Creación y especificaciones lexicas para tokens determinados

Universidad internacional de la rioja

Rogerio Orlando Beltran Castro

Procesadores de lenguajes

Oscar David Bocanegra Capera

16/septiembre/2024

Para dar solución a este trabajo lo que vamos a hacer es un analizador léxico con Python en donde él va a reconocer palabras, frases entre otras cosas de un lenguaje de programación las cuales estaré explicando más adelante las que se implementaran para este trabajo.

Para poder dar un poco más de contexto del trabajo que voy a presentar puedo indicar que lo realice con Python ya que es uno de los lenguajes que más uso en mis labores como ingeniero y con el que mejor me desempeño, por esta razón y para poder desarrollar este trabajo lo que hacemos es usar la biblioteca de PLY la cual nos va a facilitar la creación de nuestro analizador léxico, por ende, anexare una imagen de cómo es la importación de dicha librería.

Aclarado esto pasamos a uno de los puntos más importantes de este trabajo y es aclarar que tokens son los que vamos a utilizar para esta actividad, por esta razón voy a pegar una imagen de los tokens que he definido, y en donde adicionalmente también el programa va a ser capaz de reconocer palabras clave que se usa en el lenguaje de programación.

Definición de Tokens.

```
# Lista de nombres de tokens
tokens = [
    'KEYWORD',
    'IDENTIFIER',
    'NUMBER',
    'STRING',
    'ARITHMETIC_OPERATOR',
    'LOGICAL_OPERATOR',
    'COMPARISON_OPERATOR',
    'PARENTHESIS',
    'SEMICOLON',
    'COMMENT_LINE',
    'COMMENT_BLOCK'
]
```

```
# Definir palabras clave
keywords = {
    'if': 'KEYWORD',
    'else': 'KEYWORD',
    'while': 'KEYWORD',
    'for': 'KEYWORD',
    'int': 'KEYWORD',
    'float': 'KEYWORD',
    'string': 'KEYWORD'
}
```

Ahora dando un contexto general de los tokens podemos ver que los “keyword” son las palabras clave que usamos en el programa como lo son el if, else, while etc.

Number va a identificar los numero enteros o flotantes.

String va a identificar las cadenas de texto que le pasemos, cabe aclarar que en este caso se debe de pasar con doble comilla ya que es como definimos una cadena de texto en Python.

Atithmetic_operator este se encarga de identificar todo lo que sea un operador matemático como lo puede ser alguno de los siguientes +, *, -, /

También encontramos en la lista los operadores lógicos los cuales como lo son &&, || etc.

De la misma manera encontramos los operadores de comparación los cuales son <, >, ==, !=

El programa también va a pdoer reconocer los paréntesis, llaves y corchetes

Va a poder reconocer los comentarios como se harían en el programa de Python los cuales se hacen usando la siguiente estructura “//”

Y por último va a tener también uno especial que identifica los caracteres desconocidos ya que estos no fueron definidos.

Descripción código.

Ahora como ya vimos los tokens que se definieron ahora en la siguiente imagen vamos a poder ver cómo se definieron las reglas del analizador léxico para reconocer los tokens que definimos anteriormente.

```python
# Definir comentarios
def t_COMMENT_BLOCK(t):
    r'/\*.*?\*/'
    pass  # Ignorar comentarios de bloque

def t_COMMENT_LINE(t):
    r'//.*'
    pass  # Ignorar comentarios de una línea

# Definir números (enteros y flotantes)
def t_NUMBER(t):
    r'\d+(\.\d+)?'
    t.value = float(t.value) if '.' in t.value else int(t.value)
    return t

# Definir cadenas de texto
def t_STRING(t):
    r'\"([^\\"]|\\.)*\"'
    return t

# Definir identificadores y palabras clave
def t_IDENTIFIER(t):
    r'[a-zA-Z_][a-zA-Z_0-9]*'
    t.type = keywords.get(t.value, 'IDENTIFIER')  # Verificar si es una palabra clave
    return t

# Manejo de errores
def t_error(t):
    print(f"Caracter ilegal '{t.value[0]}'")
    t.lexer.skip(1)

# Construir el lexer
lexer = lex.lex()
```

Por ende, podemos decir que en la imagen anterior vemos el proceso que realiza el programa para poder identificar qué tipo de carácter está recibiendo el programa y después de esto informándonos a que hace referencia.

```
david@pcdavid:~/personal_folder/Universidad/procesadores_de_lenguaje/trabajo2$ python3 lexer.py
Bienvenido al analizador léxico interactivo.
Introduce el código que deseas analizar (o 'salir' para terminar):
>> if
Token encontrado: KEYWORD, Valor: if
>> else
Token encontrado: KEYWORD, Valor: else
>> els
Token encontrado: IDENTIFIER, Valor: els
>> "Hola mundo"
Token encontrado: STRING, Valor: "Hola mundo"
>> 10
Token encontrado: NUMBER, Valor: 10
```

Ahora en la imagen anterior podemos ver que el programa es algo interactivo con el usuario, y además de esto podemos ver la prueba la identificación de diferentes tokens como lo son un String, un indefinido, un número y una palabra clave.

Ahora veremos cómo es esta parte interactiva del programa a nivel código, ya que en este caso decidí hacerlo así por el hecho de que sea interactivo con el usuario y este pueda preguntar los diferentes tokens que se le ocurran y el programa le contesta de la mejor manera posible.

Anexo la imagen del código en la parte interactiva

```python
# Interacción con el usuario
def main():
    print("Bienvenido al analizador léxico interactivo.")
    print("Introduce el código que deseas analizar (o 'salir' para terminar):")

    while True:
        # Solicitar entrada del usuario
        user_input = input(">> ")

        # Si el usuario escribe 'salir', termina el programa
        if user_input.lower() == 'salir':
            print("Saliendo del analizador léxico. ¡Hasta luego!")
            break

        # Alimentar el código ingresado al lexer
        lexer.input(user_input)

        # Tokenizar la entrada y mostrar resultados
        while True:
            tok = lexer.token()
            if not tok:
                break
            print(f"Token encontrado: {tok.type}, Valor: {tok.value}")

# Ejecutar el programa principal
if __name__ == "__main__":
    main()
```