Assignment 7 - Program Slicing

In class, we worked with a program that generates a control flow graph (CFG) for a limited subset of Python. For this assignment, transform that program into a program slicer.

Your slicer should be able to handle:

- Straight-line programs
- The if then conditionals we added during class. **Remember that the AST→CFG transformation program that's available to you on the website doesn't handle if-then conditionals! You have to extend it to handle if-then conditionals! (If you completed this activity in class, you're good to go!)**
- While loops

The template code supports this usage:
`python slicer_template.py filename line_number variable_name`

For the autograder to work, the output of `Slicer`'s `slice` method should be the line numbers of all lines included in the slice, ordered from lowest to highest. **Use the line numbers that the `ast` library includes in the AST node objects that it creates.**

Examples

For example, for the following program:

```
x = 40
y = 50
if (x > 2):
   x = 2
else:
   y = 2
print(x)
```

`python slicer.py filename 6 x`
should yield: [1]
`python slicer.py filename 7 x`
should yield: [1, 3, 4]

Tips and Tricks
You are *not* required to follow any of the tips and tricks on the next page. Feel free to structure your program slicer however you prefer (as long as it still plays nice with the autograder)! But if you're feeling stuck, take a look at the next page for some ideas that might help you find next steps.

Tips, tricks, and thought-provoking questions!

In addition to a top-level slice function like
```
slice(self, filename, linenum, varname)
```
You might want a recursive helper like:
```
slice_helper(self, node, relevant_vars, nodes_in_slice)
```

Or perhaps something similar.

What's the right point in the program to figure out the control set? Is it when we're building the CFG? Is it when we're traversing the CFG?

You'll probably want some helper functions for figuring out what's defined at a given node and what's referenced at a given node. For figuring out what variables are referenced at a given node, you'll need to traverse an AST subtree. Maybe the helper function will look something like this:
```
def referenced_at_node_helper(ast_node):
    if (type(ast_node) == ast.Constant):
        return []
    elif (type(ast_node) == ast.BinOp):
        return referenced_at_node_helper(ast_node.left) +
               referenced_at_node_helper(ast_node.right)
    ....
```
What other node types do we need to handle?