**Escuela Politécnica del Litoral**

**Proyecto de Lenguaje de Programación - Dart**

**Primer Avance de Proyecto: Analizando Sintaxis**

**Integrantes:**

* Rogwi Cajas
* Isaac Solis

PAO II 2020

# INTRODUCCIÓN

Dart es un lenguaje de código abierto que fue desarrollado por Google para permitir a los desarrolladores utilizar un lenguaje de programación orientado a objetos y con un análisis estático de tipo. Su primera versión estable fue en el año 2011, pero ha cambiado mucho desde entonces, tanto el lenguaje en si como su objetivo principal, cambiando el análisis estático por uno opcional y agregando Flutter que cambio totalmente el objetivo de dicho lenguaje.

Dart fue desinado para hacer el proceso de desarrollo mucho más cómodo, rápido para los programadores, por ello viene con un set bastante extenso de herramientas integradas, como su propio gestor de paquetes, compiladores, un analizador y formateador, entre otras cosas. Dart también proporciona una máquina virtual que ayuda a presentar los cambios realizados en el código ejecutándolo inmediatamente. También permite que cuando el código esta producción se pueda compilar en lenguaje nativo, por lo que no es necesario un entorno especial para su ejecución, es decir si se quiere hacer desarrollo web Dart traspilla a JavaScript.

Por otro lado, la sintaxis de Dart es muy similar a otros lenguajes de programación como Java, C++ por que aprenderlo es sumamente sencillo por su gran similitud.

# OBJETIVO GENERAL

Construir un analizador léxico para un lenguaje de programación específico, en este caso será Dart, con la finalidad de ayudar a los personas que quiera aprender este lenguaje a identificar las principales funciones de manejo de datos soportados.

# OBJETIVOS ESPECÍFICOS

* Aplicar expresiones regulares para reconocer la sintaxis léxica de los componentes del lenguaje de programación Dart.
* Validar la declaración de variables y tipos de datos, estructuras de control soportados por Dart.
* Establecer tokens que permitan reconocer las instrucciones establecidas.
* Validar la declaración de operaciones matemáticas, lógicas y de comparación.

# ALCANCE

Para la viabilidad del proyecto, será analizará la declaración de variables y su inicialización, así como los tipos de datos que soporta el lenguaje de programación. También se analizará los operadores aritméticos y lógicos para realizar operaciones, al igual que operadores de comparación. También se revisará las estructuras de control más importantes como las iterativas y condicionales. Finalmente, también se revisara funciones para la entrada y salida de texto y otras para crear y manejar arreglos y conjuntos de elementos.

# TIPOS DE DATO

|  |  |  |
| --- | --- | --- |
| **TIPO DE DATO DESCRIPCIÓN EJEMPLO** | | |
| **Entero** | Tipo de dato numérico | var entero= 99;  Int valor =90;  Double flotante = 90.99; |
| **Booleano** | Tipo de dato booleano, puede ser true o false de acuerdo con la condición de que obedece | var verdad = true;  bool valor = true; |
| **String** | Cadena de caracteres | var valor = “12345abcd??”;  var valor = ‘hola’; |
| **Listas** | Objeto de alcance global, destinado a la construcción de matrices | var flybyObjects = ['Jupiter', 'Saturn', 'Uranus', 'Neptune']; |
| **Date** | Representación de fechas, siguen el orden de año/mes/día | var berlinWallFell = new DateTime.utc(1989, 11, 9); |
| **Conjunto** | Colección ordenada de elementos únicos | Var rangos = {“Gold”, “Silver”, “Platinum”} |

* 1. **OPERADORES ARITMÉTICOS**

|  |  |
| --- | --- |
| OPERADORES DESCRIPCIÓN | |
| + | Adición |
| - | Sustracción |
| \* | Multiplicación |
| \*\* | Exponenciación |
| / | División |
| **%** | Modulo |

