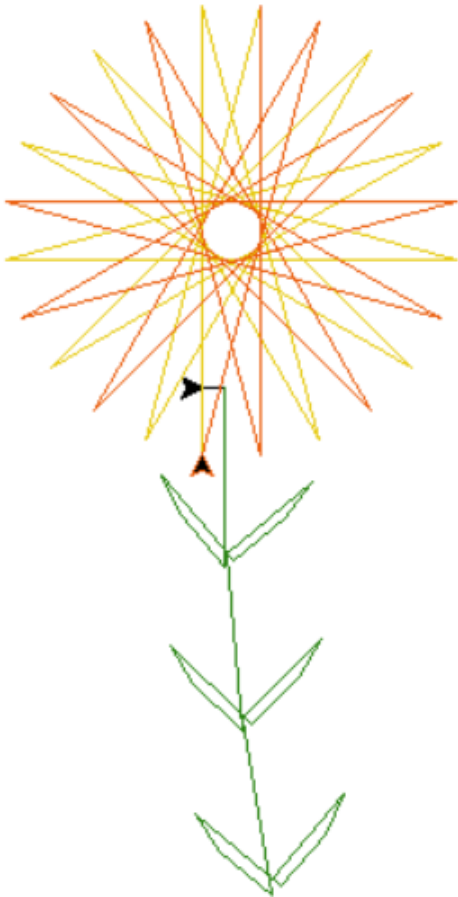
[3]	Input	repetir 4[av 100 gi 90]
	Salida esperada	Dibujo: Cuadrado.
	Output	<p>El código se ha ejecutado con éxito Enter para cerrar</p> 

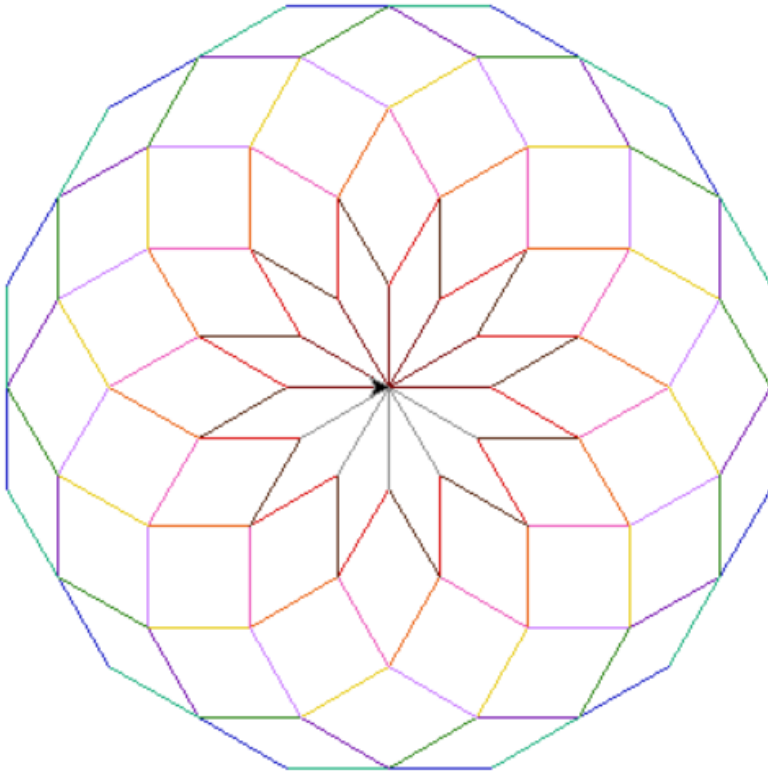
[4]	Input	repetir 24[gd 15 repetir 18[av 150 gd 45 av 10 gd 45 av 10 gd 170]]
	Salida esperada	Dibujo
	Output	<p>El código se ha ejecutado con éxito Enter para cerrar</p> 

[5]	Input	color rojo repetir 4[av 100 gd 90] br av 100
	Salida esperada	Dibujo: Cuadrado rojo (no será visible en output por cómo se ejecuta el dibujado) que después se borrará y hará una sola línea recta
	Output	El código se ha ejecutado con éxito Enter para cerrar

[6]	Input	color rosa repetir 5[av 20 nd av 20 sd]
	Salida esperada	Dibujo: Línea discontinua rosa.
	Output	El código se ha ejecutado con éxito Enter para cerrar

[7]	Input	repetir 360[av 1 gd 1]
	Salida esperada	Dibujo: Círculo.
	Output	El código se ha ejecutado con éxito Enter para cerrar

