Quantum Tools

Miguel Sozinho Ramalho

November, 2018

Table of contents

Introduction

This week is all about getting to know the **tools** at our disposal. Some previous knowledge is required of the Python programming language, but other than that we will mention what matters the most about the tools in this course. Deeper and wider knowledge of them will arise with use!

Python

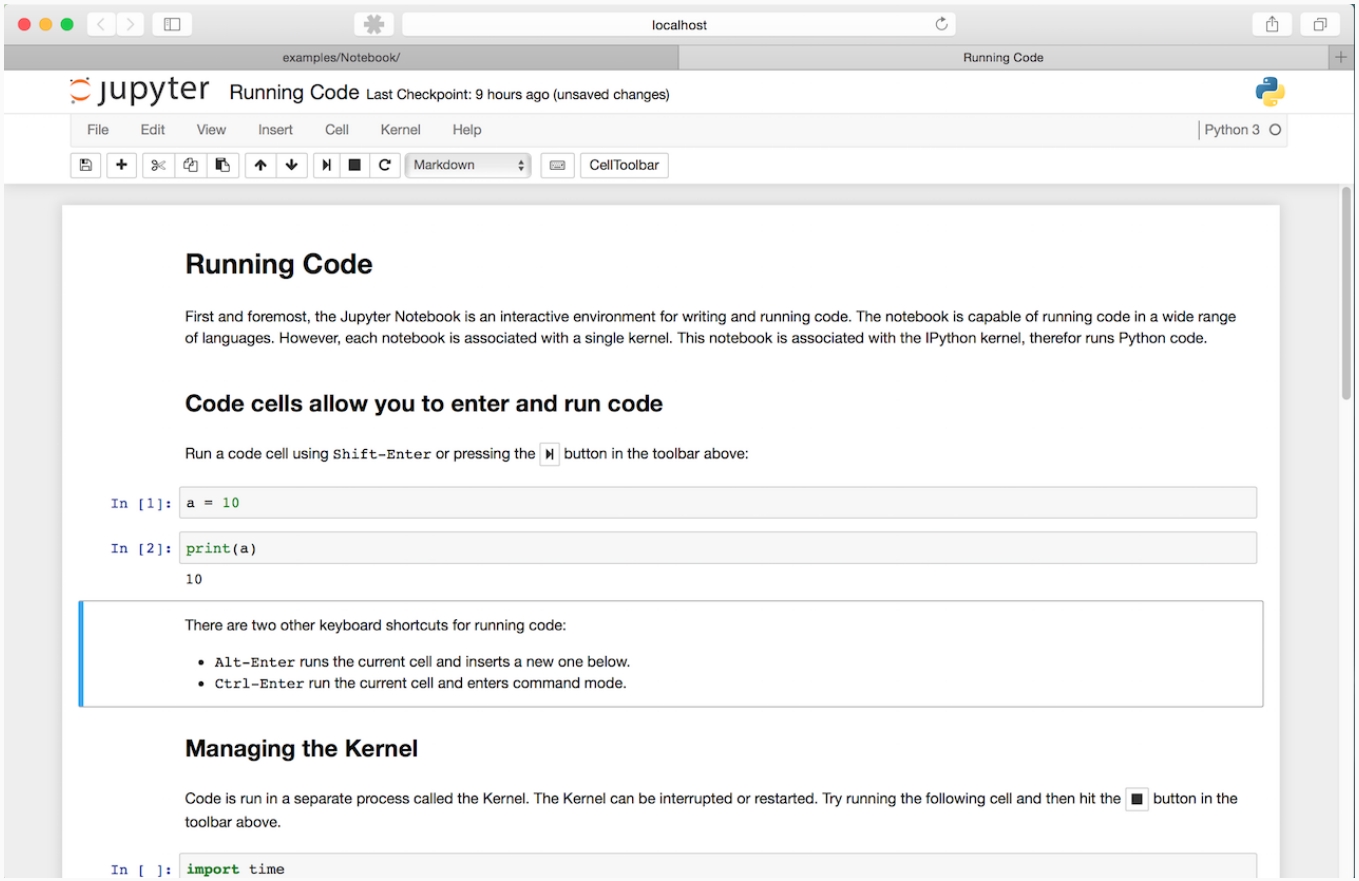
Python is a high-level scripting language that is widely used for prototyping due to its simplicity and abstractions. Most importantly, it has a huge community that builds and shares tools and frameworks that are easily used by Python users. Most of these **packages** are available at PyPi and can be installed easily through `pip install PACKAGE_NAME`.