* 1. **OPERADORES LÓGICOS:**

|  |  |
| --- | --- |
| OPERADORES DESCRIPCIÓN | |
| && | AND |
| || | OR |
| ! | NEGACIÓN |

* 1. **OPERADORES DE COMPARACIÓN**

|  |  |
| --- | --- |
| OPERADOR DESCRIPCIÓN | |
| == | Es igual |
| != | No es igual |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual |
| >= | Menor o igual |

* 1. **IMPRESIÓN DE UN STRING**

|  |  |  |
| --- | --- | --- |
| print(...); | Función común para imprimir mensajes en la consola, es necesario importar la librería con import ‘dart:io’; | print(“hello world”); |

* 1. **ENTRADA DE UN STRING (INPUT)**

|  |  |  |
| --- | --- | --- |
| Stdin.readLineSync(); | Permite al usuario ingresar texto, no permite argumentos a manera de texto descriptivo | Var entrada = stdin.readLineSync(); |

* 1. **FUNCIONES CON ARREGLOS**

|  |  |  |  |
| --- | --- | --- | --- |
| length() | Devuelve el número total de elementos en el arreglo | array.length(); | var arr = [1,2,3,4]  Var a= arr.length()  //a contiene a 4 |
| first() | Retorna el primer elemento del arreglo | arr.first() | Var arr = [1,2,5,4,3]  Var a= arr.first()  //a es igual a 1 |
| last() | Retorna el último elemento del arreglo | array.last() | Var arr = [‘a’,’b’,’c’,’d’,’e’]  Var a= arr.last();  //a es igual a ‘e’ |
| indexacion | Retorna el elemento del arreglo que indica el indice | Arr[a] | Var arr = [1,2,5,4,3] ;  Var a = Arr[2];  // a es igual a 5 |
| sublist(a,b) | Retorna una lista con los elementos dentro del rango seleccionado | Arrr,slice(a,b) | Var arr = [1,2,5,4,3] ;  Var a = Arr. sublist(0,3);  // a será igual a [1,2,5,4] |

* 1. **FUNCIONES CON CONJUNTOS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Add(...) | Agrega un nuevo elemento al conjunto, no lo añadirá si este ya existe. | | | | var numeros = {1,2,3,4}  Numeros.add(5);  // {1,2,3,4,5} |
| length | Retorna el número de ítems del conjunto | | | | var numeros = {1,2,3,4}  Numeros.length;  // 4 |
| Remove(item) | Elimina el item enviado como parametro |  |  |  | var numeros = {1,2,3,4}  Numeros.remove(4);  // {1,2,3} |

* 1. **CREACION DE UNA FUNCION**

Las funciones en Dart sirven para agrupar una tarea que será recurrente en otras partes del codigo, siempre debe indicar que tipo de dato retorna, “void” en caso de que no retorne ningún valor. Se definen de la siguiente manera.

type nombreFuncion(param1,…,paramn) {

//Declaracion de la funcion

Return typeValue;

}

//llamada a la funcion

var result = nombreFuncion(...);

* 1. **ESTRUCTURAS DE CONTROL BASICAS**

|  |  |  |
| --- | --- | --- |
| **FOR** | for (variable in objeto) {……} | for (var object in flybyObjects) {  print(object);  } |
| **WHILE** | while(condition) {} | while (year < 2016) {  year += 1;  } |
| **IF-ELSE IF-ELSE** | If (condition) {…} else if(condition){….}  else{…...} | if (age >= 60) {  print('tercera edad');  } else if (year>=18) {  Print('adulto joven');  } else{  Print(‘menor de edad’);  } |

# 5. Resultados

Después de haber desarrollado el componente léxico y sintáctico para este proyecto con el fin de reconocer las reglas y sentencias mas comunes del lenguaje de programación Dart se hicieron varias pruebas para ver si el proyecto reconoce correctamente todas las reglas tanto a nivel sintáctico, como al nivel léxico.

