Proyecto: Lexer para Fangless Python

Descripción del Proyecto

El presente proyecto consiste en el desarrollo de la primera fase de un compilador: el **analizador léxico (Lexer)**. El objetivo es el diseño e implementación de un componente que traduzca una versión simplificada del lenguaje Python, denominada "Fangless Python", en una secuencia de tokens.

El analizador léxico constituye la base fundamental para las fases subsecuentes del proceso de compilación, como el análisis sintáctico. Por lo tanto, su correcta implementación es crítica. Esta entrega se enfocará exclusivamente en el Lexer; el analizador sintáctico (Parser) será abordado en una entrega posterior.

Aspectos Administrativos y Técnicos

- **Herramienta de Desarrollo:** La herramienta designada para el desarrollo es **Python**, con el uso obligatorio de la librería **PLY (Python Lex-Yacc)**.
- **Idioma:** El código fuente, los comentarios internos y toda la documentación técnica deberán redactarse en **inglés**.
- **Control de Versiones:** Se requiere el uso del sistema de control de versiones **Git** en conjunto con la plataforma **GitHub**. La integración de cambios al repositorio principal deberá realizarse exclusivamente a través de *Pull Requests*.
- **Nomenclatura de Ramas:** Las ramas de desarrollo deben adherirse estrictamente al formato TASK_#_BriefDescription.
- **Equipos de Trabajo:** El proyecto será desarrollado en equipos de hasta **tres estudiantes**.

- **Fecha de Entrega:** La fecha límite para la entrega del proyecto es el **jueves 4 de setiembre de 2025, a las 23:59 hrs**.
- **Buenas Prácticas de Código:** Se espera que el código producido sea limpio, eficiente, modular y mantenible, facilitando así su futura extensión y revisión.
- El valor de este proyecto es de un 20% de la totalidad de la rúbrica proyectos.

Requerimientos del Lexer (Analizador Léxico)

Entrada y Salida

- **Entrada:** Un archivo de texto que contiene código fuente escrito en el subconjunto "Fangless Python".
- **Salida:** Una secuencia de tokens que representa los elementos léxicos identificados en el código fuente, tales como palabras clave, identificadores, literales y operadores.

Características Incluidas en la Sintaxis

El analizador léxico deberá ser capaz de identificar y tokenizar los siguientes elementos sintácticos.

1. **Palabras Clave (Keywords):**
    - **Control de flujo:** if, else, elif, while, for, break, continue, pass.
    - **Definiciones:** def, return, class.
    - **Tipos booleanos:** True, False.
    - **Operadores lógicos:** and, or, not.
2. **Identificadores:**

- Nombres asignados a variables, funciones y clases. Deben iniciar con una letra (a-z, A-Z) o un guion bajo (_), seguido de cualquier combinación de letras, dígitos (0-9) y guiones bajos.

3. **Literales:**
    - **Numéricos:** Secuencias de dígitos para enteros (ej., 123) y números con punto decimal para flotantes (ej., 123.45).
    - **Cadenas de Texto:** Delimitadas por comillas simples (') o dobles ("). Deben soportar secuencias de escape básicas (\n, \t, \\, \", \').
4. **Operadores:**
    - **Aritméticos:** +, -, *, /, //, %, **.
    - **Relacionales:** ==, !=, <, >, <=, >=.
    - **Asignación:** =, +=, -=, *=, /=, %=, //=, **=.
5. **Delimitadores y Símbolos:**
    - ( ) [ ] { } : , ..
6. **Comentarios:**
    - Comentarios de una sola línea, que inician con # y se extienden hasta el final de la línea, deben ser ignorados por el analizador.
7. **Indentación:**
    - Se deben procesar los espacios en blanco y tabulaciones al inicio de las líneas para definir bloques de código. El Lexer debe generar tokens especiales INDENT y DENT para señalar cambios en el nivel de indentación.

Manejo de Errores Léxicos

El analizador léxico debe gestionar los errores de forma robusta, sin interrumpir su ejecución abruptamente.

- **Caracteres Desconocidos:** Cualquier carácter que no corresponda a un token definido debe generar un error léxico.
- **Secuencias de Escape Inválidas:** El uso de secuencias de escape no reconocidas dentro de literales de cadena debe ser reportado como un error.
- **Indentación Incorrecta:** Deben manejarse y reportarse los errores en la estructura de indentación del código.

Rúbrica de Evaluación

Criterio	Puntos	Descripción
Completitud del Lexer	40	Identificación correcta de todos los tokens especificados en el subconjunto del lenguaje, incluyendo una gestión precisa de la indentación.
Calidad del Código	30	El código debe ser bien estructurado, modular y estar debidamente documentado, demostrando una implementación eficiente.
Manejo de Errores	15	Capacidad del sistema para detectar y reportar errores léxicos y sintácticos con mensajes claros y precisos que faciliten la depuración.
Documentación	15	Calidad y claridad de la documentación técnica (comentarios) y de usuario (guía de uso), explicando los componentes y su funcionamiento.