

# Y390: A Lab Manual

Adrian German

July 1, 2019

## Abstract

In which we provide a record of the 6W2 Summer 2019 work with Andrew Weissman, Lukas Cavar and Yifan Chen. We start with the Moran book. The format is temporary and subject to change soon.

## 1 Jupyter Notebooks

Log into silo and create a folder then bring the code<sup>1</sup> in:

```
-bash-4.2$ pwd
/u/dgerman
-bash-4.2$ mkdir qc-6w2-2019
-bash-4.2$ cd qc-6w2-2019/
-bash-4.2$ pwd
/u/dgerman/qc-6w2-2019
-bash-4.2$ ls
-bash-4.2$ git clone \
> https://github.com/PacktPublishing/Mastering-Quantum-Computing-with-IBM-QX
Cloning into 'Mastering-Quantum-Computing-with-IBM-QX'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 180 (delta 12), reused 21 (delta 12), pack-reused 159
Receiving objects: 100% (180/180), 1.33 MiB | 0 bytes/s, done.
Resolving deltas: 100% (71/71), done.
-bash-4.2$ ls
Mastering-Quantum-Computing-with-IBM-QX
-bash-4.2$
```

---

<sup>1</sup>Careful with the backslash, it's meant to be followed immediately by a return, or you can type everything on the same line, if you prefer.

Let's see now what we got:

```
-bash-4.2$ pwd
/u/dgerman/qc-6w2-2019
-bash-4.2$ ls
Mastering-Quantum-Computing-with-IBM-QX
-bash-4.2$ ls -l Mastering-Quantum-Computing-with-IBM-QX/
total 16
drwxr-xr-x 2 dgerman faculty 77 Jun 26 19:00 Chapter01
drwxr-xr-x 2 dgerman faculty 43 Jun 26 19:00 Chapter02
[...]
drwxr-xr-x 2 dgerman faculty 43 Jun 26 19:00 Chapter13
-rw-r--r-- 1 dgerman faculty 1062 Jun 26 19:00 LICENSE
-rw-r--r-- 1 dgerman faculty 5450 Jun 26 19:00 README.md
-rw-r--r-- 1 dgerman faculty 59 Jun 26 19:00 requirements.txt
-bash-4.2$
```

Let's see how we can switch<sup>2</sup> to Python 3 and set up a virtual environment:

```
-bash-4.2$ pwd
/u/dgerman/qc-6w2-2019
-bash-4.2$ python
Python 2.7.5 (default, Jun 11 2019, 12:19:05)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
>>> exit()
-bash-4.2$ module load python/3
-bash-4.2$ python
Python 3.6.6 (default, Aug 27 2018, 10:45:14)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
>>> exit()
-bash-4.2$ module list
Currently Loaded Modulefiles:
  1) soic.use.own  2) soic  3) mpi/mpich-x86_64  4) python/3
-bash-4.2$ module unload python/3
-bash-4.2$ module list
Currently Loaded Modulefiles:
  1) soic.use.own  2) soic  3) mpi/mpich-x86_64
-bash-4.2$ python
Python 2.7.5 (default, Jun 11 2019, 12:19:05)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
>>> exit()
-bash-4.2$
```

---

<sup>2</sup>This is a property of the session not of the folder we're in. Do it after you log in and that's what you will use for the remaining of your session (until you log out).

Having seen how you can switch between the versions of Python that you want to use let's point out you can load a specific version (in the example I loaded the most recent Python 3 available on the system, in this case 3.6.6; the book mentions 3.4). To create and activate a virtual environment you need to first choose a location for it. It does not have to be anywhere near the code you want to work with, but it does have to have a location.

```
-bash-4.2$ pwd
/u/dgerman/qc-6w2-2019
-bash-4.2$ cd ..
-bash-4.2$ python --version
Python 2.7.5
-bash-4.2$ module load python/3
-bash-4.2$ python --version
Python 3.6.6
-bash-4.2$ ls MoranQC
ls: cannot access MoranQC: No such file or directory
-bash-4.2$ python -m venv MoranQC
-bash-4.2$ ls MoranQC
bin  include  lib  lib64  pyvenv.cfg
-bash-4.2$ ls -ld MoranQC
drwxr-xr-x 5 dgerman faculty 93 Jun 26 19:29 MoranQC
-bash-4.2$ cd qc-6w2-2019/
-bash-4.2$ cd Mastering-Quantum-Computing-with-IBM-QX/
-bash-4.2$ cd Chapter01
-bash-4.2$ ls
Hello Quantum World.ipynb  test_installation.py
-bash-4.2$ source ~/MoranQC/bin/activate
(MoranQC) -bash-4.2$
-bash-4.2$
```

