Transpiling Flex code

1. Read a line
    a. It could be composed of multiple statements separated by a semicolon, so split it at the semicolons to get individual statements. Add them all to the *statement queue*.
2. Process a statement
    a. If the statement conforms to any one of the language's regular expressions, parse it and extract entities from it, storing them as a dictionary in the *parsed statement list*. For example, the statement,

    ```
    SomeFunction(arg1, arg2, arg3)
    ```

    Will match the regular expression for the beginning of a function definition (since it's not indented, it cannot be a function call), and will be stored in the form:

    ```
    {
      'intent': 'function_def',
      'entities': {
        'name': 'SomeFunction',
        'args': [
          {
            'name': 'arg1'
          },
          {
            'name': 'arg2'
          },
          {
            'name': 'arg3'
          }
        ]
      }
    }
    ```

    b. Otherwise, pass it as a query to the Rasa server running locally.
        i. The server will parse the statement and will return a response in JSON format. As an example, the response returned after parsing the statement,

        ```
        for each vertex v in graph.vertices
        ```

        would look like:

        ```
        {
          'intent': 'begin_loop',
          'entities' : {
            'loop_type': [{
              'value': 'for_each',
              'confidence': 0.993
            }],
            'loop_over': [{
        ```

```
      'value': 'graph.vertices',
      'confidence': 0.984
    }],
    'loop_as': [{
      'value': 'v',
      'confidence': 0.975
    }]
  }
}
```

ii. If all entities have a confidence above a minimum threshold (say 90%), reduce the above response and store it in a dictionary in the *parsed statement list*, as in step 2.a. Otherwise, report an error at that line.

3. Parse each entry (line) from the *parsed statement list*, keeping track of the block and the scope of the variables, and convert it to the target language. If a required value is missing, report an error at the line `l+1`, where `l` is the index of the line in the *parsed statement list*.