

## Parser recursivos

SVG (Scalable Vector Graphics) es un formato de imagen basado en XML que se utiliza para describir gráficos vectoriales bidimensionales. A diferencia de los formatos de imagen rasterizados como JPEG o PNG, SVG permite que las imágenes se escalen a cualquier tamaño sin perder calidad. Esto se debe a que las figuras en SVG se definen mediante coordenadas y formas geométricas, como líneas, círculos y rectángulos, en lugar de píxeles.

En el siguiente recuadro, se da una lista de tokens.

```
ABRE ::= <\w+(\s|[\w-]+=\"[\\w, :/.]+\" )*>
CIERRE ::= </\w+>
ABRECIERRA ::= <\w+(\s|[\w-]+=\"[\\w, :/.]+\" )*/>
ESPACIOS ::= \s+
COMENTARIO ::=
```

- 1) Complete el lexer, donde solamente se detecten tres tipos de etiquetas, las etiquetas de abrir, cierre y las que se cierran sobre sí mismas. Ignore las etiquetas de comentarios.

Es interesante guardar los atributos de las etiqueta en un diccionario, donde la clave es el nombre del atributo y el valor corresponde a la parte entre comillas, pero sin éstas.

- 2) Modifique el lexer, en caso de no haberlo hecho antes, para que el valor corresponda con el diccionario y añada un atributo con el nombre.

La gramática de SVG está dada por un par de reglas.

```
<Start> ::= <Svg> EOF
<Svg> ::= (ABRE <Svg> CIERRE )* | ABRECIERRA <Svg>
```

- 3) Implemente un parser recursivo descendente, utilizando las clases representadas en `representacion.py`.

Matplotlib es una librería para hacer gráficos, y su uso es muy intuitivo:

```
import matplotlib.pyplot as plt

# Crear la figura y los ejes
fig, ax = plt.subplots()

# Dibujar el triángulo
x_triangle = [1, 2, 3, 1]
y_triangle = [1, 3, 1, 1]
ax.fill(x_triangle, y_triangle, 'g', label='Triángulo') # 'g' es el color verde

# Dibujar el rectángulo
rectangle = plt.Rectangle((0.1, 0.1), 0.6, 0.3, color='b', label='Rectángulo') # Esquina inferior izqui
ax.add_patch(rectangle)

# Dibujar el círculo
circle = plt.Circle((0.5, 0.5), 0.4, color='r', label='Círculo') # Centro en (0.5, 0.5) y radio 0.4
ax.add_patch(circle)
# Configurar los ejes para que tengan la misma escala y mostrar
ax.set_aspect('equal')
plt.show()
```

- 4) *Añada un atributo `plt` y un método `dibuja` a la clase `SVG` que, al llamar a la función permita dibujar los diferentes objetos que se muestran en la clase.*