FLCD Scanner Documentation

Github repo: https://github.com/MihaiSilinc/FLCD

Symbol Table:

The structure of the Symbol Table is an Alphabetically Sorted List that has 2 methods implemented, insert and search. The insert method adds a new element into the list in the correct position and returns that position.

Program Internal Form:

The PIF is kept as a simple list having pairs consisting of (token, position), where position represents the position of said token in the Symbol Table. In case of operators, separators and reserved words they are not stored in the Symbol Table, so I just inform the user that they are either an Operator / Sep / Reserved.

Tokenizing:

The tokenizing method verifies if that character is an operator, separator, string, is building an identifier or constant, adds the token to a list and returns that list.

In the ***Run*** function, I take each line from the file, strip it of spaces and run the tokenizing method on it, and for each token found we add it to the PIF.

If its an Operator / Separator / Reserved Word the position from the PIF won't be shown and if it's an Identifier or Constant I print the type and the position in the Symbol Table. If it founds an error, it appends that error to an string that contains all errors.

```
    def run(self, error=None):
        self.readFile()
        print(self.scanner.separators, self.scanner.operators, self.scanner.keywords)
        fileName = "p3.txt"
        tokensList = self.scanner.keywords + self.scanner.separators + self.scanner.operators
        error = ''
        with open(fileName, 'r') as file:
            lineIndex = 0
            for line in file:
                lineIndex += 1
                tokens = self.scanner.tokenize(line.strip())
                extra = ''
                for i in range(len(tokens)):
                    if tokens[i] in tokensList:
                        if tokens[i] == ' ':
                            continue
                        self.pif.add(tokens[i], "Operator / Sep / Reserved")
                    elif self.scanner.checkIdentifier(tokens[i]):
                        identifier = self.symbolTable.insert(tokens[i])
                        self.pif.add("Identifier", identifier)
                    elif self.scanner.isConstant(tokens[i]):
                        constant = self.symbolTable.insert(extra+tokens[i])
                        extra = ''
                        self.pif.add("Constant", constant)
                    else:
                        error += 'Error at token ' + tokens[i] + ', line ' \
                                 + str(lineIndex) + "\n"
```