Jupyter

Project Jupyter has produced an amazing tool, the Jupyter Notebooks: a web-based environment that can be used to write code as if it were a book, using markdown, executing code block by block, and viewing the output next to the code that produced it. This makes it easy to document, understand and share code.

We will be using these notebooks for most of the practical exercises.

Installing Jupyter Notebooks



There are two options for installing Jupyter Notebooks in your computer, both properly described here. After that, head to a console and do `jupyter notebook`.

Google Colaboratory

Although it is still bellow critical mass (for now), there is an online version of Jupyter Notebooks that allows for collaborative and real-time edition of your notebooks (much like Google Docs): Google Colaboratory!
Even if it is not directly used in this course, you may find it useful to work with others and, at times, to use Jupyter Notebooks in computers without a local installation of Jupyter Notebooks.

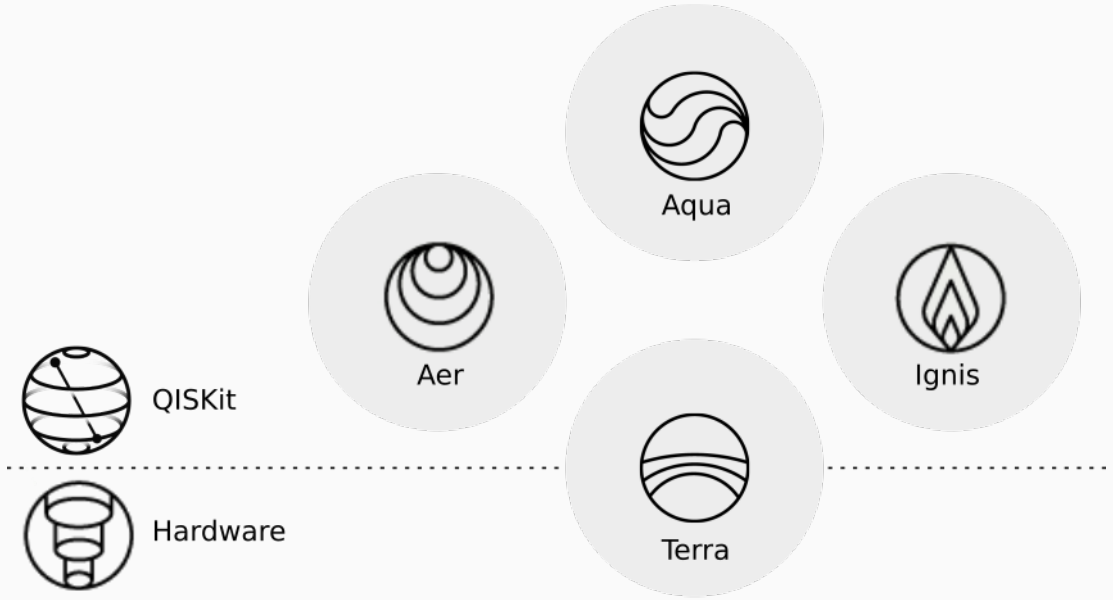
Binder

Much like Colaboratory, Binder is a fantastic tool to interact with Jupyter Notebooks online and without any previous setup. This may come in useful in the future. However it is best suited for Notebooks hosted in GitHub, which is not a problem in this course.
Binder is available at mybinder.org!

QISKit

Quoting, QISKit is 'an open-source quantum computing framework for leveraging today's quantum processors and conducting research'. It can be used through a python interface (our focus) and it is, no doubt, **the most important tool of this course**. (You read it like biscuit, but with a Q!)

The QISKit framework



The QISKit framework is subdivided into four tools:

- QISKit Terra (`pip install qiskit`)
- QISKit Ignis (part of qiskit package)
- QISKit Aer (part of qiskit package)
- QISKit Aqua (`pip install qiskit-aqua`)

The QISKit framework

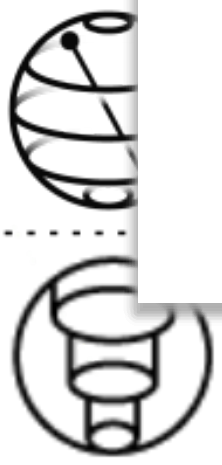
QISKit can be seen as a **Full Stack Quantum Framework**. It allows developers and researchers to both get 'their hands dirty' on low-level quantum gates (Terra) and also to work with all minutiae abstracted, as if it were a classical program (Aqua).

We will mainly focus on QISKit Terra. The following four *fundamental slides* are based on this post and provide the base understanding of each of these elements.

