Developments in mining QC repos from GitHub

1. The paper suggests we *search_code* using the query *“from qiskit import”.* However, I find it a bit misleading since some repos (maybe built by professional developers) refer to specific modules to import. For example, *“from qiskit.operators import blah blah”*.

   To overcome this incorrect search issue, the use of regex is suitable. I did something like

   repos_by_sc = g.search_code(r“from qiskit(.?)”)* *(.*?) is greedy*
   repo_by_sc.get_page(0)
   The above line is added to avoid totalCount from maxing out (it is a bug!)
   print('total: ' + str(repo_by_sc.totalCount))

2. The paper suggests searching for *Q#* repos using a *query = “language: Q#”*. This query works and gives a *totalCount* of 242 repos. However, I do not see Q# being recognized as a language by GitHub. Moreover, most of the Q# libraries are recognized to use C# language (Q# is based on C#).

   As a result, I think we will have to change this search technique.

3. The issue with searching source code for cirq is that it is a very common word that appears in C language compilers and in C++ repositories as a short form for circle queue.

   It was very surprising for me to find that there are more cirq repositories than qiskit. But, hey, I could be wrong. Anyways, we will have to make sure we are checking the correct repositories and have the correct *totalCount*.

   One way of making sure could be that we search for *r”from cirq(.*?)”* in repositories that have Python and Jupyter notebook dependencies.

4. OpenQASM is an assembly language for quantum libraries like qiskit. We might also have to look for repositories with program files that end with *.qasm*.