Informe 01

Tema: Anañizador Léxico

Estudiante	Escuela	Asignatura
Lucía Luna Alencastre llunaa@ulasalle.edu.pe	Carrera Profesional de Ingeniería de Software	Compiladores Semestre: V

1. Introducción

1.1. Justificación

- El motivo por el cual se quiere desarrollar este lenguaje es debido a que en muchas ocasiones surgen errores al momento de compilar en python debido a la identacion.

1.2. Objetivos

- Se desea hacer un lenguaje similar a Python, llamado "LFDSMDFR", el cual se usen llaves para una mejor organizacion de las funciones, bucles y condicionales, ademas de ajustar algunas de la palabras reservadas para que sea más amigable.

2. Propuesta

2.1. Especificación Léxica

2.1.1. Definición de los comentarios

- Se define un comentario utilizando "//" y de finaliza con el mismo.

2.1.2. Definición de los identificadores

- Para los identificadores se puede iniciar con cualquier letra minúscula o mayúscula, sin espacios, puede contener números y guión bajo pero no puede empezar con estos.

2.1.3. Definición de las palabras clave

- Existen las siguientes palabras claves, def para definir una función, int para enteros, float para flotantes, Fifty-Fifty para booleanos, String para string, meanwhile para el bucle while, repeat para el bucle for, show para imprimir, if y else para condicionales, select para el controlador de selección.

2.1.4. Definición de los literales

- Para utilizar literales se usa " " "al inicio y al final del contenido, estos pueden contener cualquier carácter, número o carácter especial incluidos espacios. Para un solo carácter se usa " ' " al inicio y al final.

2.1.5. Definición de los operadores

- Para el operador igual se utiliza "=" entre dos identificadores, un identificador y un número o ambos combinados, para las comparaciones se utilizan "==" para igualdad, "=!" para diferente de, ">=" para mayor igual, "<=" para menor igual, ">"para mayor, "<"para menor, los paréntesis "()", las llaves"", suma"+", resta "-", multiplicación "*", división "/"

3. Expresiones regulares

Token	exprecion regular
identificador	$[a-z]([a-Z0-9]\|_)*$
numeral	[0-9]+
oper_plus	+
oper_mul	*
oper_min	−
oper_div	/
left_p	(
right_p	)
oper_equal	=
doper_equal	==
oper_diferent	=!
oper_more	>
oper_less	<
oper_moreE	>=
oper_lessE	<=
if	if
else	else
while	meanwhile
for	repeat
print	show
funcion	”def”
type_int	int
type_float	float
type_String	String
type_bool	Fifty-Fifty
Key_left	{
Key_right	}
coment	//.*//
literal	.+
Boolean	true—false
float	[0-9]+,[0-9]+
oper_dotc	;
oper_com	,
return	give

4. Gramatica

FUNCION − > def identificador DECF Key_left EXP Key_right
FUNCION − > E
FUNCION − > V E’ oper_dotc FUNCION
DECF − > left_p REC right_p
E − > TYPE Q oper_dotc
E − > BUCLE
E − > IF
E − > print left_p EXP1 EXP right_p oper_dotc
Q − > oper_equal V E’
Q − > ”
COM − > oper_com REC

COM − > ”

REC − > V COM
REC − > ”
A − > identificador
R − > oper_plus oper_plus
R − > oper_min oper_min
BUCLE − > for left_p ID C numeral oper_com ID C numeral oper_com ID R right_p Key_left EXP Key_right EXP
BUCLE − > while left_p ID C V right_p Key_left EXP Key_right EXP
BUCLE − > ”
IF − > if left_p EXP1 right_p Key_left EXP Key_right ELSEIF
ELSEIF − > elif left_p EXP1 right_p Key_left EXP Key_right ELSE
ELSEIF − > ELSE
ELSE − > else Key_left EXP Key_right
ELSEIF − > ”
C − > doper_equal
C − > oper_equal
C − > oper_diferent
C − > oper_more
C − > oper_less
C − > oper_moreE
C − > oper_lessE
J − > V E’
E’ − > O V E’
E’ − > DECF
E’ − > ”
O − > oper_plus
O − > oper_min
O − > oper_mul
O − > oper_div
O − > oper_equal
O − > doper_equal
O − > reserved_and
O − > reserved_or
EXP − > ID E’ oper_dotc E
EXP − > return left_p J right_p E’ oper_dotc EXP]
EXP − > E
EXP1 − > V E’
V − > numeral
V − > float
V − > literal
V − > Boolean
V − > ID
ID − > identificador
TYPE − > type_String identificador
TYPE − > type_int identificador
TYPE − > type_float identificador
TYPE − > type_bool identificador

5. Ejemplos de código correcto

5.1. declaracion de variable

x = 5; if (y == true) { give (x) ; }
identificador oper_equal numeral oper_dotc if left_p identificador doper_equal Boolean right_p Key_left return left_p identificador right_p oper_dotc Key_right

5.2. Declaracion funcion

def promedio(x, y){give (x+y)/2; a = 1; meanwhile (a < 10){ b = promedio(a, 5); if (b == true){ show (”finish1”);} else show (”finish2”) ; }}}
def identificador left_p identificador oper_com identificador right_p Key_left return left_p identificador oper_plus identificador right_p oper_div numeral oper_dotc identificador oper_equal numeral oper_dotc while left_p identificador oper_less numeral right_p Key_left identificador oper_equal identificador left_p identificador oper_com numeral right_p oper_dotc if left_p identificador right_p Key_left print left_p literal right_p oper_dotc Key_right else Key_left print left_p literal right_p oper_dotc Key_right Key_right Key_right

5.3. Funcion con if

x = 5; if (y == true) { show (”hola”) ; }
identificador oper_equal Boolean oper_dotc if left_p identificador doper_equal Boolean right_p Key_left print left_p literal right_p oper_dotc Key_right

6. Ejemplos de código erroneo

6.1. Error en puntuación

x = 5 if (y == true) { give (x) ; }
identificador oper_equal numeral if left_p identificador doper_equal Boolean right_p Key_left return left_p identificador right_p oper_dotc Key_right

6.2. Error declaracion de funcion

def (x, y){give (x+y)/2; a = 1; meanwhile (a < 10){ b = promedio(a, 5); if (b == true){ show (”finish1”);} else show (”finish2”) ; }}}
def left_p identificador oper_com identificador right_p Key_left return left_p identificador oper_plus identificador right_p oper_div numeral oper_dotc identificador oper_equal numeral oper_dotc while left_p identificador oper_less numeral right_p Key_left identificador oper_equal identificador left_p identificador oper_com numeral right_p oper_dotc if left_p identificador right_p Key_left print left_p literal right_p oper_dotc Key_right else Key_left print left_p literal right_p oper_dotc Key_right Key_right Key_right

6.3. Error en sintaxis

x = 5; if (y == true) { show (”hola”) ; }
identificador oper_equal Boolean oper_dotc if left_p identificador doper_equal Boolean right_p Key_left left_p literal right_p oper_dotc Key_right