Documentación SPSScript

Estudiantes

William Cole - 22241363

Fabrizio Ramos - 22241140

Clase

Lenguajes de Programación

Catedrático

Ing. Lucas Cueva

Fecha

30-11-2024

Índice

Introducción

Este lenguaje de programación, se inspira en expresiones y modismos característicos de San Pedro Sula, Honduras. Su objetivo es hacer que el desarrollo de software sea más accesible y entretenido para personas familiarizadas con el lenguaje coloquial de la región.

SPSScript utiliza una sintaxis basada en palabras clave locales que sustituyen términos técnicos comunes en otros lenguajes. Por ejemplo, "PoneleQue" se utiliza para declarar variables, y "Ejenie" se usa para imprimir en consola.

Ejecutar

Para poder empezar a utilizar el lenguaje de programación SPS Script se debe de hacer lo siguiente:

1. Clonar el repositorio de SPSScript: https://github.com/coleexz/SPSScript.git
2. Abrir el repositorio en tu editor de código preferido
3. Abrir la terminal de tu editor de código y escribir lo siguiente: python3 server.js
4. Presionar la tecla enter
5. Después escribir este comando: cd interprete-frontend
6. Ejecutar el siguiente comando para correr el frontend: npm start
7. Abrir un archivo dentro de la página con extensión .sps, y empezar a programar!

Palabras Clave

A continuación, se detallan las palabras clave y su equivalente funcional en términos tradicionales:

Palabra Clave	**Funcion**	**Ejemplo De Uso**
PoneleQue	Declarar variable	PoneleQue a = 3
y	Operador logico AND	PoneleQue a = Real y Casaca
o	Operador logico OR	PoneleQue a = Real o Casaca
NadaQueVer	Operador NOT	PoneleQue a = NadaQueVer Real
PoneteAPensar	Condicional IF	PoneteAPensar a Tonces Ejenie "Hola"
Decidite	Condicional ELSEIF	Decidite b Tonces Ejenie "Tal vez"
HaceteLoco	Condicional ELSE	HaceteLoco Tonces Ejenie "Adiós"
PasarLista	Ciclo For	PasarLista PoneleQue i = 0 Hasta 10 Tonces ...
AhoritaQue	Ciclo While	AhoritaQue a < 10 Tonces ...
HacemeElParo	Declarar una funcion	HacemeElParo suma(a, b) -> a + b
Cheque	End	PoneteAPensar a Tonces ... Cheque

SigaMiPerro	Continuar loop	Dentro de un ciclo: SigaMiPerro
CortalaMiPerro	Cerrar loop	Dentro de un ciclo: CortalaMiPerro
Ejenie	Imprimir	Ejenie ("Hola, San Pedro")
Nada	Nulo	PoneleQue a = Nada
Real	True	PoneleQue b = Real
Casaca	False	PoneleQue c = Casaca

Tipos de Datos

El lenguaje soporta los siguientes tipos de datos:

- Números (Number): Representa enteros y flotantes. Ejemplo: 5, 3.14.
- Cadenas (String): Texto delimitado por comillas dobles. Ejemplo: "Hola".
- Listas (List): Colecciones de elementos. Ejemplo: [1, 2, 3].
- Booleanos (Boolean): Representan Real (verdadero) o Casaca (falso).
- Nulo (Nada): Representa ausencia de valor.

Funciones Incorporadas

El lenguaje cuenta con funciones integradas para operaciones comunes:

Funcion	Descripcion	Ejemplo
EsFilaIndia	Verifica si un valor es una lista	EsFilaIndia(x)
EsEntero	Verifica si un valor es un entero	EsEntero(x)
EsParo	Verifica si un valor es una funcion	EsParo(multiplicar)
EsChambre	Verifica si un valor es un string	EsChambre(nombre)
Saca	Quita un elemento de una lista	Saca(x,4)
Mete	Agrega un elemento a una lista	Mete(x,4)
Mide	Longitud de una lista	Mide(x)

Operadores

Aritméticos

Operador	Significado	Ejemplo
+	Suma	5+3
-	Resta	5-3
*	Multiplicación	5*3
/	División	5/3
^	Potencia	2^3

Comparaciones

Operador	**Significado**	**Ejemplo**
==	Igual	5==5
!=	Distinto	5!=3
>	Mayor que	3<5
<	Menor que	5>3
<=	Menor o igual que	3<=5
>=	Mayor o igual que	5>=3

Ejemplos

Declaración de Variables

```
PoneleQue x = 10
PoneleQue mensaje = "Bienvenido a San Pedro"
Ejenie(mensaje)
```

Condicionales

```
PoneleQue edad = 18
PoneteAPensar edad >= 18 Tonces
  Ejenie ("Sos mayor de edad")
Cheque
```

Ciclos

```
PasarLista PoneleQue i = 0 Hasta 5 Tonces
  Ejenie(i)
Cheque
`
```

Funciones

```
HacemeElParo saludar(nombre) ->
  Ejenie ("Hola, ",nombre)

saludar("San Pedro")
```

