Instituto Tecnológico de Costa Rica

Escuela de Computación

Ingeniería en computación

Compiladores e Intérpretes

Grupo 2

Profesor: Ericka Marín

Análisis léxico

Adrián López Quesada, 2014081634

Josué Arrieta Salas, 2014008153

Cartago

Lunes 18 de abril

**Índice**

[Análisis de resultados 3](#_Toc448690657)

[Casos de pruebas 4](#_Toc448690658)

[Prueba 1 : Tokens de operadores 4](#_Toc448690659)

[Prueba 2 : Tokens de palabras reservadas 5](#_Toc448690660)

[Prueba 3 : Comentarios de línea y de bloque 6](#_Toc448690661)

[Prueba 4: Código exhaustivo 7](#_Toc448690662)

[Prueba 5: Strings y Chars 8](#_Toc448690663)

[Prueba 6: Números y flotantes 9](#_Toc448690664)

[Prueba numero 7: identificadores 10](#_Toc448690665)

[¿Cómo compilar y correr el Scanner? 12](#_Toc448690666)

# Análisis de resultados

Se toman en cuenta las reglas de Python para la realización de este análisis de resultados.

|  |  |
| --- | --- |
| **Objetivo** | **Porcentaje de éxito (100%)** |
| Programa recibe código fuente escrito en Python y analiza el archivo. | 100% |
| Se listan errores léxicos encontrados (por línea y el error). | 100% |
| El programa ante un error léxico ser recupera de este y no despliega errores en cascada. | 100% |
| Se listan los tokens encontrados (por tipo, línea del código fuente donde se presentan y cantidad de ocurrencias de cada token en cada línea). | 100% |
| La lista de tokens encontrados es ordenada en orden alfabético | 100% |
| Tokens de operadores son procesados correctamente. | 100% |
| Tokens de literales (strings, enteros, flotantes, char) son procesados correctamente. | 100% |
| Tokens de identificadores son procesados correctamente. | 100% |
| Tokens de palabras reservadas son procesados correctamente. | 100% |
| El programa identifica comentarios (de bloque o de línea) y omite los tokens dentro de ellos. | 100%: se ha de mencionar que si un comentario de bloque no se cierra, y se alcanza el EOF deberá notificar error. |
| Números aceptados en cualquier formato de Python (binario, hexadecimal y octal). | 100%: este objetivo es importante ya que se estará realizando un scanner lo más parecido posible al lenguaje Python real. |
| Se lograron características en un buen scanner. | 100%: este objetivo es lo que diferencia un buen scanner de uno malo. Casos como mañana, 999hola o ho+as son tomados como incorrectos. |
|  |  |
|  |  |

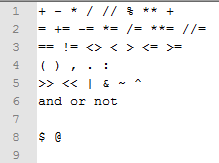
# Casos de pruebas

## Prueba 1 : Tokens de operadores

Objetivo: cada operador permitido en el lenguaje es analizado. Se agregan operadores que no existen para verificar errores.

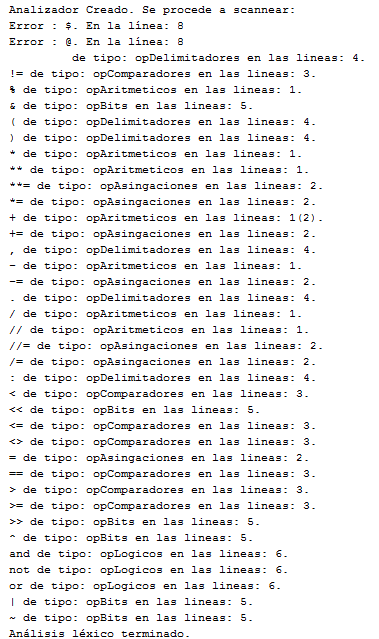
Resultados esperados: se espera la lista de errores para los operadores incorrectos y el listado de tokens de operadores correctos (agrupados por familia de operadores). Se ha de mencionar que anteriormente se tenía un Token por cada operador, sin embargo ahora es por familia de operadores. Esta es la primera prueba

Entrada: En el archivo Prueba.mypy se tiene el siguiente texto:



* En la línea 8 se agregan operadores no existentes. Deben ser errores.
* En la línea 1 se agregan 2 ‘+’ que deben Salir repetidos.