So hopefully you were able to follow all that. I put the virtual environment at the top level since it needs to be activated as soon as I log in (along with the switch to Python 3, of course, we can automate that). Now you could try running `python test_installation.py` but if you do that (go ahead and try it!) you will be told that you're missing modules and the following piece of advice will be posted in response:

```
Traceback (most recent call last):
  File "test_installation.py", line 8, in <module>
    raise Exception("make sure to pip install -r requirements.txt")
Exception: make sure to pip install -r requirements.txt
```

So let's move to the right folder and try to follow<sup>3</sup> the advice:

```
(MoranQC) -bash-4.2$ ls
Hello Quantum World.ipynb  test_installation.py
(MoranQC) -bash-4.2$ cd ..
(MoranQC) -bash-4.2$ pwd
/u/dgerman/qc-6w2-2019/Mastering-Quantum-Computing-with-IBM-QX
(MoranQC) -bash-4.2$ ls -l
total 16
drwxr-xr-x 2 dgerman faculty  77 Jun 26 19:00 Chapter01
drwxr-xr-x 2 dgerman faculty  43 Jun 26 19:00 Chapter02
[...]
drwxr-xr-x 2 dgerman faculty  43 Jun 26 19:00 Chapter13
-rw-r--r-- 1 dgerman faculty 1062 Jun 26 19:00 LICENSE
-rw-r--r-- 1 dgerman faculty 5450 Jun 26 19:00 README.md
-rw-r--r-- 1 dgerman faculty  59 Jun 26 19:00 requirements.txt
(MoranQC) -bash-4.2$
(MoranQC) -bash-4.2$ pip install -r requirements.txt
[...]
Installing collected packages: numpy, [...] matplotlib, [...] qiskit [...]
  Running setup.py install for mpmath ... done
  Running setup.py install for psutil ... done
  Running setup.py install for networkx ... done
  Running setup.py install for pycparser ... done
Successfully installed [...] matplotlib-3.0.2 [...] numpy-1.16.0 [...] qiskit-0.7.0 [...]
You are using pip version 10.0.1, however version 19.1.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(MoranQC) -bash-4.2$
(MoranQC) -bash-4.2$ pwd
/u/dgerman/qc-6w2-2019/Mastering-Quantum-Computing-with-IBM-QX
(MoranQC) -bash-4.2$ cd Chapter01
(MoranQC) -bash-4.2$ python test_installation.py
(MoranQC) -bash-4.2$
```

So let's summarize what we've done: we have activated Python 3, created and activated a virtual environment, and then installed a number of packages. These packages are available in this virtual environment regardless of the location of our code. We can test (or illustrate) that as follows: type `python` then `import qiskit` at the prompt. It loads it after a few seconds and then a new prompt offered. Type `exit()` then `deactivate` to get out of the virtual environment and go back into Python and try to import the module again:

---

<sup>3</sup>This is in fact very similar to how `npm install` uses a `package.json` to make sure all project dependencies are satisfied.

this time it will complain that there is no such module available since your installation is local to the `MoranQC` virtual environment<sup>4</sup>. Now we're ready to set up Jupyter as instructed in the book:

```
(MoranQC) -bash-4.2$ pwd
/u/dgerman
(MoranQC) -bash-4.2$ pip install ipykernel
Collecting ipykernel
[...]
Installing collected packages: tornado, ipython-genutils, traitlets, parso, [...]
  Running setup.py install for tornado ... done
  Running setup.py install for backcall ... done
Successfully installed backcall-0.1.0 ipykernel-5.1.1 ipython-7.5.0 ipython-[...]
You are using pip version 10.0.1, however version 19.1.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(MoranQC) -bash-4.2$
(MoranQC) -bash-4.2$ ipython kernel install --user --name=goombah
Installed kernelspec goombah in /nfs/nfs6/home/dgerman/.local/share/jupyter/kernels/goombah
(MoranQC) -bash-4.2$
```

The book uses “bookkernel” where I used “goombah”. I was going to use “abibliophobia” but then I gave up and settled<sup>5</sup> on “goombah” instead. So now we're ready to start the server and please pay attention how this goes:

```
(MoranQC) -bash-4.2$ pwd
/u/dgerman
(MoranQC) -bash-4.2$ cd qc-6w2-2019/
(MoranQC) -bash-4.2$ cd Mastering-Quantum-Computing-with-IBM-QX/
(MoranQC) -bash-4.2$ cd Chapter01
(MoranQC) -bash-4.2$ jupyter notebook --port=27182 --ip=silo.cs.indiana.edu --no-browser
[I 21:35:14.024 NotebookApp] JupyterLab extension loaded from /l/python3.6.6/lib/pytho[...]
[I 21:35:14.024 NotebookApp] JupyterLab application directory is /nfs/nfs5-insecure/ho[...]
[I 21:35:14.026 NotebookApp] Serving notebooks from local directory: /nfs/nfs6/home/dg[...]
[I 21:35:14.026 NotebookApp] The Jupyter Notebook is running at:
[I 21:35:14.026 NotebookApp] http://silo.cs.indiana.edu:27182/?token=7fd4a3f77f58390bb[...]
[I 21:35:14.026 NotebookApp] Use Control-C to stop this server and shut down all kerne[...]
[C 21:35:14.026 NotebookApp]
[P]aste this URL into your browser when you connect first time to login with a token:
http://silo.cs.indiana.edu:27182/?token=7fd4a3f77f58390bb82d2f66b26a5b922233f43d7a2448ee
```

