Open-Source Contribution: QisKit

Languages and Frameworks:

I have some experience in four programming languages. On a scale of 1 (ability to write a basic "Hello World" program) to 10 (mastery of the language); I would rate my language skills in the following manner:

- JavaScript: 1.5 (No experience in any specific frameworks)
- Java: 2
- C/C++: 4
- Python: 7

I'm confident that my Java and JavaScript abilities would not allow me to positively contribute to any project in those languages, ruling those options out. C/C++ is an option, but very little in that language comes naturally to me, and I still have much to learn about writing code that is both secure and has high-performance. I therefore defaulted to looking for python projects, and thankfully, there were many to be found.

Project Search:

Based on the guides provided on the assignment page, I focused my search for large projects with issues tagged with "good first issue". One thing I learned in my search is that not all project teams utilize this tag in the way that I expected. Some teams used it to identify issues that could be tackled effectively by new contributors, but several projects seemed to use the tag on issues that the core team simply didn't want to deal with. One that stood out was a project with an issue for doubling the code coverage of the entire testing suite tagged as a "good first issue." I quickly moved on to look at other projects. In the end, I settled on a project called QisKit.

The Project:

QisKit, per their readme, "is an open-source SDK for working with quantum computers at the level of extended quantum circuits, operators, and primitives." Quantum computing is in the running for the next big thing in computer science, but it is still relatively early for this technology and society has yet to really benefit from this technology. Their documentation suggests that this toolkit is primarily designed for learners (they additionally support their own quantum computing course) and for use by researchers. This project has also partnered with IBM's quantum efforts, allowing users to connect to actual quantum computers with an appropriate API key. This library had its first public release at the end of 2018 and as of my fork it has had a total of 7805 commits. This project has been forked over 2,000 times and currently has 3,800 stars. I would also classify this as a large project; per GitHub insights, over the last month almost 300 files have been edited and 55,000 lines of code have been added or changed.

Best Practices:

In general, I would argue that best practices for new contributors can be divided into two categories: learning the code base and learning the contribution process for that specific project. It is vital to understand how the pieces of code that we might contribute to interact with the rest of the project. This is true from both a functional standpoint and a stylistic one. Additionally, the project that we contribute to likely has an established process for contributing to the repo, with

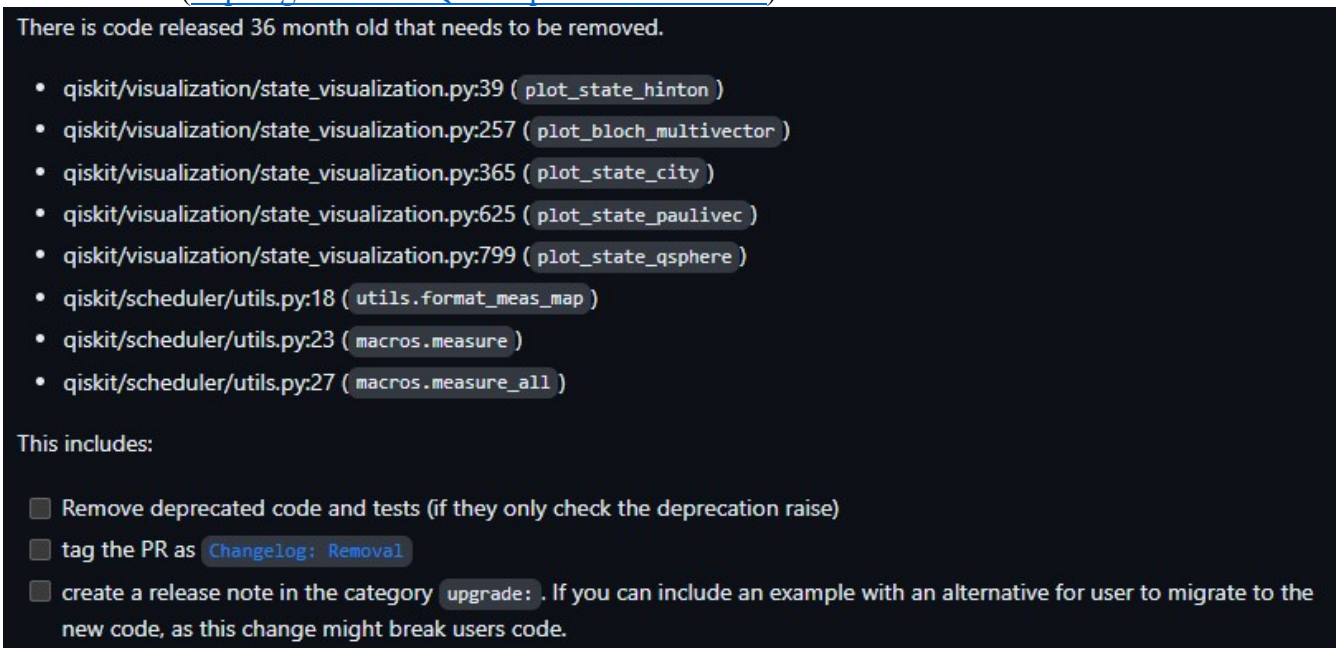
many procedures in place to (hopefully) prevent any contributor from breaking production or negatively impacting the project in other ways. Learning these two areas are essential starting points for any new project contributor.

