Project 1 Tips and Tricks

Here are some hints/guidelines that you might find useful when doing Project 1.

Notes:

- It's not needed to follow these guidelines -- there are many possible different implementations. These are just in case you get stuck and are unsure on what to do next.
- we won't assign keywords to different values in our testing (e.g. we won't do `assign True False`); you can consider this undefined behavior
- you shouldn't be concerned about Python's recursion limit, if you chose to use recursion (i.e. the default limit of 1000 calls should be fine)
- This project is expected to take ~15-20 hours to complete.

Optional tips/tricks:

- you'll probably need to implement some form of tokenization: this includes splitting on spaces, getting rid of comments, and turning capturing quoted areas as strings
    - you can either tokenize the entire program and then parse, or tokenize each line
- for variables, we want a mapping from name -> value (e.g. python dictionary)
- for function calls (e.g. `funccall foo`), we want to:
    - go to the start of the function definition
    - run the statements till the end of the function
    - end up at the next instruction after `funccall foo`
- suggestion: try and split up implementation into as many functions as possible
    - many Brewin keywords require similar implementations -- abstracting the similarities into a common function will reduce duplicate code and make debugging easier
- indentation is similar to Python indentation; this requires keeping track of the number of left-spaces, in some way
    - suggestion: use this for handling if/else statements, function definitions, and while statements
    - otherwise, you'll struggle with indentation on nested if-statements

- Some parts of Brewin require us to know the type of a value (Int, Bool, or String). You'll need some way to determine this
- input and print are remarkably similar 👀
- One way to handle expressions is through a stack. Brewin uses prefix notation
- note operators can be overloaded (e.g. `+` for both integer addition and string concatenation)
- valid syntax includes empty functions, return statements not at the end of function definitions, and functions without return statements