Resultados Obtenidos (correctos):

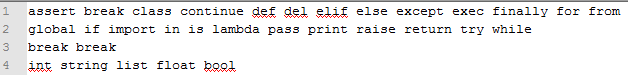


## Prueba 2 : Tokens de palabras reservadas

Objetivo: cada palabra reservada en el lenguaje es analizado.

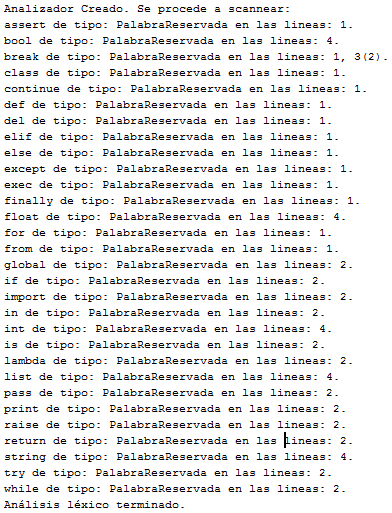
Resultados esperados: se espera el listado de tokens de operadores correctos Se ha de mencionar que anteriormente se tenía un Token por cada palabra reservada, sin embargo ahora se agrupan en un solo token. Esta es la primera prueba.

Entrada: En el archivo Prueba.mypy se tiene el siguiente texto:



En la línea 3 se puso 2 palabras reservadas repetidas.

Resultados obtenidos (correctos):

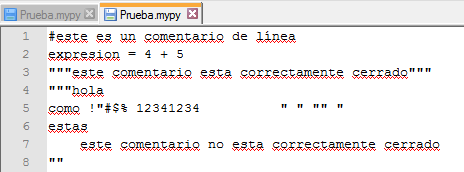


## Prueba 3 : Comentarios de línea y de bloque

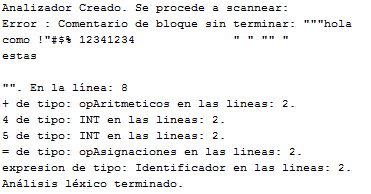
Objetivo: comentarios son prbados..

Resultados esperados: Se deben ignorar. Si un comentario de bloque no se cierra y se llega al final del archivo, es considerado un error. Se deben listar los tokens y errores correctamente. Esta es la primera prueba.

Entrada: En el archivo Prueba.mypy se tiene el siguiente texto:



Resultados obtenidos (correctos):



Los comentarios correctos son ignorados.

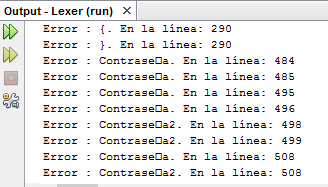
## Prueba 4: Código exhaustivo

Objetivo: probar el lexer en un código verdadero

Se prueba código de 592 líneas, trozo relevante:

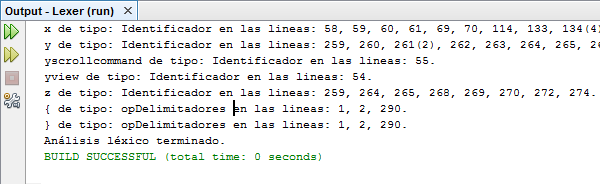


Resultados obtenidos (Incorrectos):



Los errores en la línea 290 no deberían dar error, los demás son problemas de ñ’s y todo lo demás sale correcto.

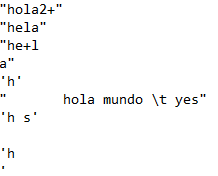
Resultados obtenidos tras la correcion:



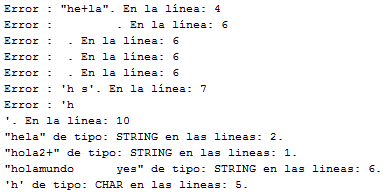
## Prueba 5: Strings y Chars

Objetivo: probar errores y reconocimiento de strings y chars

Se prueba código de 592 líneas, trozo relevante:

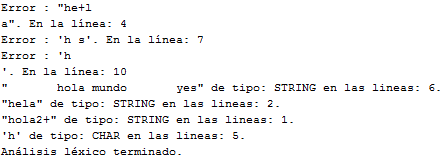


Resultados obtenidos (Incorrectos):



No reconce tabs normales y espacios en strings, los errores de la línea 6 no deberían aparecer, los otros si tienen sentido ya que están malos. Acepto todos los otros strings y chars correctamente.

