Evaluating Postfix Expression using Stack

2019330001 류시모

First, we open the 2 files using open command. For reading, we use f1 and for writing we’ll use f2.

f1 = open('postfix.in','r')

comds = f1.read().split('\n')

f2 = open('postfix.out','w')

Then we will use .split(‘\n’) command to take the entire string and verify them as lists of postfix expressions.

This will make the whole string into n different commands, so we can evaluate their values individually.

string = ''

while comds:

r = comds.pop(0)

#print(r)

if not r in ['',' ']:

pfexp = list(r)

#print(pfexp)

stack = []

We have two while statements. First one will go on until the “comds” is empty, meaning it will be responsible for list of different postfix expressions and their evaluation. Every command that is either empty or ‘ ‘ will be ignored.

while pfexp:

expr = pfexp.pop(0)

if expr != ' ':

#print(expr)

#print(stack)

if not expr in ['-','+','/','\*']:

stack.append(expr)

else:

r1 = stack.pop()

r2 = stack.pop()

command = ''.join([str(r2),expr,str(r1)])

stack.append(eval(command))

string = string + str(stack.pop()) + '\n'

Per every postfix expression, we have the next while statement that will separately make stack and manage the actual evaluation. As every element in comds is a string, we will convert it into list of characters. If a character is empty or ‘ ‘, we will ignore it. If a character is an operater(i.e. +, -. \*, /), we will do two pop, and evaluate the value using eval command.

After the second while statement is over, add the last element in the stack as final result. The entire thing will be written into postfix.out file using write command.

f2.write(string)

.