Installation Tip: If `pip install` fails, try `pip3 install`. To change this behaviour on Linux, visit this post.

QISKit Terra

'Terra, the *earth* element, is the foundation on which the rest of the software lies. Terra provides a bedrock for composing quantum programs at the level of circuits and pulses, to optimize them for the constraints of a particular device, and to manage the execution of batches of experiments on remote-access devices. Terra defines the interfaces for a desirable end-user experience, as well as the efficient handling of layers of optimization, pulse scheduling and backend communication.'



Hardware

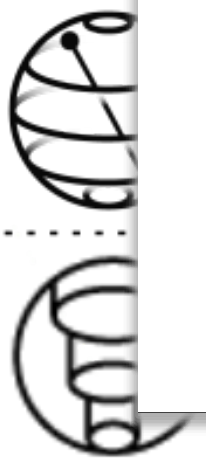
Terra

QISKit Aqua

'Aqua, the *water* element, is the element of life. To make quantum computing live up to its expectations, we need to find real-world applications. Aqua is where algorithms for NISQ (Noisy Intermediate-Scale Quantum) computers are built. These algorithms can be used to build applications for quantum computing. Aqua is accessible to domain experts in chemistry, optimization or AI, who want to explore the benefits of using quantum computers as accelerators for specific computational tasks, without needing to worry about how to translate the problem into the language of quantum machines.'

QISKit Ignis

'Ignis, the *fire* element, is dedicated to fighting noise and errors and to forging a new path. This includes better characterization of errors, improving gates, and computing in the presence of noise. Ignis is meant for those who want to design quantum error correction codes, or who wish to study ways to characterize errors through methods such as tomography, or even to find a better way for using gates by exploring dynamical decoupling and optimal control. While we have already released parts of this element as part of libraries in Terra, an official stand-alone release will come soon.'

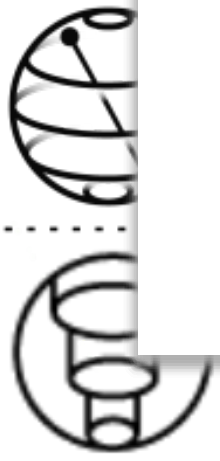


QISKit Aer

'Aer, the *air* element, permeates all Qiskit elements. To really speed up development of quantum computers we need better simulators, emulators and debuggers. At IBM Q, we have built high-quality, high-performance simulators and continue to improve their scalability and features. Aer will help us understand the limits of classical processors by demonstrating to what extent they can mimic quantum computation. Furthermore, we can use Aer to verify that current and near-future quantum computers function correctly. This can be done by stretching the limits of simulation to accommodate 50+ qubits with reasonably high depth, and by simulating the effects of realistic noise on the computation.'

QISKit Wrap-up

Each piece fits into the puzzle, and although you may end up working on high-level quantum code, in this course we will walk through all the low-level bases that will shed some more light into the Quantum beauty!
Many new and potentially unknown terms were used in the definitions of these fundamental elements of QISKit, in time some will become clearer and you will begin to see real implementation of these tools in your work. So don't worry if you feel lost, it is just to give you a glimpse of the Quantum World!



Community

QISKit brings along a growing community, you can interact with it through Slack, GitHub, Medium, Facebook and more!

VSCode extension

VSCode is an open-source code editor that is both light and extremely versatile, being extremely customisable to the code you need to write. Recently, a new extension for QISKit has been created which eases the whole process of developing for QISKit. Check it on GitHub, as it may prove useful for this course and also for your future work.

IBM Q Experience

Finally, on the list of main tools for this course, IBM Q Experience. This is a manifestation of all the more sidereal nature of QISKit into the palpable world. IBM Q represents IBM's efforts of making Quantum Computing have an impact on industry and science.

IBM Q Experience means IBM's Quantum Computers are accessible to the every day developer. To us, this means writing some quantum circuits and executing, not only in a simulation environment, but on one of the real Quantum Computers that IBM has. All you have to do is create an account and you are ready design and execute your circuits!

Exercises

For this week, there are some boring tasks:

- Install Python (and pip)
- Install Jupyter
- Install QISKit
- Create an account on IBM Q Experience

As well as some interesting tasks:

- Checkout the community for QISKit
- Do this week's exercises (local setup of IBMQ account)

Where to learn more?

- Qiskit and its Fundamental Elements
- Quantum Computing in the NISQ era and beyond, John Preskill
- Python tutorial (plenty other tutorials out there)
- Jupyter tutorial