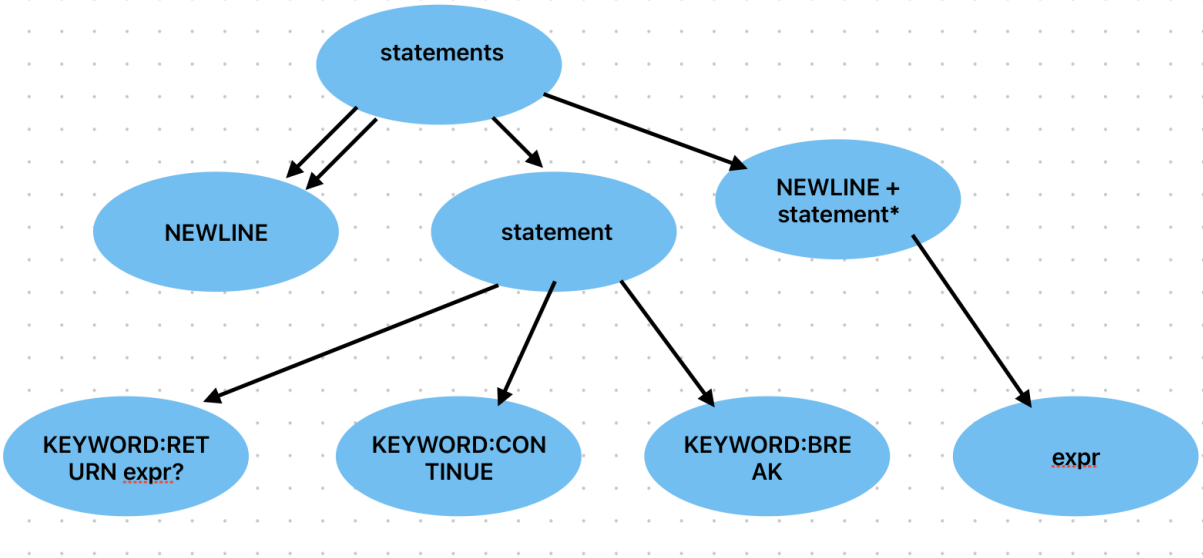
Gramática Tecnica

- Esta es la gramatica técnica que utiliza nuestro lenguaje SPS Script

```
statements  : NEWLINE* statement (NEWLINE+ statement)* NEWLINE*

statement                : KEYWORD:RETURN expr?
                                                    : KEYWORD:CONTINUE
                                                    : KEYWORD:BREAK
                                                    : expr

expr        : KEYWORD:VAR IDENTIFIER EQ expr
            : comp-expr ((KEYWORD:AND|KEYWORD:OR) comp-expr)*

comp-expr   : NOT comp-expr
            : arith-expr ((EE|LT|GT|LTE|GTE) arith-expr)*

arith-expr  :     term ((PLUS|MINUS) term)*

term        : factor ((MUL|DIV) factor)*

factor      : (PLUS|MINUS) factor
            : power

power       : call (POW factor)*

call        : atom (LPAREN (expr (COMMA expr)*)? RPAREN)?

atom        : INT|FLOAT|STRING|IDENTIFIER
            : LPAREN expr RPAREN
            : list-expr
            : if-expr
            : for-expr
            : while-expr
            : func-def

list-expr   : LSQUARE (expr (COMMA expr)*)? RSQUARE

if-expr     : KEYWORD:IF expr KEYWORD:THEN
              (statement if-expr-b|if-expr-c?)
            | (NEWLINE statements KEYWORD:END|if-expr-b|if-expr-c)

if-expr-b   : KEYWORD:ELIF expr KEYWORD:THEN
              (statement if-expr-b|if-expr-c?)
            | (NEWLINE statements KEYWORD:END|if-expr-b|if-expr-c)
```

