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 (Josué) 3](#_Toc453067753)

[Casos de pruebas 4](#_Toc453067754)

[Prueba 1 : Funciones (Adrián) 4](#_Toc453067755)

[Prueba 2 : Expresiones (Josué) 5](#_Toc453067756)

[Prueba 2.1 – Primeras pruebas con Expresiones 5](#_Toc453067757)

[Prueba 2.1 – Segundas pruebas con Expresiones 6](#_Toc453067758)

[Prueba 3 : Declaración de variables (Josué) 7](#_Toc453067759)

[Prueba 4: Asignaciones (josué) 8](#_Toc453067760)

[Prueba 5: Print e Input (Josué) 8](#_Toc453067761)

[Prueba 6: While (Adrián) 9](#_Toc453067762)

[Prueba numero 7: If (Adrián) 9](#_Toc453067763)

[Prueba numero 8: For (Adrián) 9](#_Toc453067764)

[Prueba numero 9: Try (Josué) 10](#_Toc453067765)

[Prueba numero 10: Break - Continue (Adrián) 10](#_Toc453067766)

[Prueba numero 11: OOP (Josué) 11](#_Toc453067767)

[Prueba 11.1 – Estructura Básica 11](#_Toc453067768)

[Prueba 11.2 – El ; 12](#_Toc453067769)

[¿Cómo compilar y correr el Scanner? (Adrián) 13](#_Toc453067770)

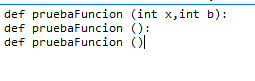
# Análisis de resultados (Josué)

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 : Funciones (Adrián)



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:



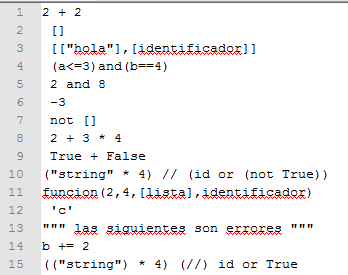
## Prueba 2 : Expresiones (Josué)

### Prueba 2.1 – Primeras pruebas con Expresiones

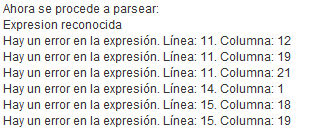
Esta prueba fue realizada en el transcurso de la realización del proyecto, y es considerada precisamente por esa razón. En el momento que se realizó las pruebas, una asignación no era considerada una expresión (ahora sí lo es) para que se tome en cuenta.

Objetivo: probar todo tipo de expresiones en el lenguaje MyPython.

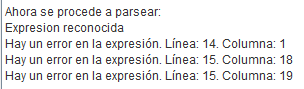
Se tiene el siguiente documento de texto de entrada:



Se espera errores en la línea 14 y en la línea 15, sin embargo las demás son expresiones válidas. Se tienen los siguientes resultados (incorrectos):

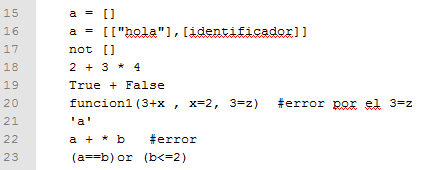


Se muestra que hay un error en la línea 11 debido a que las funciones no se contemplaron como expresiones. Se aplica un arreglo y al correr el mismo archivo se tienen los siguientes resultados (correctos):

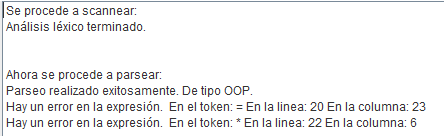


### Prueba 2.1 – Segundas pruebas con Expresiones

Objetivo: realizar aún más tipos de prueba con expresiones. En este caso las asignaciones si son tomadas en cuenta como expresiones. Se tiene el siguiente archivo de entrada:

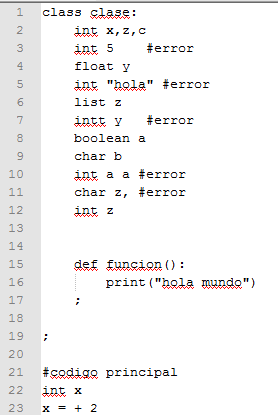


Resultados esperados: Se espera que haya errores en las líneas 22 y 20. Se obtuvieron los siguientes resultados (correctos):

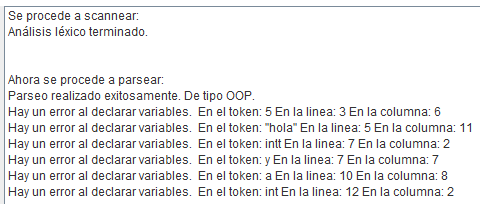


## Prueba 3 : Declaración de variables (Josué)

Objetivo: probar todo tipo de declaración de variables en el lenguaje MyPython. Se tiene el siguiente código de entrada:



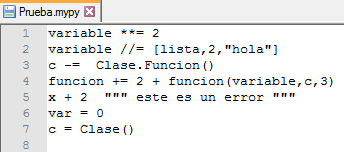
Resultados esperados: Se espera error en las líneas 3, 5 , 7 , 10 y 11. Se obtuvieron los siguientes resultados (correctos):



El error de la línea 11 se muestra en la línea 12 debido a la naturaleza del error y el error surge al encontrar “int” en la línea 12 y no un identificador. Este mensaje es considerado correcto.