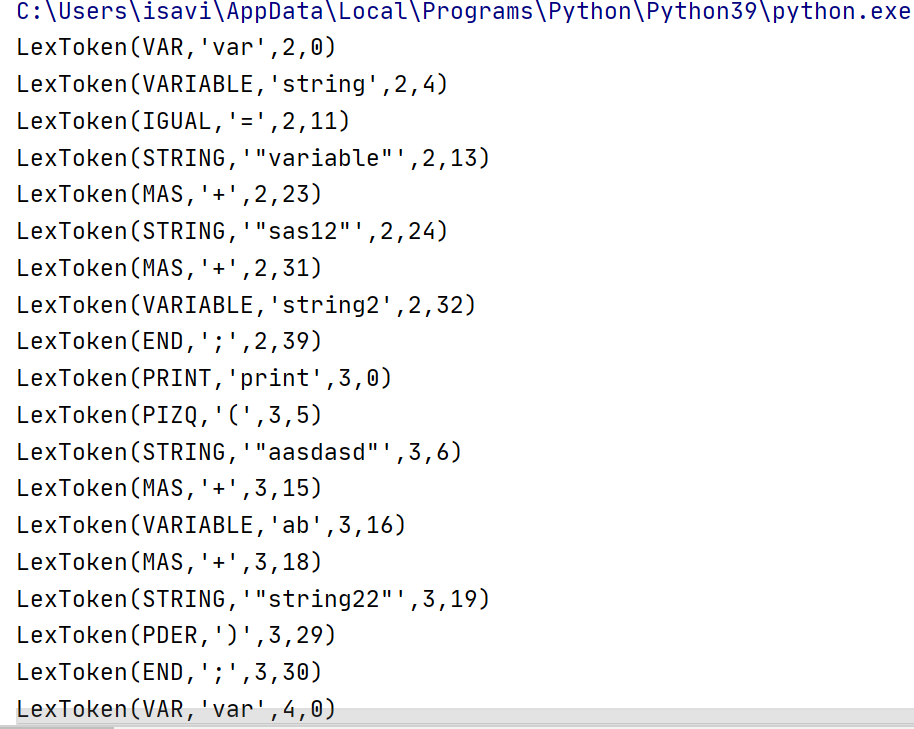
Primero se revisará los resultados del análisis léxicos de los dos archivos de prueba, mostrando cuando los resultados sin error y cuando si lo hay, revisando los tokens que se destacaron y los errores que existieron.

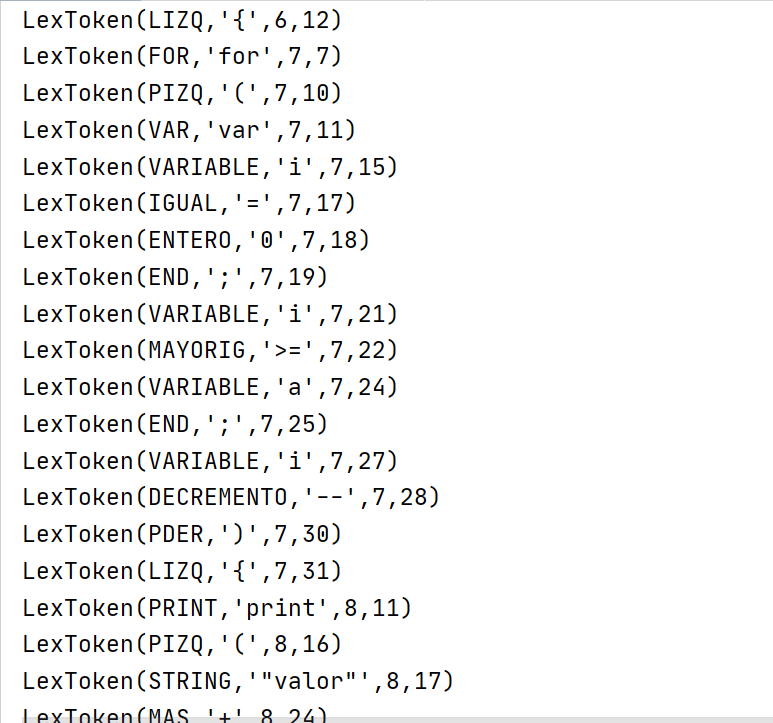
/\*comentario\*/  
var string = "variable"+"sas12"+string2;  
print("aasdasd"+ab+"string22");  
var a = 3+4/23;  
while(a&&(a>c)){  
 if(a>50){  
 for(var i =0; i>=a; i--){  
 print("valor"+"asas");   
 }  
 }else if(a>25){  
 a=20;  
 }else{  
 print("valor muy bajo");  
 }

