Preston Bennett

Dr. Vamsi Paruchuri

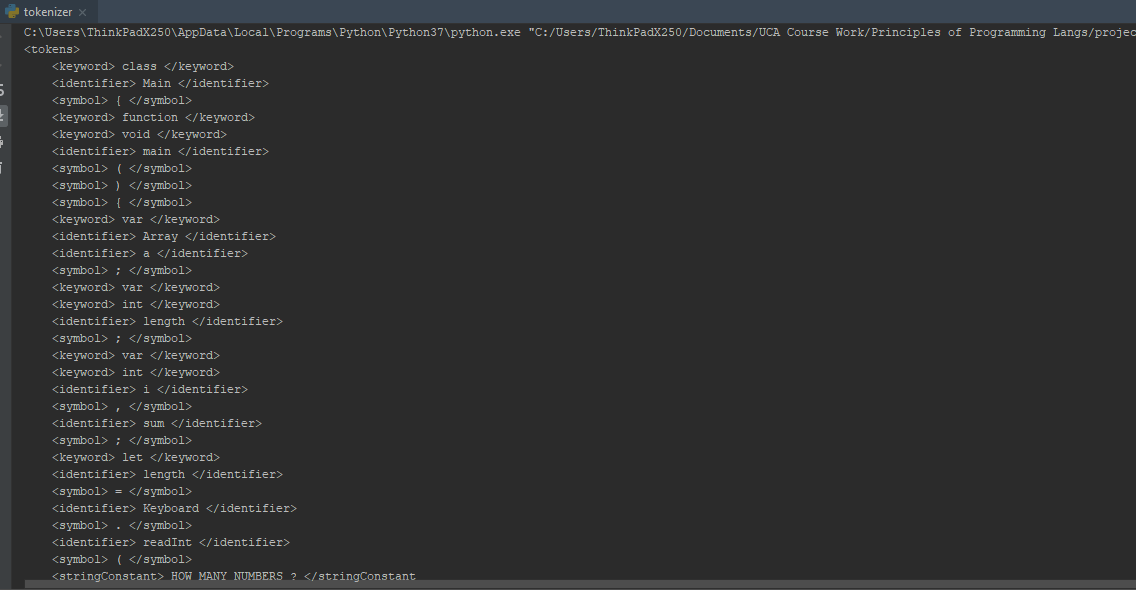
CSCI 3370

10/4/19

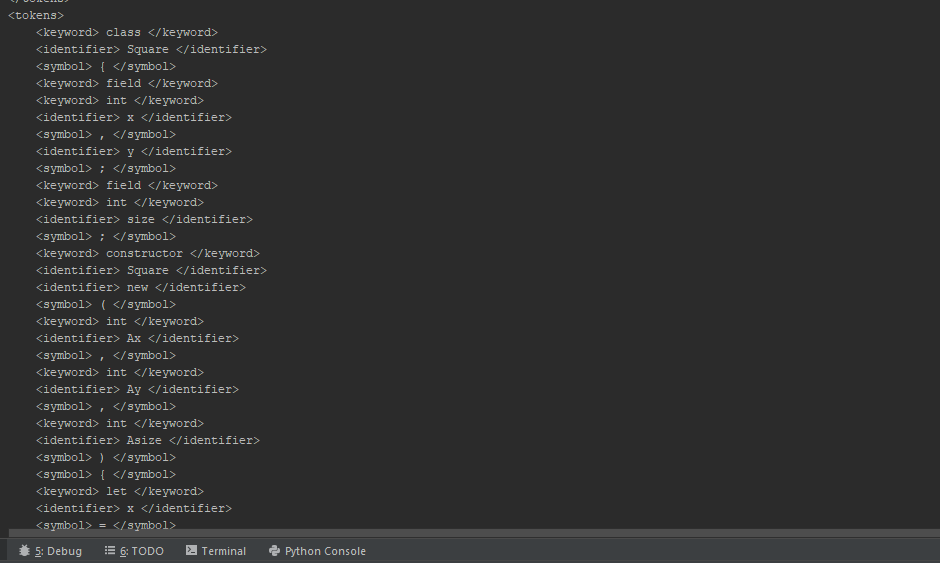
Lab 1: Jack Tokenizer

Screenshots of output:

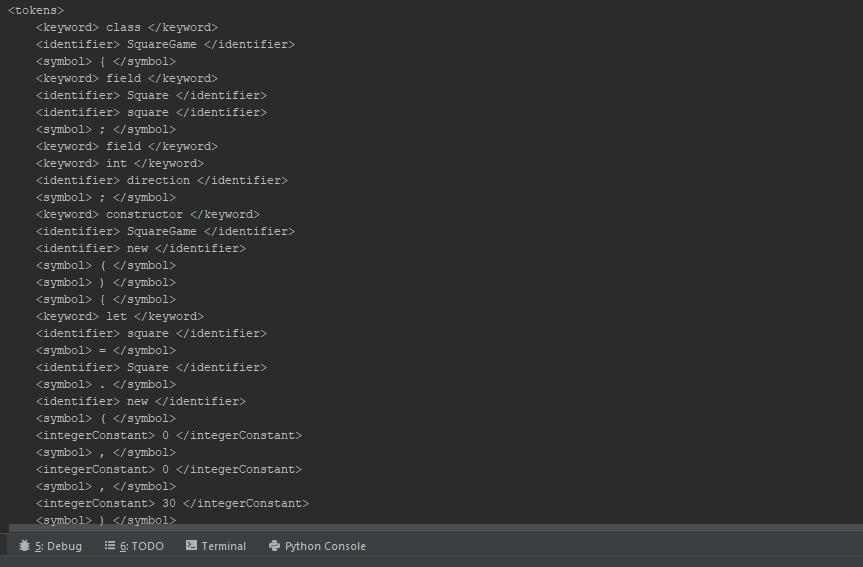
Output for processing Main.jack:



Output for processing Square.jack:



Output for processing SquareGame.jack:



Approach:

While programming in python, I split the tasks of the program into four: opening files, extracting text, removing comments, then tokenizing the text. I used a lot of regular expressions and python’s re module to be able to find and remove patterns out of the source file. This not only helped clearing comments first and foremost but made it easier to split each line into a list of tokens that can each be analyzed for matching type. Two control flows were implemented in the tokenize method: one for when the token has been discovered to be a string constant and one for everything else. Once that is done, I insert the created xml body into a new file, ending my tokenizer.

Challenges:

There were three main challenges I faced while programming and testing. The first was having to relearn Python as Python isn’t one of those languages I use most of the time. The second was figuring out how to remove the comments. I knew that tokenizing line by line with comments in the way would lead to confusing programming, so I had to get those out of the way first. Regular expressions helped make this a possibility. The third and last challenge was figuring out how to deal with string constants. Since I use lists of tokens for each line, those tokens with multiple words that begin and end with quotations had to be addressed in the beginning of my main loop with a separate control flow.

Key Take-aways:

From this lab I’ve discovered two take-aways: one is that regular expressions are seriously powerful tools that can get a programmer out of a troubling situation fast when it comes to processing text. I couldn’t imagine writing a tokenizer or any other text processing program without it. The second take-away was that a lot of languages, including jack, are easy to understand from the ground up. The syntax says a lot about what the semantics could mean, and the mark of a good language is syntax that does just that.