SOCCER CHAT - BOT

ANALIZADOR LEXICO Y ANALIZADOR SINTÁCTICO

LENGUAJES FORMALES Y DE PROGRAMACION

PROYECTO 2

2022

ESTUARDO SEBASTIÁN VALLE BANCES
202001954
3674674601O1

INDICE

1. Definición de Expresiones Regulares

Lenguaje de Entrada (Comandos)

RESULTADO equipo VS equipo TEMPORADA <YYYY-YYYY>

JORNADA número TEMPORADA <YYYY-YYYY> [-f archivo]

GOLES condición equipo TEMPORADA <YYYY-YYYY>

TABLA TEMPORADA <YYYY-YYYY> [-f archivo]

PARTIDOS equipo TEMPORADA <YYYY-YYYY> [-f archivo] [-ji número] [-jf número]

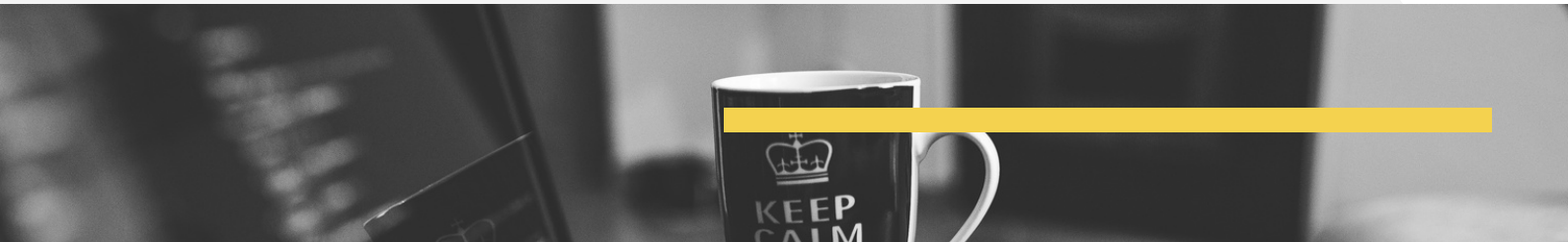
TOP condición TEMPORADA <YYYY-YYYY> [-n número]

ADIOS

Expresiones Regulares

1. Definicion de Palabras Reservadas

Desglosando el Lenguaje				
PALABRAS RESERVADAS				
No.		Caracter	Lexema	Token
1	RESULTADO	RESULTADO	RESULTADO	RESULTADO
2	TEMPORADA	TEMPORADA	TEMPORADA	TEMPORADA
3	VS	VS	VS	VERSUS
4	JORNADA	JORNADA	JORNADA	JORNADA
5	.f	/F	/f	GUIONF
6	GOLES	GOLES	GOLES	GOLES
7	LOCAL	LOCAL	LOCAL	LOCAL
8	TOTAL	TOTAL	TOTAL	TOTAL
9	VISITANTE	VISITANTE	VISITANTE	VISITANTE
10	TABLA TEMPORADA	TABLA TEMPORADA	TABLA TEMP	TEMPTABLA
11	PARTIDOS	PARTIDOS	PARTIDOS	PARTIDOS
12	/JI	/ji		GUIONJI
13	/JI	/ji		GUIONJF
14	TOP	TOP	TOP	TOP
15	/n	/n	/n	GUIONN
16	SUPERIOR	SUPERIOR	SUPERIOR	SUPERIOR
17	INFERIOR	INFERIOR	INFERIOR	INFERIOR
18	ADIOS	ADIOS	ADIOS	ADIOS
19	PARTIDOS	PARTIDOS	PARTIDOS	PARTIDOS