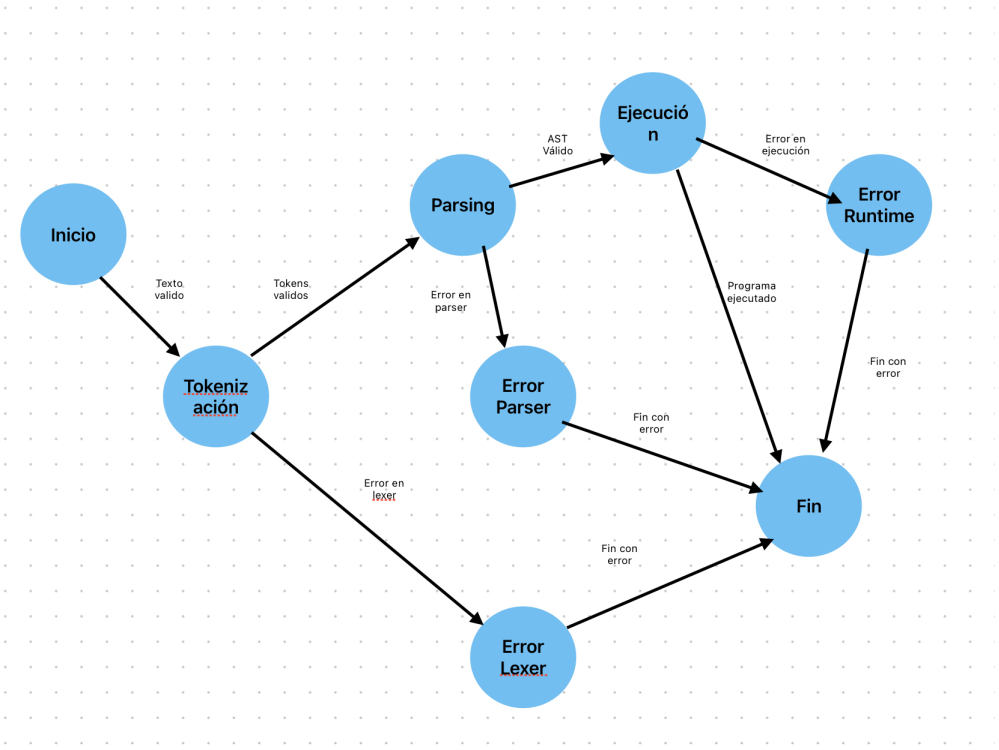
```
if-expr-c   : KEYWORD:ELSE
              statement
            | (NEWLINE statements KEYWORD:END)

for-expr    : KEYWORD:FOR IDENTIFIER EQ expr KEYWORD:TO expr
              (KEYWORD:STEP expr)? KEYWORD:THEN
              statement
            | (NEWLINE statements KEYWORD:END)

while-expr  : KEYWORD:WHILE expr KEYWORD:THEN
              statement
            | (NEWLINE statements KEYWORD:END)

func-def    : KEYWORD:FUN IDENTIFIER?
              LPAREN (IDENTIFIER (COMMA IDENTIFIER)*)? RPAREN
              (ARROW expr)
            | (NEWLINE statements KEYWORD:END)

class-def : KEYWORD:Clase IDENTIFIER LBRACE statements RBRACE

method-call : atom DOT IDENTIFIER LPAREN (expr (COMMA expr)*)? RPAREN
```

Árbol de gramática

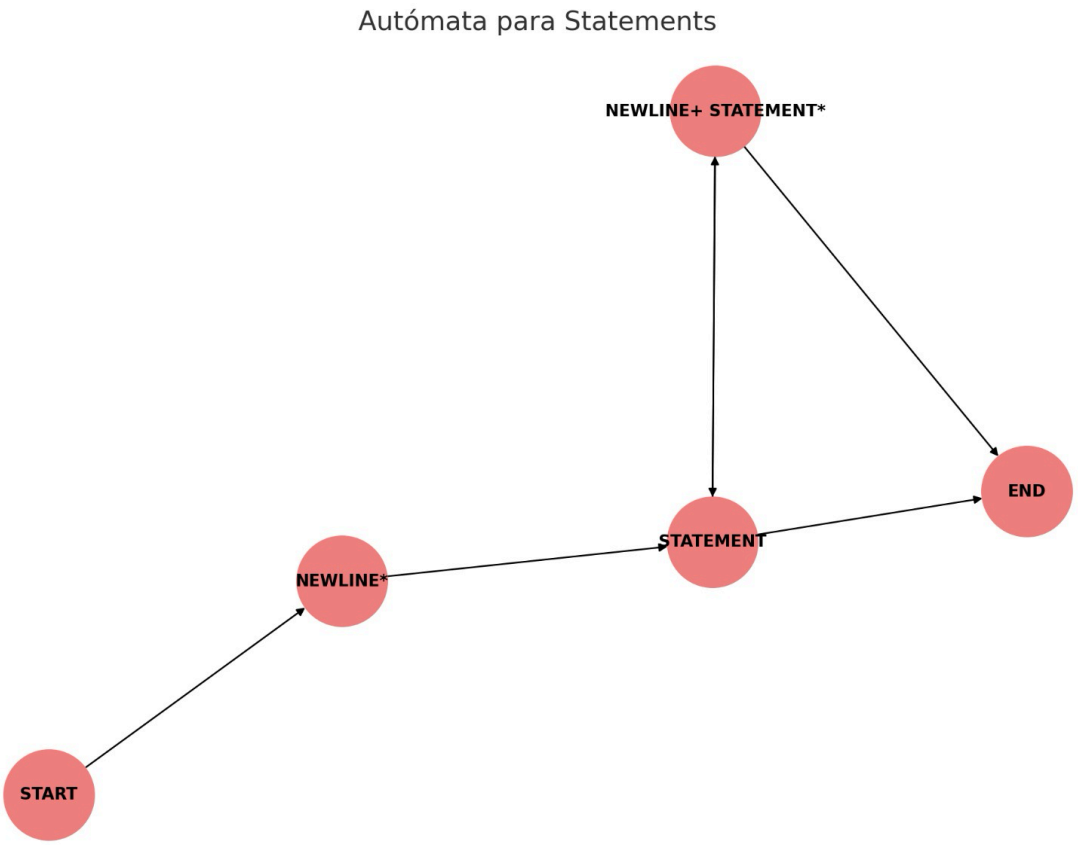


Autómatas

Autómata para errores en ejecución



Autómata para statements



Autómata para flujo de ejecución