[8]	Input	<pre> av 10 gi 90 color verde repetir 3[rv 80 gi 40 av 30 gd 15 av 20 gd 165 av 50 gi 90 av 30 gi 15 av 20 gi 165 av 50 gd 125] nd ct gi 90 rv 30 sd color amarillo repetir 12[av 200 gd 165] color naranja repetir 12[av 200 gd 165] </pre>
	Salida esperada	<p>Dibujo: Girasol.</p>
	Output	<p>El código se ha ejecutado con éxito Enter para cerrar</p> 

[9] Input	<pre> repetir 12[gd 30 color tinto av 50 gd 30 color rojo av 50 gd 30 color naranja av 50 gd 30 color amarillo av 50 gd 30 color verde av 50 gd 30 color turquesa av 50 gd 30 color azul av 50 gd 30 color morado av 50 gd 30 color lila av 50 gd 30 color rosa av 50 gd 30 color cafe av 50 gd 30 color gris av 50 gd 30] </pre>
Salida esperada	<p>Dibujo: Figura de doce colores.</p>
Output	<p>El código se ha ejecutado con éxito Enter para cerrar</p> 

[1 0]	Input	av 100 gi av 200
	Salida esperada	ERROR: Se esperaba número después de 'gi' Código terminado
	Output	ERROR: Se esperaba número después de 'gi' Código terminado
		-

[1 1]	Input	900 av
	Salida esperada	ERROR: El código no pudo ejecutarse. Error en '900' Código terminado
	Output	ERROR: El código no pudo ejecutarse. Error en '900' Código terminado
		-

[1 2]	Input	av repetir 6 [av 11]
	Salida esperada	ERROR: Se esperaba número después de 'av' Código terminado
	Output	ERROR: Se esperaba número después de 'av' Código terminado
		-

[1 3]	Input	gd repetir 6 [av 12]
	Salida esperada	ERROR: Se esperaba número después de 'gd' Código terminado
	Output	ERROR: Se esperaba número después de 'gd' Código terminado
		-

[1 4]	Input	rv repetir 6 [av 13]
	Salida esperada	ERROR: Se esperaba número después de 'rv' Código terminado
	Output	ERROR: Se esperaba número después de 'rv' Código terminado -

[1 5]	Input	av 20 color azulChiclamino
	Salida esperada	ERROR: No se pudo interpretar 'azulchiclamino' Código terminado
	Output	ERROR: No se pudo interpretar 'azulchiclamino' Código terminado -

[1 6]	Input	av 20 color 215
	Salida esperada	ERROR: Se esperaba el nombre de un color después de 'color' (Ej.: 'tinto', 'rojo', 'naranja', 'amarillo', 'verde', 'lima', 'turquesa', 'azul', 'morado', 'lila', 'rosa', 'cafe', 'negro', 'gris', 'blanco') Código terminado
	Output	ERROR: Se esperaba el nombre de un color después de 'color' (Ej.: 'tinto', 'rojo', 'naranja', 'amarillo', 'verde', 'lima', 'turquesa', 'azul', 'morado', 'lila', 'rosa', 'cafe', 'negro', 'gris', 'blanco') Código terminado -

[1 7]	Input	repetir [av 100]
	Salida esperada	ERROR: Se esperaba número después de 'repetir' Código terminado
	Output	ERROR: Se esperaba número después de 'repetir' Código terminado
		-

[1 8]	Input	repetir 4 av 100]
	Salida esperada	ERROR: Se esperaba '[' Código terminado
	Output	ERROR: Se esperaba '[' Código terminado
		-

[1 9]	Input	repetir 4 [av 100
	Salida esperada	ERROR: Se esperaba ']' Código terminado
	Output	ERROR: Se esperaba ']' Código terminado
		-

[2 0]	Input	repetir 0[av 999]
	Salida esperada	ERROR: El ciclo repetir debe tener un valor igual o mayor a 1 Código terminado
	Output	ERROR: El ciclo repetir debe tener un valor igual o mayor a 1 Código terminado -

Conclusión: Todos los output corresponden a las salidas esperadas.

[5] Reflexión

A decir verdad, elegir el lenguaje en qué hacer el programa fue un inicio desastroso.

Primero traté de hacerlo en JavaScript con los mismos frameworks que utilizaba esta página <https://www.transum.org/software/Logo/> , pero, al no tener muchos conocimientos de desarrollo web, mi idea original fue un fracaso total. Después decidí empezar desde cero otra vez con Python, pero el verdadero reto fue el cómo relacionar los temas de clase con crear el nuevo lenguaje, aun teniendo una idea de en teoría cómo sería el léxico y sintaxis, programarlo fue un proceso doloroso, pero conforme iba programando y fallando, encontraba más sentido entre la práctica y los temas que vimos en clase, así todo se fue haciendo más fácil.

Ahora, con toda la teoría y experiencia de desarrollo de este nuevo lenguaje, puedo decir que comprendo muchísimo mejor los temas que hemos visto hasta ahora.