Documentation

Introduction:

Lexical Analyzer, is the process of converting a sequence of characters into a sequence of tokens (strings with an assigned and thus identified meaning). A program that performs lexical analysis may be termed a scanner.

Scanner: The first stage, the scanner, is usually based on a finite-state machine (FSM). It has encoded within it information on the possible sequences of characters that can be contained within any of the tokens it handles (individual instances of these character sequences are termed lexemes). For example, an integer token may contain any sequence of numerical digit characters. In many cases, the first non-whitespace character can be used to deduce the kind of token that follows and subsequent input characters are then processed one at a time until reaching a character that is not in the set of characters acceptable for that token (this is termed the maximal munch, or longest match, rule). In some languages, the lexeme creation rules are more complex and may involve backtracking over previously read characters. For example, in C, one 'L' character is not enough to distinguish between an identifier that begins with 'L' and a wide-character string literal.

CODE:

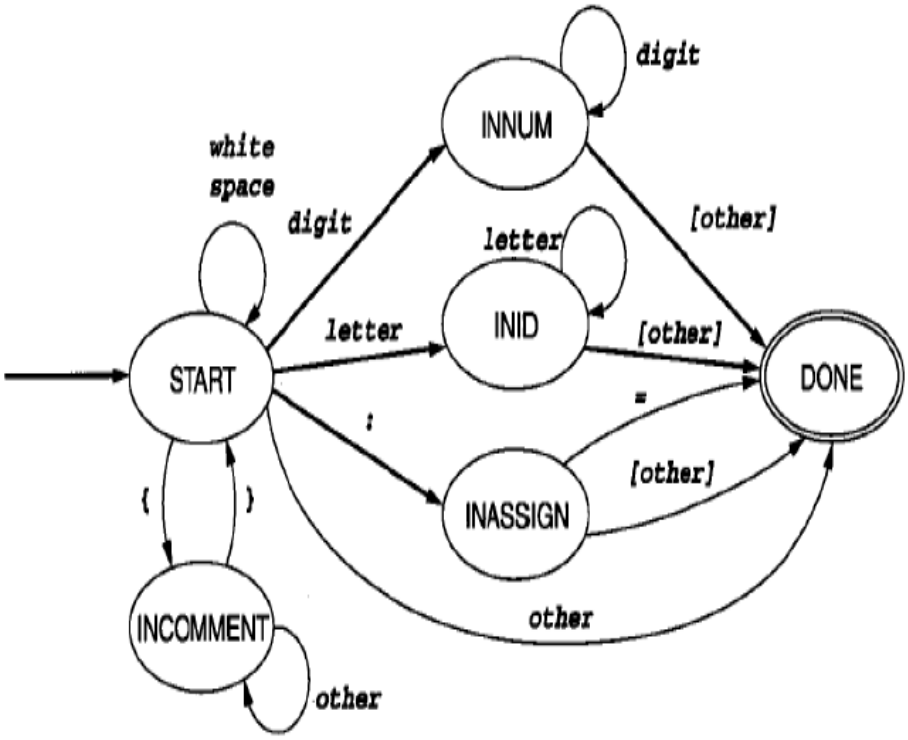
It contains 2 files, LexicalAnalyzerClass.py in which source code is written and the other file LexicalAnalyzer.py executes the source code.

LexicalAnalyzer.py | LexicalAnalyzerClass.py

```
1  from LexicalAnalyzerClass import LexicalAnalyzer
2  X = LexicalAnalyzer()
3  X.file_process()
```

- **LexicalAnalyzerClass.py:**
  It contains 3 parts:
  - def__init__ ():
    in which I define reserved words, special symbols, token "empty array", set value and type of tokens empty, and initially define current state = 1 and current position = 0.

  - def take_token ():



First, I suppose the initial state >> START “current state = 1”
Then we follow the incoming input to know the next state
Cases:

1- next state = “{“we go to INCOMMENT “current state = 2”
2- next state = “:” we go to INASSIGN “current state = 3”
3- next state = “+ - * / = < ( ) ; :=” we go to DONE “current state = 6”
4- next state = “letter” we go to INID “current state = 4”
5- next state = “number” we go to INNUM “current state = 5”
   - if the current state = 2 and the next input was “}” it will go to state number 1 “START”.
   - If the current state = 3 and the next input was “=” It will go to state number 6 “DONE”.
   - If the current state = 4 and the next input was letter it will go to the same state number 4 “INID” but if the next input was anything other it will go to state number 6 “DONE”.
   - If the current state = 5 and the next input was number it will go to the same state number 5 “INNUM” but if the next input was anything other it will go to state number 6 “DONE”.

When it reaches to state number 6 “DONE” it appends the value and the type of the input in tokens.

- def file_process ():
  in which we do a process of read the file. We pass through each character line by line and put it tokens then do operations and write the results in output file which we generate it.