For the project itself, QisKit has a code of conduct which primarily covers interactions between contributors and was about what was a fairly standard document. The project also has a contribution guide for the overall project and one for the specific repo that I was looking to contribute to. These cover everything from a pull request checklist to how to build release notes. The guidelines for the specific repo that I am looking to work in (QisKit-Terra) also extensively covers testing procedures. This repo does not state its preferred code style, but it does utilize automatic style and lint checking as part of the testing suite. One additional important section of the guidelines was how contributors should identify and deal with depreciated code, which was of particular interest to me given the issue that I decided to tackle.

The Issue:

The issue that I have chosen to address deals with the removal of depreciated code in the visualization portion of the codebase. The main technology that I will be interacting with is python and the testing suite. This project utilizes the tox suite, which unfortunately is a testing framework that I am unfamiliar with. In an ideal situation, I estimate that this change will likely take between two and five hours based on the descriptions in the issue and the additional information available in the changelog, with most of this time spent learning and utilizing the testing framework. A potentially significant issue that may arise is that I have no background knowledge on quantum computing and before diving into the code base I can't tell how much of a detrement this might be. To remedy this deficiency, I intend to spend 5-10 hours going through QisKit's provided tutorials in an attempt to provide some context for what the code is doing before attempting to fix the issue. Depending on the code that needs to be changed, this does have the potential to be insufficient and I will reevaluate once I look at the code.

GitHub Issue (https://github.com/Qiskit/qiskit/issues/10742):



There is code released 36 month old that needs to be removed.

- qiskit/visualization/state_visualization.py:39 (`plot_state_hinton`)
- qiskit/visualization/state_visualization.py:257 (`plot_bloch_multivector`)
- qiskit/visualization/state_visualization.py:365 (`plot_state_city`)
- qiskit/visualization/state_visualization.py:625 (`plot_state_paulivec`)
- qiskit/visualization/state_visualization.py:799 (`plot_state_qsphere`)
- qiskit/scheduler/utils.py:18 (`utils.format_meas_map`)
- qiskit/scheduler/utils.py:23 (`macros.measure`)
- qiskit/scheduler/utils.py:27 (`macros.measure_all`)

This includes:

- [ ] Remove deprecated code and tests (if they only check the deprecation raise)
- [ ] tag the PR as `Changelog: Removal`
- [ ] create a release note in the category `upgrade:`. If you can include an example with an alternative for user to migrate to the new code, as this change might break users code.

Changelog description:

The name of the first positional parameter for the `qiskit.visualization` functions `plot_state_hinton()`, `plot_bloch_multivector()`, `plot_state_city()`, `plot_state_paulivec()`, and `plot_state_qsphere()` has been renamed from `rho` to `state`. Passing in the value by name to `rho` is deprecated and will be removed in a future release. Instead you should either pass the argument positionally or use the new parameter name `state`.

Part 2:
Video link: https://youtu.be/T8-vmR55cW8

Bernstein-Vazirani tutorial: https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/Coding_With_Qiskit/ep6_Bernstein-Vazirani_Algorithm.ipynb

Pull request link: https://github.com/Qiskit/qiskit/pull/10779

Links to files I updated:
Visualizations:
-Before:
https://github.com/Qiskit/qiskit/blob/main/qiskit/visualization/state_visualization.py
-After:
https://github.com/unit0113/qiskit/blob/fixissue10742branch/qiskit/visualization/state_visualization.py

Other functions:
-Before: https://github.com/Qiskit/qiskit/blob/main/qiskit/scheduler/utils.py
-After: Deleted