code_generator.py - El Traductor

¿Qué hace este archivo?

El generador de código es como un traductor especializado que toma la estructura de datos del parser y la convierte en código Python ejecutable, específicamente usando la librería Turtle para dibujar.

El Patrón Visitor

Usa el patrón "Visitor", donde para cada tipo de nodo del AST hay un método específico que sabe cómo convertirlo a código Python.

Mapeo de Comandos

El generador traduce comandos en español a funciones de Python Turtle:

- "avanzar" → `t.forward()`
- "girar" → `t.right()`
- "levantar_lapiz" → `t.penup()`
- "bajar_lapiz" → `t.pendown()`
- "color" → `t.color()`

Generación de Estructuras de Control

- **Condicionales**: "si... entonces... sino..." se convierte en "if... else..."
- **Bucles**: "mientras..." se convierte en "while...", "para..." en "for..."
- **Repeticiones**: "repetir n veces" se convierte en "for _ in range(n):"

Manejo de Indentación

El generador automáticamente maneja la indentación de Python, crucial para que el código funcione correctamente.

Cómo Trabajan Juntos los Tres Archivos

El Flujo Completo

1. **Entrada**: Pseudocódigo en español

```
repetir 4 veces
    avanzar 100
    girar 90
fin repetir
```

2. **Parser**: Convierte a estructura de datos

```
RepeatStatement(
    times=Literal(4),
```

```
        body=[
            Command("avanzar", [100]),
            Command("girar", [90])
        ]
    )
```

3. **Code Generator**: Produce código Python

```python
import turtle
t = turtle.Turtle()

for _ in range(4):
    t.forward(100)
    t.right(90)
```

Beneficios del Diseño

1. **Separación de responsabilidades**: Cada archivo tiene una función específica
2. **Extensibilidad**: Fácil agregar nuevos comandos o estructuras
3. **Mantenimiento**: Cambios en una parte no afectan las otras
4. **Reutilización**: El AST puede usarse para generar código en otros lenguajes

---

Casos de Uso Prácticos

Ejemplo 1: Dibujar un Cuadrado

```
repetir 4 veces
    avanzar 100
    girar 90
fin repetir
```

Ejemplo 2: Condicional

```
si x > 0 entonces
    avanzar x
sino
    girar 180
fin si
```

Ejemplo 3: Bucle con Variable

```
para i desde 1 hasta 5
    avanzar i * 10
    girar 72
fin para
```

Conclusión

Este parser demuestra cómo se puede crear un lenguaje de programación sencillo pero potente que permite a estudiantes y principiantes escribir programas de dibujo

de forma intuitiva. La arquitectura modular hace que el sistema sea robusto, mantenible y extensible.

El resultado es una herramienta educativa que combina conceptos de programación con resultados visuales inmediatos, haciendo el aprendizaje más atractivo y comprensible.