---

<sup>4</sup>To go back into it you need to type `source ~/MoranQC/bin/activate`

<sup>5</sup>[https://www.alphadictionary.com/articles/100\\_funniest\\_words.html](https://www.alphadictionary.com/articles/100_funniest_words.html)

Now start a browser (on your laptop) and type the URL indicated in it. Notice the server you started (via PuTTY I suppose) runs on `sil0` while your browser runs on same machine as the secure shell client. It will look like this:

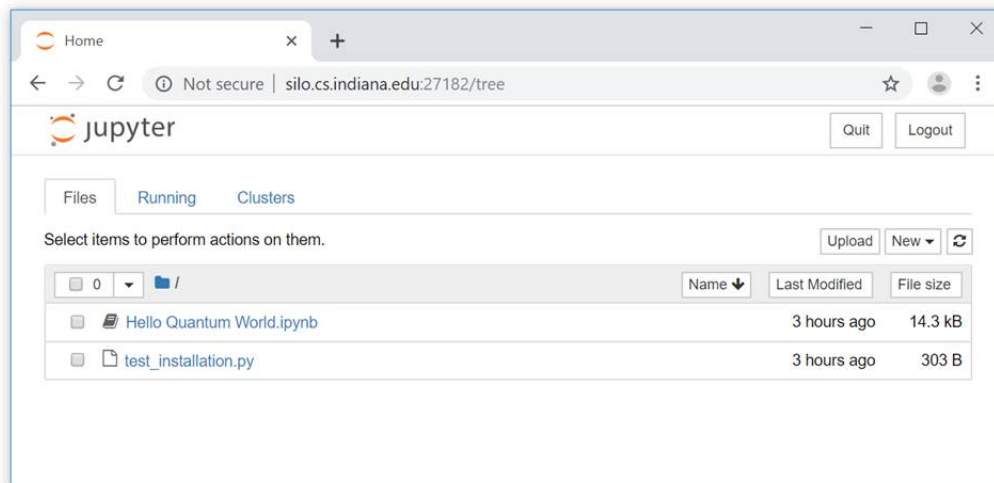


Figure 1: Your notebook is up. Let's create a new file. See the button?

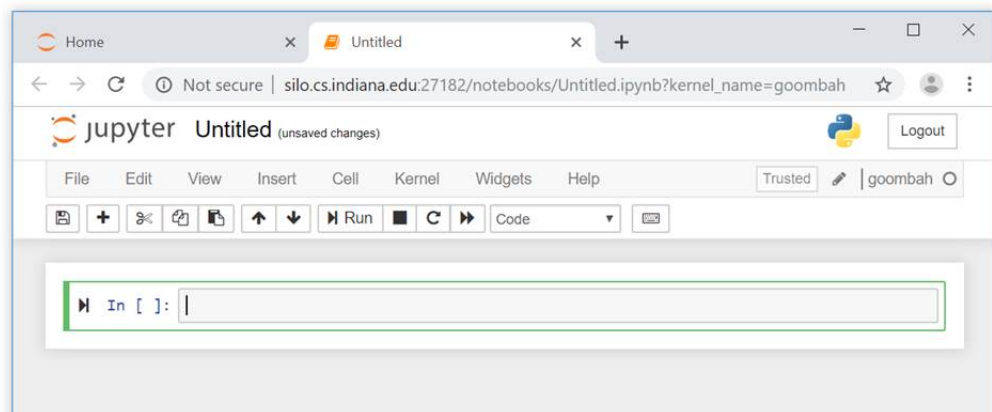


Figure 2: Click **New**, choose `goombah` and here's your new empty notebook. If you look carefully you see you now have a new tab (for the new notebook). Old tab is still there and it shows you created a new file. After you check that and confirm it, go back to your new notebook and type some code from `matplotlib` tutorial or online documentation<sup>6</sup> as shown on the next page.

