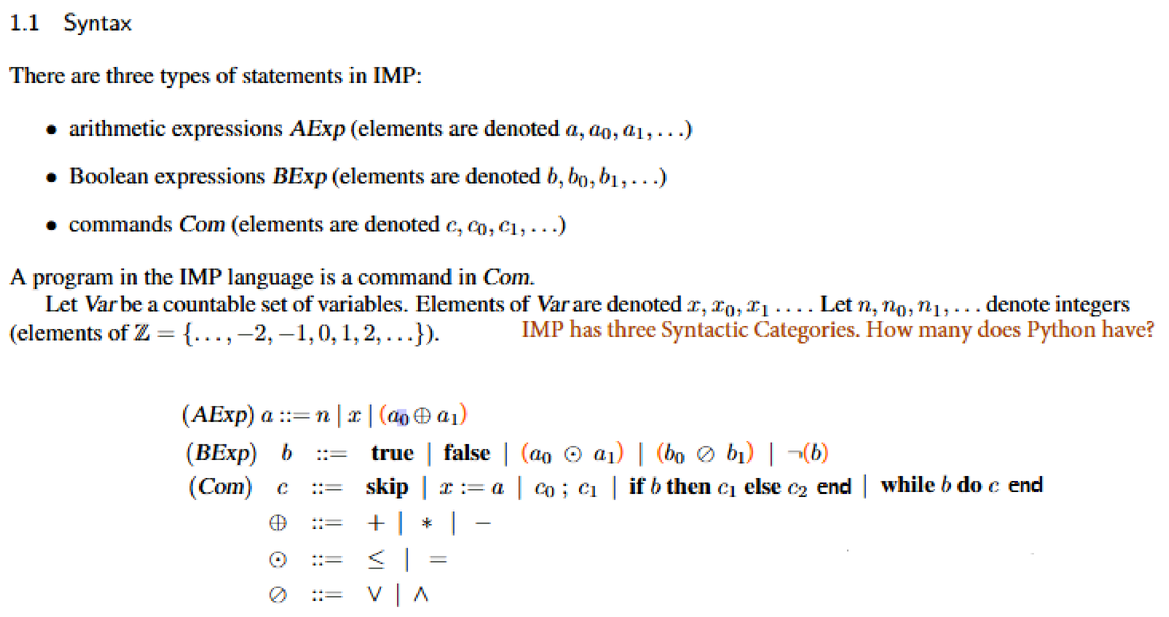
290I Assignment 5 Due Nov 1 11:59 pm

Please submit all your files in a zip file.

Note: While the provided skeleton code is in Python, you can use any language you prefer. If you choose a different language, you must implement the Lexer and Parser yourself or port the provided logic. Just make sure you use a similar AST structure.

Problem 1: IMP Interpreter

In this problem, you will implement the core execution logic for the IMP language. The syntax is as follows (please refer to the class note):



1.1 Syntax

There are three types of statements in IMP:

- arithmetic expressions *AExp* (elements are denoted $a, a_0, a_1, \ldots$)
- Boolean expressions *BExp* (elements are denoted $b, b_0, b_1, \ldots$)
- commands *Com* (elements are denoted $c, c_0, c_1, \ldots$)

A program in the IMP language is a command in *Com*.
Let *Var* be a countable set of variables. Elements of *Var* are denoted $x, x_0, x_1 \ldots$. Let $n, n_0, n_1, \ldots$ denote integers (elements of $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$). IMP has three Syntactic Categories. How many does Python have?

$$\begin{array}{lcl}
(AExp)\ a & ::= & n \mid x \mid (a_0 \oplus a_1) \\
(BExp)\ b & ::= & \textbf{true} \mid \textbf{false} \mid (a_0 \odot a_1) \mid (b_0 \oslash b_1) \mid \neg(b) \\
(Com)\ c & ::= & \textbf{skip} \mid x := a \mid c_0 ; c_1 \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ end} \mid \textbf{while } b \textbf{ do } c \textbf{ end} \\
\oplus & ::= & + \mid * \mid - \\
\odot & ::= & \leq \mid = \\
\oslash & ::= & \vee \mid \wedge
\end{array}$$

We have provided the two components:

- `lexer.py`: Converts the input source code into a sequence of tokens.
- `parser.py`: consumes those tokens to construct an Abstract Syntax Tree (AST).

The AST structure is defined in the `nodes.py` file. While the semantics of the arithmetic and boolean logic nodes are already implemented, the nodes responsible for state management and control flow are incomplete.

Refer to the class notes and complete the **`eval`** functions for the following classes (marked by "TODO") in **`nodes.py.`** :

- `NodeIdentifier`
- `NodeAssign`
- `NodeSequence`
- `NodeIf`
- `NodeWhile`

Deliverable 1:

Submit all source code files.

Note: Please submit the full package so your code can be verified easily. Missing files may result in a point deduction.

Problem 2: Fibonacci Program in IMP

Write a program that computes the nth Fibonacci number using IMP language. Put your program in the `fibonacci.py` and test with n = 10. Your program should be a sequence of characters (string)

Deliverable 2:

The completed `fibonacci.py` file.