1. Definición de Expresiones Regulares

Lenguaje de Entrada

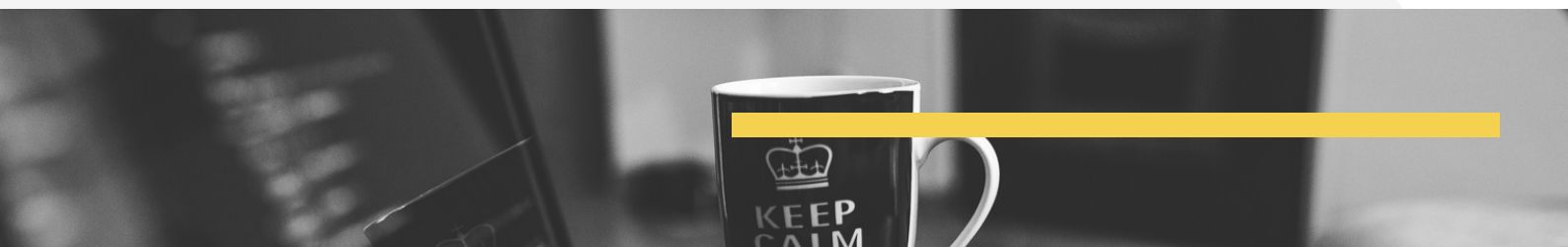
1. Definicion de Signos

		PALABRAS RESERVADAS			
No.		Caracter		Lexema	Token
1	RESULTADO	RESULTADO		RESULTADO	RESULTADO
2	TEMPORADA	TEMPORADA		TEMPORADA	TEMPORADA
3	VS	VS		VS	VERSUS
4	JORNADA	JORNADA		JORNADA	JORNADA
5	.f	/F		/f	GUIONF
6	GOLES	GOLES		GOLES	GOLES
7	LOCAL	LOCAL		LOCAL	LOCAL
8	TOTAL	TOTAL		TOTAL	TOTAL
9	VISITANTE	VISITANTE		VISITANTE	VISITANTE
10	TABLA TEMPORADA	TABLA		TABLA	TABLA
11	PARTIDOS	PARTIDOS		PARTIDOS	PARTIDOS
12	/JI	/ji			GUIONJI
13	/JI	/ji			GUIONJF
14	TOP	TOP		TOP	TOP
15	/n	/n		/n	GUIONN
16	SUPERIOR	SUPERIOR		SUPERIOR	SUPERIOR
17	INFERIOR	INFERIOR		INFERIOR	INFERIOR
18	ADIOS	ADIOS		ADIOS	ADIOS
19	PARTIDOS	PARTIDOS		PARTIDOS	PARTIDOS
		PATRONES NO ESPECIFICADOS			
No.		Caracter(es)		Lexema	Token
20	equipo	Cadena		C	STRING
21	year	YYYY		YYYY	YEAR
22	<	<		<	MENORQUE
23	>	>		>	MAYORQUE
24	archivo	Cadena		Delimitador (Letra_Numero)	ARCHIVO
25	numero	L[0-9]		L[0-9]	NUMERO
		[		[	LLAVEABRIR
		]		]	LLAVECERRAR
		/			GUION

AUTOMATA

2. Automata Finito Determinista

https://app.diagrams.net/#G1Nl7TLcPzmo9pR_-fJyWGqjvP4lhGhFII



3. Gramatica Definida

RESULTADO equipo VS equipo TEMPORADA <YYYY-YYYY>

JORNADA número TEMPORADA <YYYY-YYYY> [-f archivo]

GOLES condición equipo TEMPORADA <YYYY-YYYY>

TABLA TEMPORADA <YYYY-YYYY> [-f archivo]

PARTIDOS equipo TEMPORADA <YYYY-YYYY> [-f archivo] [-ji número] [-jf número]

TOP condición TEMPORADA <YYYY-YYYY> [-n número]

ADIOS

Terminales

T = { resultado, cadena, versus, temporada, jornada, integer, local, visitante, anos, archivo}

