1. All lexemes in origin file must be separated by a space, at which point they are matched to their proper token.
2. S ::🡺 Operation

Operation ::🡺 operand operator operand | ( operand operator operand ) | operand operator operand operator Operation

Operand ::🡺 int\_literal | identifier

Operator ::🡺 + | - | \* | / | % | = | != | ==

Code (also in attached .py file):

with open("sample.txt","r") as word\_list:

def isInt(s):

try:

int(s)

return True

except ValueError:

return False

def isOp(d):

if d == '+' or d == '-' or d == '\*' or d == '/' or d == '%' or d == '==' or d == '!=' or d == '=':

return True

else:

return False

#divides file into lexemes

lexemes = word\_list.read().split(' ')

syntax = True

#identifies tokens in file

for x in range(len(lexemes)):

if lexemes[x] == '-':

print(lexemes[x] + " token 22, subtract operator")

elif lexemes[x] == '+':

print(lexemes[x] + " token 21, add operator")

elif lexemes[x] == '\*':

print(lexemes[x] + " token 23, multiplication operator")

elif lexemes[x] == '/':

print(lexemes[x] + " token 24, division operator")

elif lexemes[x] == '(':

print(lexemes[x] + " token 25, left parenthesis")

elif lexemes[x] == ')':

print(lexemes[x] + " token 26, right parenthesis")

elif isInt(lexemes[x]) :

print(lexemes[x] + " token 10, integer literal")

elif lexemes[x] == '==':

print(lexemes[x] + " token 27, equal to")

elif lexemes[x] == '=':

print(lexemes[x] + " token 28, assignment operator")

elif lexemes[x] == '!=':

print(lexemes[x] + " token 29, not equal to")

elif lexemes[x] == '%':

print(lexemes[x] + " token 30, modulo")

else :

print(lexemes[x] + " token 11, identifier")

#checks syntax rules: closing parenthesis for every opening one, no hanging operators, etc

#checks proper paranthesis (an opening paranthesis always must have a closing one after)

par = 0

for y in range(len(lexemes)):

if lexemes[y] == '(':

par += 1

print(par)

elif lexemes[y] == ')':

par -= 1

print(par)

if par < 0:

syntax = False

#checks no hanging operators at start

if lexemes[0] == '+' or lexemes[0] == '-' or lexemes[0] == '\*' or lexemes[0] == '/' or lexemes[0] == '%' or lexemes[0] == '==' or lexemes[0] == '!=' or lexemes[0] == '=':

syntax = False

#checks no hanging operators at end

if lexemes[len(lexemes)-1] == '+' or lexemes[len(lexemes)-1] == '-' or lexemes[len(lexemes)-1] == '\*' or lexemes[len(lexemes)-1] == '/' or lexemes[len(lexemes)-1] == '%' or lexemes[len(lexemes)-1] == '==' or lexemes[len(lexemes)-1] == '!=' or lexemes[len(lexemes)-1] == '=':

syntax = False

#print (lexemes[x])

#checks each operator has no operators on either end (no instances of 'a + - b' or similar)

for z in range(len(lexemes)):

if isOp(lexemes[z]):

if z != 0 and z != len(lexemes)-1 and z != len(lexemes)-2:

if isOp(lexemes[z-1]) or isOp(lexemes[z+1]):

syntax = False

#print true if passed all syntax checks

print (syntax)]

sample text:

int ( x ) = 30 - 20