### Expression Tree ( Due 13 Nov 2020 )

For this assignment you will read a file [expression.in](https://www.cs.utexas.edu/users/mitra/csFall2020/cs313/assgn/expression.in) and create an expression tree. The expression will be a valid infix expression with the all the necessary parentheses so that there is no ambiguity in the order of the expression. You will evaluate the expression and print the result. You will also write the prefix and postfix versions of the same expression without any parentheses.

In an expression tree the nodes are either operators or operands. The operators will be in the set ['+', '-', '\*', '/', '//', '%', '\*\*']. The operands will be either integers or floating point numbers. All the operand nodes will be leaves of the expression tree. All the operator nodes will have exactly two children.

The outline of your program will be as follows:

import sys

class Stack (object):

class Node (object):

class Tree (object):

def \_\_init\_\_ (self):

def create\_tree (self, expr):

def evaluate (self, aNode):

def pre\_order (self, aNode):

def post\_order (self, aNode):

def main():

# read infix expression

line = sys.stdin.readline()

expr = line.strip()

# evaluate the expression and print the result

# get the prefix version of the expression and print

# get the postfix version of the expression and print

if \_\_name\_\_ == "\_\_main\_\_":

main()

To run this code on the command line on the Mac you will do

python3 ExpressionTree.py < expression.in

To run the code on the command line on a Windows machine you will do

python ExpressionTree.py < expression.in

The function *create\_tree()* will take as input parameter an infix expression with parentheses as a String and create an Expression Tree from it. *Assume that the expression string is valid and there are spaces between the operators, operands, and the parentheses.*

You will take the expression string and break it into tokens. There are four different kinds of tokens - left parenthesis, right parenthesis, operator, and operand. When we read a left parenthesis we are starting a new expression and when we read a right parenthesis we are ending an expression. Here is the algorithm that you will use. Start with an empty node that is going to be your root node. Call it the current node. Then start parsing the expression.

1. If the current token is a left parenthesis add a new node as the left child of the current node. Push current node on the stack and make current node equal to the left child.
2. If the current token is an operator set the current node's data value to the operator. Push current node on the stack. Add a new node as the right child of the current node and make the current node equal to the right child.
3. If the current token is an operand, set the current node's data value to the operand and make the current node equal to the parent by popping the stack.
4. If the current token is a right parenthesis make the current node equal to the parent node by popping the stack *if it is not empty*.

For the input expression, this is what your program will output:

( ( 8 + 3 ) \* ( 7 - 2 ) ) = 55.0

Prefix Expression: \* + 8 3 - 7 2

Postfix Expression: 8 3 + 7 2 - \*

The file that you will be submitting will be called **ExpressionTree.py**. We will be looking for good documentation, descriptive variable names, clean logical structure, and adherence to the coding conventions discussed in class.

You may work with a partner on this assignment. Both of you must read the paper on [Pair Programming](https://www.cs.utexas.edu/users/mitra/csFall2020/cs313/assgn/PairProg-CACM-1999.pdf) and abide by the ground rules as stated in that paper. The file will have a header of the following form:

# File: ExpressionTree.py

# Description:

# Student's Name:

# Student's UT EID:

# Partner's Name:

# Partner's UT EID:

# Course Name: CS 313E

# Unique Number:

# Date Created:

# Date Last Modified:

If you are working with a partner you will be submitting only one program (from either account) but make sure that you have your partner's name and eid in your program. If you are working alone, then remove the two lines that has the partner's name and eid in the header.

Use the [Canvas](http://canvas.utexas.edu/)system to submit your **ExpressionTree.py** file. We should receive your work by 11 PM on Friday, 13 Nov 2020. There will be substantial penalties if you do not adhere to the guidelines. Remember Python is case sensitive. The name of your file must match exactly what we have specified.

* Your Python program should have the proper header.
* Your code must run on the command line before submission.
* You should be submitting your file through the web based *Canvas* program. We will not accept files e-mailed to us.

#### References

* [Binary Expression Tree](http://en.wikipedia.org/wiki/Binary_expression_tree)