No Terminales

N = {INICIO, BLOQUEDOWHILE}

Producciones

INICIO ::= BLOQUERESULTADO

| BLOQUEJORNADA

| BLOQUEGOLES

| BLOQUETABLA

| BLOQUEPARTIDOS

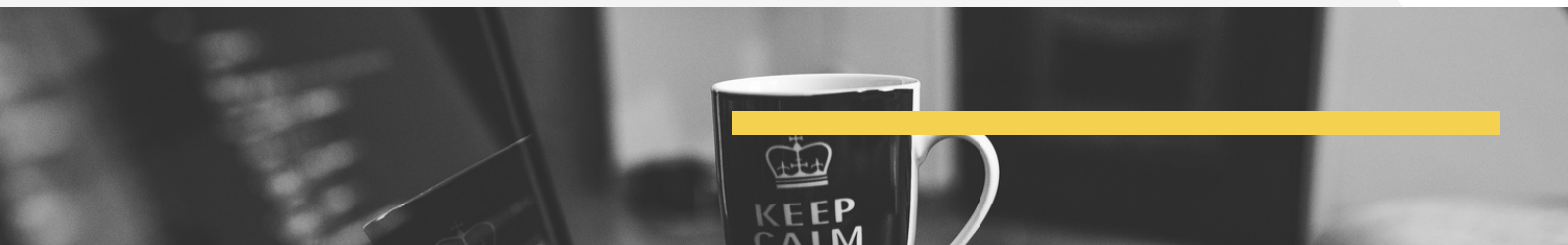
| BLOQUETOP

| BLOQUEADIOS

3. Gramatica Definida

```
BLOQUERESULTADO ::= resultado string versus string BLOQUETEMPORADA

BLOQUETEMPORADA::= temporada menorque integer guion integer mayorque

BLOQUEGOLES ::= goles CONDICION string BLOQUETEMPORADA

BLOQUETABLA ::= BLOQUETEMPORADA BLOQUEARCHIVO

BLOQUEARCHIVO::= llaveabrir BLOQUETIPO llavecerrar BLOQUEARCHIVOAUXILIAR


BLOQUEARCHIVOAUXILIAR ::= llavearbrir BLOQUETIPO llavecerrar BLOQUEARCHIVOAUXILIAR
                        | lambda

BLOQUEJORNADA ::= jornada integer BLOQUETEMPORADA BLOQUETIPO

BLOQUEADIOS :: = adios

BLOQUEPARTIDOS ::= partidos string BLOQUETEMPORADA BLOQUEARCHIVO

BLOQUETOP ::= top CONDICION BLOQUETEMPORADA BLOQUEARCHIVO

BLOQUETIPO ::= guionf archivo BLOQUETIPO
             | guionn number BLOQUETIPO
             | guionif number BLOQUETIPO
             | guionii number BLOQUETIPO
             | Epsilon()
```

```
CONDICION ::= local
            | visitante
```



3. Implementación

```python
class Token:
    def __init__(self, token: str, lexema: str, fila: int, col: int) -> None:
        self.token: str = token
        self.lexema: str = lexema
        self.fila: int = fila
        self.col: int = col
```

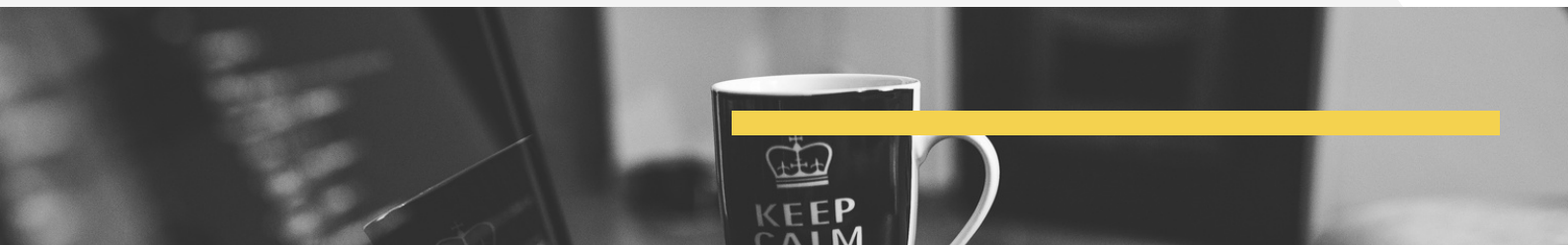
<tokenClass, lexema>

Al entrar en un estado de aceptación, se crea un objeto de Token y se añade a una Lista de Tokens

Además es necesario saber donde se encuentra el Token para un futuro analisis, por lo que col y fila declaran la columna y linea.

```python
class ErrorEntry:
    def __init__(self, linea: int, col: int, char: str) -> None:
        self.linea: int = linea
        self.col: int = col
        self.char: str = char
        self.type: str = 'Lexico'
        self.token: str = 'STRING'
        self.razon: str = 'No pertenece al Lenguaje'
```

Si no se cumplen las condiciones del automata, este entra en un error. Los errores deben ser rastredos a partir de linea y columna.



3. Implementación

```
def automata(input: str) -> Tuple[Tuple[Token], Tuple[ErrorEntry]]:

    # Agregar char al final
    tokens: List[Token] = []  # Lista tokens
    errores: List[ErrorEntry] = []  # Lista errores
    estado: int = 1  # Estado inicial
    lexema: str = ''  # lexema actual
    index: int = 0  # indice

    row: int = 1  # fila
    col: int = 0  # columna
    print("El Archivo>>")
    print(str(len(input)))
```

El automata fue programado a partir de n WHILE que se mueve con la variable CHAR indicada por INDEX

```
while index < len(input):
    char = input[index]

    # Estado inicial
    if estado == 1:
        #Lista de transiciones
        if char == 'f' or char == 'F':
            estado = 2
            index += 1
            col += 1
            lexema += char
```

