Gauti's_Scanner

March 5, 2018

1 *Implementation of Scanner*

Limitations :- Comments must be only at the end of RegEx
To find the correct result always restart the kernel and run all cells
This is necessary since we have used a global list

```
In [1]: from collections import defaultdict
        import numpy as np
        import re

In [2]: identifier_list = []
        flag = " "

In [3]: operators = ["+","-","/","*","%","<",">","(",")","="]
        names = ["PLUS","SUB","DIV","MUL","MOD","LT","GT","LPAREN","RPAREN","ASSIGN"]

        Dict = defaultdict()
        for i in range(len(operators)):
            Dict[operators[i]] = names[i]

In [4]: def splitter(info,line,flag):
            order = [" "]*len(line)
            pos = info.start()
            order[pos] = flag
            for i in range(len(line)):
                if line[i] in operators:
                    order[i] = Dict[line[i]]
            for i in range(len(order)):
                if order[i] != " ":
                    identifier_list.append(order[i])
            return

In [5]: def operator_checker(info,line):
            len_diff = info.end() - info.start()
            if len_diff == len(line):
                return 1
```

```
In [6]: def find_ID(line):
            if re.search(r"[0-9]+[a-zA-Z_]+",line):
                print "Invalid Token"
                return
            info = re.search(r"[a-zA-Z_][a-zA-Z0-9_]*",line)
            if info:
                flag = "ID"
                if operator_checker(info,line):
                    identifier_list.append(flag)
                    return 1
                else:
                    order = splitter(info,line,flag)
                    return 1

In [8]: def find_Num(line):
            if re.search(r"0[0-9]+",line):
                print "Invalid Number Token"
                return
            info = re.search(r"[0-9]+",line)
            if info:
                flag = "NUM"
                if operator_checker(info,line):
                    identifier_list.append(flag)
                    return 1
                else:
                    order = splitter(info,line,flag)
                    return 1

In [9]: def find_Operator(line):
            order = [" "]*len(line)
            for i in range(len(line)):
                if line[i] in operators:
                    order[i] = Dict[line[i]]
            if len(order) > 1:
                if order[0] == order[1]:
                    if order[0] == "ASSIGN":
                        order[0] = "EQUALTO"
                    if order[0] == "MUL":
                        order[0] = "EXP"
                order[1] = " "
            for i in order:
                if i != " ":
                    identifier_list.append(i)
            return 1

In [10]: def remove_comments(line):
             part = line.split("$")
             return part[0]
```

```
In [11]: def get_tokens(line):
             return line.split(" ")

In [12]: def RegEx2FsmConverter(regEx):
             regEx = remove_comments(regEx)
             tokens = get_tokens(regEx)
             for i in range(len(tokens)):
                 if find_ID(tokens[i]):
                     continue
                 if find_Num(tokens[i]):
                     continue
                 if find_Operator(tokens[i]):
                     continue

In [13]: #identifier_list = []
         regEx = "a_1 == (b < 0) + 2 ** c / 6$ can it be fun $"
         RegEx2FsmConverter(regEx)
         for i in identifier_list:
             print i,

ID EQUALTO LPAREN ID LT NUM RPAREN PLUS NUM EXP ID DIV NUM
```