## Prueba 4: Asignaciones (josué)

Objetivo: probar diferentes tipos de asignaciones posibles en el lenguaje MyPython. Se tiene el siguiente código de entrada:

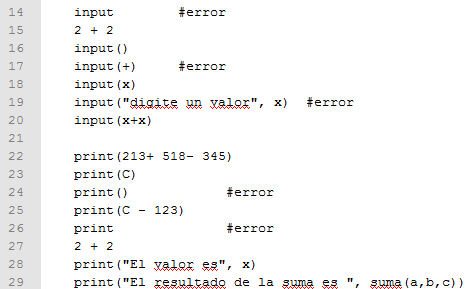


Resultados esperados: Se espera un error en la línea 5 ya que “x + 2” no es una asignación sino una expresión, sin embargo se espera que el compilador se recupere de este error. Se obtuvieron los siguientes resultados (correctos):

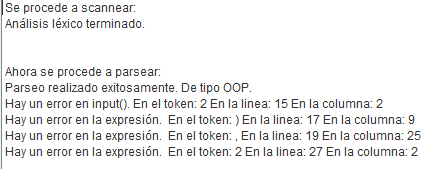


## Prueba 5: Print e Input (Josué)

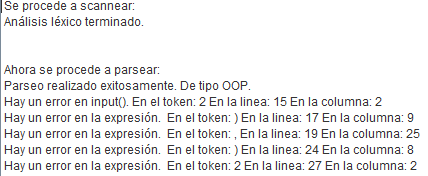
Objetivo: probar la estructura de las funciones print() e input() predefinidas. Se tiene el siguiente código de entrada:



Se espera que haya errores en las líneas 14, 17, 19,24 y 26. También se espera que se recupere del error y siga parseando. Se obtuvieron los siguientes resultados (incorrectos):



Como se ha mencionado, algunos errores los muestra en la línea después (en el caso del erro de la línea 14 y 26). Sin embargo el error de la línea 24 no se mostró. Print() es considerado correcto. Se aplica una corrección y se vuelve a correr el mismo archivo, y se obtienen los siguientes resultados (correctos):



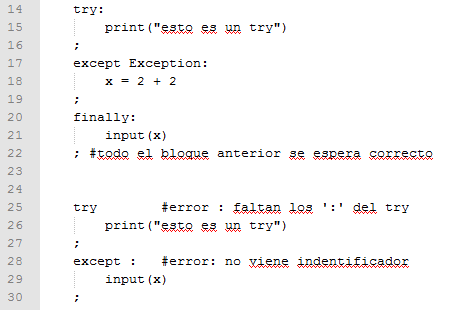
## Prueba 6: While (Adrián)

## Prueba numero 7: If (Adrián)

## Prueba numero 8: For (Adrián)

## Prueba numero 9: Try (Josué)

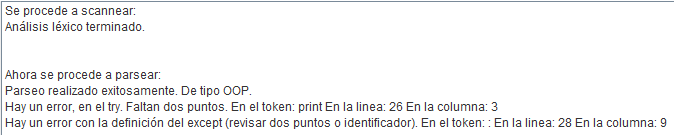
Objetivo: probar la estructura para el control de errores Try-Except en MyPython. Se tiene el siguiente archivo de entrada:



Resultados esperados:

* Error en la línea 25 ya que faltan los ‘:’ del try. Debe tirarlo en la línea 26 ya que ahí está el token que posee el error.
* Error en la línea 28 ya que falta un identificador en el except.

Resultados obtenidos (correctos):

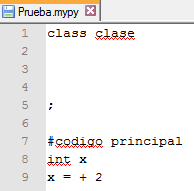


## Prueba numero 10: Break - Continue (Adrián)

## Prueba numero 11: OOP (Josué)

### Prueba 11.1 – Estructura Básica

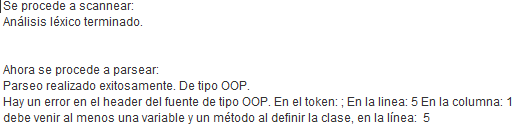
Objetivo: probar la estructura básica de la clase de un archivo MyPython. Se tiene el siguiente código de entrada:



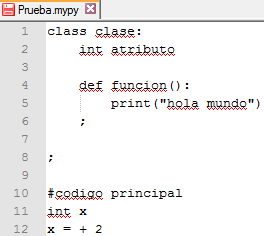
Resultados esperados: se espera que surjan los errores:

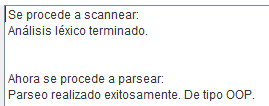
* Faltan los ‘:’ en el header de la clase.
* Se queje de que no vengan variables ni métodos (tienen que venir al menos 1):

Resultados obtenidos (correctos):



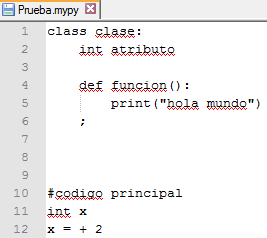
Si se añade el siguiente archivo (con los errores corregidos) y no hay errores sintácticos:





### Prueba 11.2 – El ;

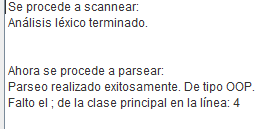
Objetivo: probar cuando no viene el ; al definir una clase. Se tiene el siguiente archivo:



Resultados esperados: error sintáctico al no cerrar la clase con el ;. Resultados obtenidos (incorrectos) :



Se aplica un parche que arregle el error y al correr el mismo archivo se obtiene (resultados correctos):



# ¿Cómo compilar y correr el Scanner? (Adrián)

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