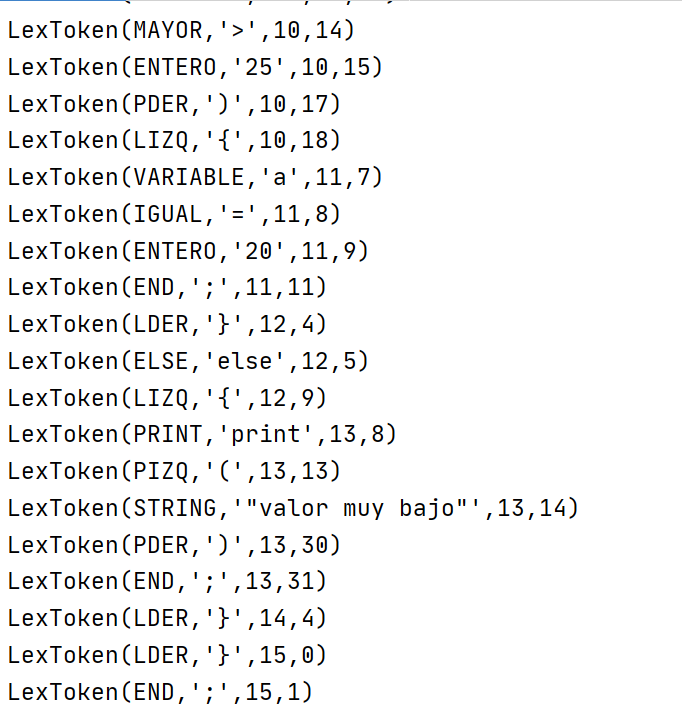
}

Descripción: Código del archivo de prueba CodigoCajas.txt

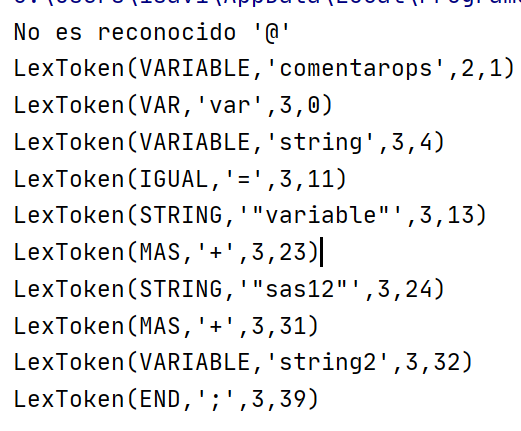
Se mostrará el resultado luego de pasar el código por el analizador léxico de Dart, el cual arrojo el siguiente resultado:







El analizador léxico devolvió todos lo tokens que forman parte del léxico del lenguaje de programación Dart. A continuación, se mostrará resultados cuando se le agrega código con errores, o con una palabra fuera del léxico.