Envio a otro Estado

Siguiente Caracter

Guardado de Caracter en Lexema

cada estado tiene una serie de validaciones

```
elif estado == 11:
    #Sera estado de aceptacion para palabras reservadas
    estado = 1
    tokens.append(Token('PALABRA RESERVADA', lexema, row, col))
    lexema = ''
```

3. Implementación

```
class AnalizadorSintactico:
    def __init__(self):
        self.listaTokens = []
        self.listaErrores = []
        self.i = 0
        self.partidos = []
```

```
def inicio(self, answer):
    if answer != None:
        return answer
    else:
        token = self.listaTokens[self.i]
        if token.token == 'RESULTADO':
            print('RESULTADO')
            return self.bloqueResultado()
        elif token.token == 'JORNADA':
            return self.bloqueJornada()
        elif token.token == 'GOLES':
            return self.bloqueGoles()
        elif token.token == 'TOP':
            return self.bloqueTOP()
        elif token.token == 'PARTIDOS':
            return self.bloquePartidos()
        elif token.token == 'ADIOS':
            return self.bloqueAdios()
        elif token.token == 'TABLA':
            return self.bloqueTabla()
```

```
BLOQUERESULTADO ::= resultado string versus string BL
BLOQUETEMPORADA::= temporada menoraue intege
BLOQUEGOLES ::= goles CONDICION string BLOQUETE
BLOQUETABLA ::= BLOQUETEMPORADA BLOQUEARCH
BLOQUEARCHIVO::= llaveabrir BLOQUETIPO llavecerro

BLOQUEARCHIVOAUXILIAR ::= llavearbrir BLOQUETIPO
                        | lambda
BLOQUEJORNADA ::= jornada integer BLOQUETEMPO
BLOQUEADIOS :: = adios
BLOQUEPARTIDOS ::= partidos string BLOQUETEM
BLOQUETOP ::= top CONDICION BLOQUETEMPORADA
BLOQUETIPO ::= quionf archivo BLOQUETIPO
             | quionn number BLOQUETIPO
             | quionif number BLOQUETIPO
             | quionii number BLOQUETIPO
             | Epsilon()
```

3. Implementación

3. 1 Funciones Base y Funciones Auxiliares

BLOQUERESULTADO ::= resultado string versus string BLOQUETEMPORADA

BLOQUETEMPORADA::= temporada menorque integer guion integer mayorque

BLOQUEGOLES ::= goles CONDICION string BLOQUETEMPORADA

BLOQUETABLA ::= BLOQUETEMPORADA BLOQUEARCHIVO

BLOQUEARCHIVO::= llaveabrir BLOQUETIPO llavecerrar BLOQUEARCHIVOAUXILIAR

Errores Base

```
else:
    error = ErrorEntry(1, 1, token.lexema)
    error.type = 'Sintactico'
    error.token = token.token
    error.razon = 'Se Esperaba un comando'
    self.listaErrores.append(error)
    return token.lexema + '\n^\nSe Esperaba un comando'
```

Errores Auxiliares

```
if token.token == 'STRING':
    equipoVisitante = token.lexema
    print(equipoVisitante)
    self.i += 1
    token = self.listaTokens[self.i]
    year1, year2, queja = self.bloqueTemporada()
    if queja != '':
        queja = '\n'+ queja
    if year1 == '' or year2 ==='':
        return 'No Hay Informacion Suficiente' + queja
```

3. Implementación

Reportar Errores

```
else:
    error = ErrorEntry(1, token.col, token.lexema)
    error.type = 'Sintactico'

    error.token = token.token
    error.razon = 'Se Esperaba un -'
    self.listaErrores.append(error)
    espacio = '    '
    queja += token.lexema + '\n' + espacio + '^\nSe Esperaba un "-"'
```

4. Funciones

```
def _insert_message(self, msg, sender):
    if not msg:
        return

    self.msg_entry.delete(0, END)
    msg1 = f"{sender}: {msg}\n\n"
    self.text_widget.configure(state=NORMAL)
    self.text_widget.insert(END, msg1)
    self.text_widget.configure(state=DISABLED)
    #Se realiza el analisis lexico
    tokens, errs = automata(msg)
    #Se anade el analisis lexico a la lista principal
    self.tokenList.extend(tokens)
    self.errList.extend(errs)

    #Se anade el analisis sintactico
    sintaxis = AnalizadorSintactico()
    answer, syntaxErrs = sintaxis.analizar(tokens, errs, self.objects)
    self.errList.extend(syntaxErrs)
```