Parser de Pseudocódigo a Python/Turtle

Introducción

Se procederá a explicar un sistema que traduce pseudocódigo (código escrito en español de forma casi natural) a código Python que puede dibujar usando la librería Turtle. Es como tener un traductor que convierte instrucciones en español a instrucciones que la computadora puede entender y ejecutar.

Visión General del Sistema

El parser consta de tres componentes principales que trabajan en secuencia:

1. **ast_nodes.py** - Define la estructura de datos (como un diccionario de conceptos)
2. **parser.py** - Lee y analiza el pseudocódigo (como un intérprete)
3. **code_generator.py** - Genera el código Python final (como un traductor)

Se puede imaginar el proceso como una línea de producción: el pseudocódigo entra por un extremo y sale como código Python ejecutable por el otro.

Archivo parser.py - El Intérprete

¿Qué hace este archivo?

El parser es como un intérprete muy inteligente que lee pseudocódigo en español y lo convierte en la estructura de datos que definimos en ast_nodes.py. Lee palabra por palabra y entiende qué significa cada parte.

El Proceso de Análisis

1. **Recibe tokens**: Recibe una lista de "tokens" (palabras y símbolos identificados previamente)
2. **Identifica patrones**: Reconoce estructuras como "si… entonces… fin si"
3. **Construye el árbol**: Crea la estructura de datos correspondiente

Métodos Principales

- **parse()**: Método principal que inicia todo el proceso
- **parse_statement()**: Identifica qué tipo de instrucción está leyendo
- **parse_if_statement()**: Maneja las estructuras condicionales
- **parse_while_statement()**: Maneja los bucles "mientras"
- **parse_for_statement()**: Maneja los bucles "para"
- **parse_expression()**: Analiza expresiones matemáticas y lógicas

Ejemplo de Funcionamiento

Si el parser lee: "si x > 5 entonces avanzar 10 fin si"

1. Reconoce “si” → llama a `parse_if_statement()`
2. Analiza “x > 5” → crea una comparación (BinaryOp)
3. Procesa “avanzar 10” → crea un comando
4. Verifica “fin si” → completa la estructura condicional

Manejo de Errores

El parser incluye manejo de errores para reportar problemas como: - Tokens inesperados - Estructuras incompletas - Sintaxis incorrecta