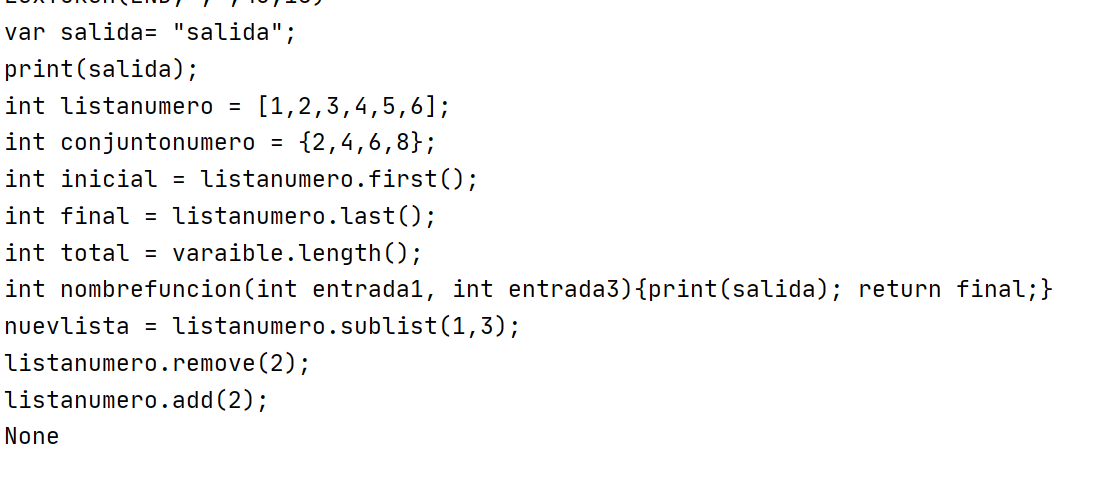


Después de revisar lo resultados del analizar léxico, vamos a revisar el analizador sintáctico el cual se encarga de controlar que los tokens que se envían formen parte de una regla ordenada que es admitida por el lenguaje Dart, se mostrara el código de prueba para revisar los resultados que devolvió.

var salida= "sali";  
print(salida);  
int listanumero = [1,2,3,4,5,6];  
int conjuntonumero = {2,4,6,8};  
int inicial = listanumero.first();  
int final = listanumero.last();  
int total = varaible.length();  
int nombrefuncion(int entrada1, int entrada3){print(salida); return final;}  
nuevlista = listanumero.sublist(1,3);  
listanumero.remove(2);  
listanumero.add(2);

Descripción: Código del archivo de prueba CodigoCajas.txt

El resultado del código al probar el analizador sintáctico Dart es el siguiente:



# 6 . Conclusiones

* Para realizar la interfaz grafica del proyecto se utilizó la librería Tkinter, la cual presenta componentes y funciones que ayudaran a desarrollar un GUI que permita la entrada de texto y analizar tanto léxicamente como sintácticamente la entrada, mostrando un cuadro de texto con los resultados correspondientes.
* Para el analizador léxico de este proyecto se usó la librería de Python llamada PLY.LEX, esta fue usada para romper cadenas de textos de entrada y compararlas con tokens específicos que son parte de una colección de reglas formadas por expresiones regulares. Con estas herramientas se pudo identificar las palabras reservadas, los caracteres y palabras que forman parte del lenguaje de programación Dart.
* Para desarrollar el analizador sintáctico se utilizó la librería de Python llamada PLY.Yacc, la cual ayudo a reconocer la sintaxis de un lenguaje de programación que se haya especificado, en este caso mediante las reglas de Dart. Es decir, analiza que la entrada que se envía cumple con las reglas previstas del análisis sintáctico sin contexto, revisando que las palabras formen parte de los tokens con ply.lex y que se encuentren en el orden de las reglas que se definieron.
* Dart es un lenguaje de programación orientado a objetos y con un análisis estático de tipo, el cual lo hace muy estricto en su sintaxis, al manejar diferentes de tipo de datos, para crear y asignar variable y estructuras de datos. Es muy similar a Java, lo que facilito comprender su sintaxis, palabras reservadas y las reglas que se debieron tomar en cuenta para desarrollar este proyecto.
* Finalmente el proyecto de analizador léxico y sintáctico del lenguaje Dart logro cubrir todo el alcance que se propuso, este reconoce las palabras reservada de Dart, funciones de entrada y salida de texto, operadores aritméticos, lógicos y de comparación, algunas estructuras de datos como Listas y Conjuntos, funciones de manejo de las estructuras, operaciones, declaración de variables, asignaciones y declaración de funciones con parámetros de entrada y salida. Se lo