Resultados obtenidos tras la corrección:



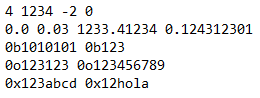
Todo caso del string y char es cubierto.

## Prueba 6: Números y flotantes

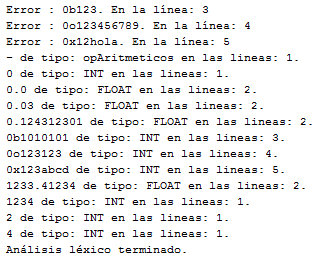
Objetivo: se prueba todos los posibles literales de números permitidos (flotantes y enteros). También se agregan números no posibles para probar errores.

Resultados esperados: se espera la lista de errores para los operadores incorrectos y el listado de tokens de operadores correctos. Se espera algunos resultados incorrectos.

Entrada: En el archivo Prueba.mypy se tiene el siguiente texto:



Resultados obtenidos (correctos):

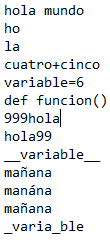


## Prueba numero 7: identificadores

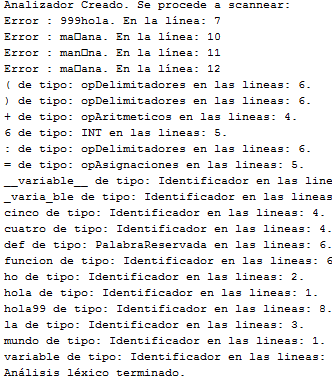
Objetivo: se prueban todos los posibles identificadores permitidos, además de errores de identificadores.

Resultados esperados: se espera la lista de errores con todo el token incorrecto, además de un listado de que incluya los identificadores correctos.

Entrada: En el archivo Prueba.mypy se tiene el siguiente texto:



Resultados obtenidos (correctos):



Se puede apreciar como contiene los errores de identificadores completos y no separa el 999 como int del hola como identificador. Pasa lo mismo con caracteres inválidos en los nombres de identificadores, pero acepta todo identificador que empieza en carácter o “\_” y cualquier identificador que tenga caracteres válidos, números y “\_” en su interior.

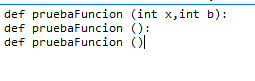
# 

# ¿Cómo compilar y correr el Scanner?

La parte esencial para compilar el proyecto es agregar el archivo “jflex-1.6.1.jar” como JAR a las librerías que tiene acceso. De esta manera se pueden utilizar todas las funcionalidades del Jflex. En caso de duda, por favor seguir las instrucciones del video siguiente: https://www.youtube.com/watch?v=w-KfjJdRas8 cerca del minuto 10, tomando en cuenta que se importa como JAR, no como librería. Otra manera es cargar el proyecto a netbeans el cual ya esta ya tiene el JAR agregado. Además, se necesita que el archivo flex.lexer esté presente en el src y aunque no es necesario para compilar se recomienda utilizar la ubicación actual del archivo de pruebas.

Aunque no es de relevancia, el proyecto se desarrolló bajo el IDE de Netbeans. Para correr el scanner se puede utilizar el .jar “Scanner ejecutable” para hacer las pruebas con el archivo “Pruebas.MYPY” que se encuentra dentro de la carpeta ejecutable. Se debe modificar el nombre del archivo por motivos de envio por correo, solo se le debe quitar la “a” al final del nombre de la extensión .jar

## Prueba 1 : funciones



Resultado:



Ya que no se habían definido “x” cantidad de funciones

Al definirlo para “x” cantidad de funciones:



Porque falta el ultimo “:”

Al arreglar ese error en la prueba:

