Last updated: 11/6/2019 3:30pm

For this assignment you will write Python interpreter for a tiny programming language, let's call it TL (Tiny Language). The language contains just three types of statements:

        let variableName = expression
        if expression goto label
        print expression1, expression2, ...
        input variableName

Each statement may contain a preceding label. A label is an alphanumeric string ending with a colon (":").

A number of simplifying assumptions have been made about the syntax of the language.
- White space (blanks) are important and must be used to separate each token including around the arithmetic operators.
- There can be only one statement per line.
- The expressions are limited to constant numbers, constant strings, variable names, and binary expressions involving just one of the following operators: "+", "-", "*", "/", "<", ">", "<=", ">=", "==", or "!=", with their conventional meanings. Note again that the operators must be surrounded by spaces, which makes for easier parsing.
- The only types are strings and floating point numbers (Float in Haskell) and strings are only used in print statements. The result of Boolean operations is 0 if false and 1 if true. Furthermore any numeric expression can be used in an if-statement and as with the C language, 0 is false and everything else is true.
- Blank lines are ignored.

let variableName = expression computes the value of *expression* then binds that value to the name *variableName*.

if expression goto label computes the value of *expression*, if the value is 0 execution continues with the next statement. If the value is non-zero then execution continues with the statement labeled *label*. If no such statement exists, the program terminates with the message: "Illegal goto *label* at line *x*." where x is the actual line number of the illegal goto statement.

print expression1, expression2, ... evaluates each expression, then prints their values, all on one line, separated by spaces, terminating the line with a newline character.

input variableName attempts to read a number from the standard input. If successful that value is bound to the name *variableName*. If the read fails the program should terminate with the message: "Illegal or missing input".

If at any point an attempt is made to evaluate an expression that references a variable *variableName* for which there is no binding, the program terminates with the message: "Undefined variable *variableName* at line *x*." where x is the actual line number of the failed expression.

Your program tli.py (tiny language interpreter) will take one command line argument, the name of the source file. It will compile the program into an internal form and then execute the compiled program. If the input program contains any syntax errors, tli should exit, printing a message of the form "Syntax error on line x." for the first syntax error detected, without attempting to execute the program. Although your program is allowed to continue and report additional syntax errors or provide additional error information, that is not required and will not affect your score.

tli.py should make just one pass over the program source building an internal representation that is a list of statements and a symbol table that maps labels into line numbers. The complete internal representation must be built before attempting to execute the first statement of the program. You may use this same symbol table to store variable bindings during execution of the program.

You are free to implement the program in anyway you see fit, with the single constraint that it must build an internal representation that is clearly derived from the starter code which provides the beginnings of a class for statements (Stmt) and a class for expressions (Expr). You are allowed to modify these as you see fit. Before program execution begins (after compilation) your program must have created a list of Stmt values that represent the statements of the TL program, and where appropriate, those Stmt objects must contain Expr values.

Here is a tiny language program that prints out a sequence of numbers.

```
        input start
        input end
        let x = start
repeat: print x
        let x = x + 1
        if x < end goto repeat
        print "that's all", x
```

Assuming the above program is stored in "prog1.txt", when executed the command "tli.py prog1.txt" and the user entering 1 and 5, tli.py should produce the output

```
1.0
2.0
3.0
4.0
that's all 5.0
```

You should turn in one file, tli.py.

Scoring will be done according to the following with the points in ()s.

(4) works for programs with no input and no goto

(3) works for programs with no input but goto

(2) works for all correct programs

(1) properly reports syntax and runtime errors

For this assignment you must use good programming style, including
- descriptive variable and function names
- appropriate decomposition into function/procedures
- appropriate data abstraction
- appropriate use of comments
- appropriate use of language constructs

A submission can be penalized up to 15% for failures of the above.