<sup>6</sup>[https://matplotlib.org/3.1.0/tutorials/introductory/sample\\_plots.html](https://matplotlib.org/3.1.0/tutorials/introductory/sample_plots.html)

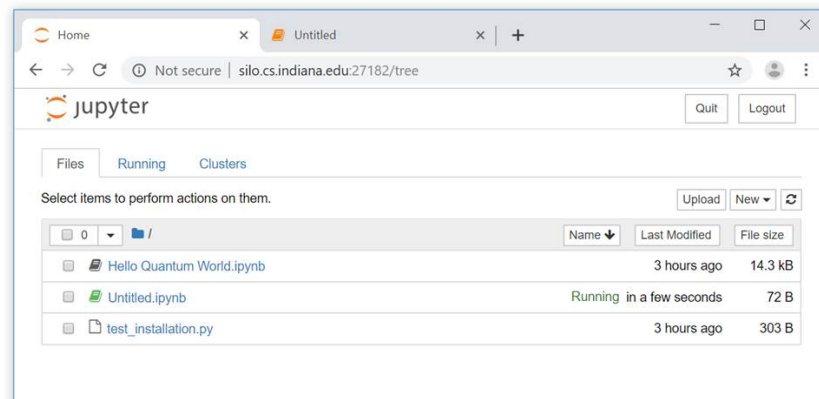


Figure 3: Notice a new file `Untitled.ipynb` has now been created.

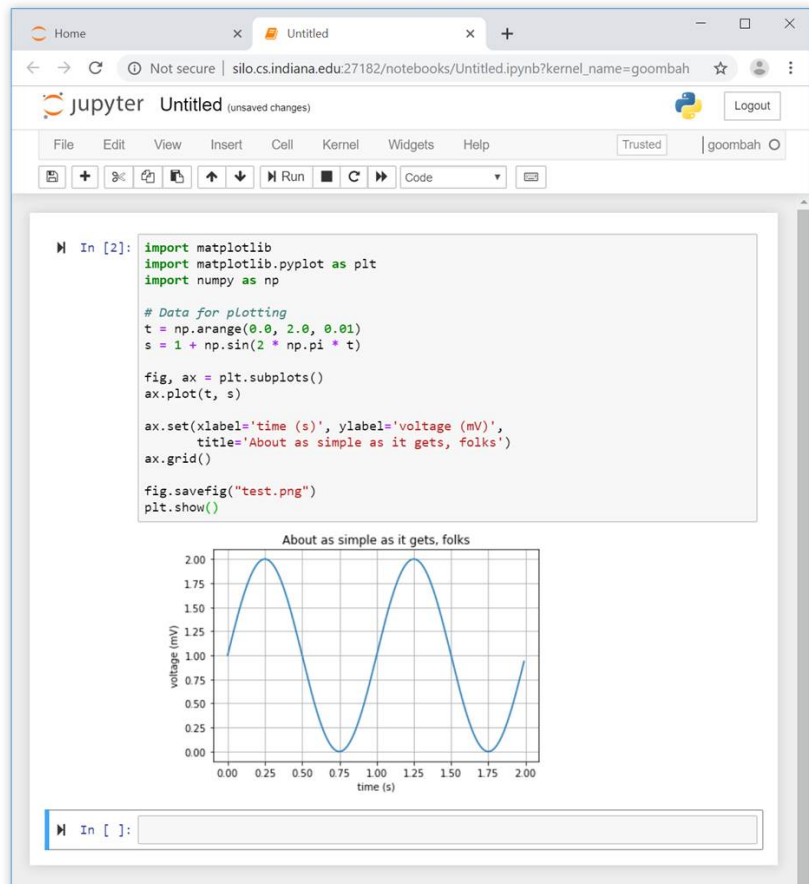


Figure 4: Type some code, press `Run` and enjoy the picture.

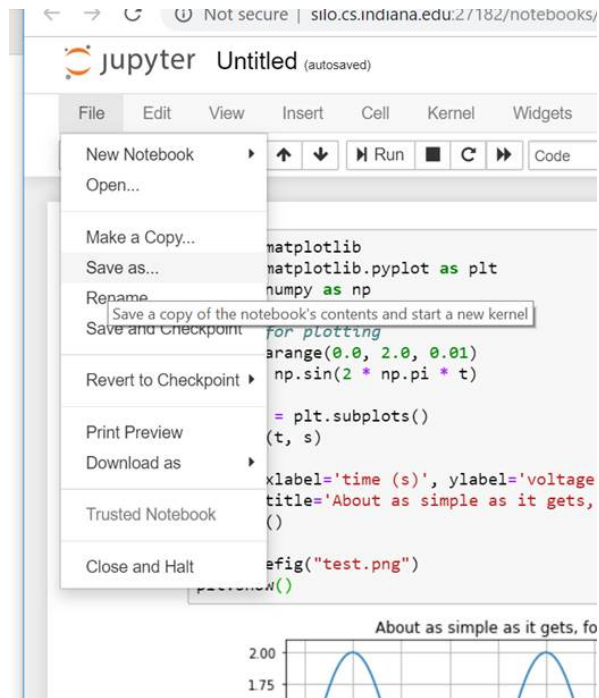


Figure 5: Choose `File` then `Save as...` to save the file.

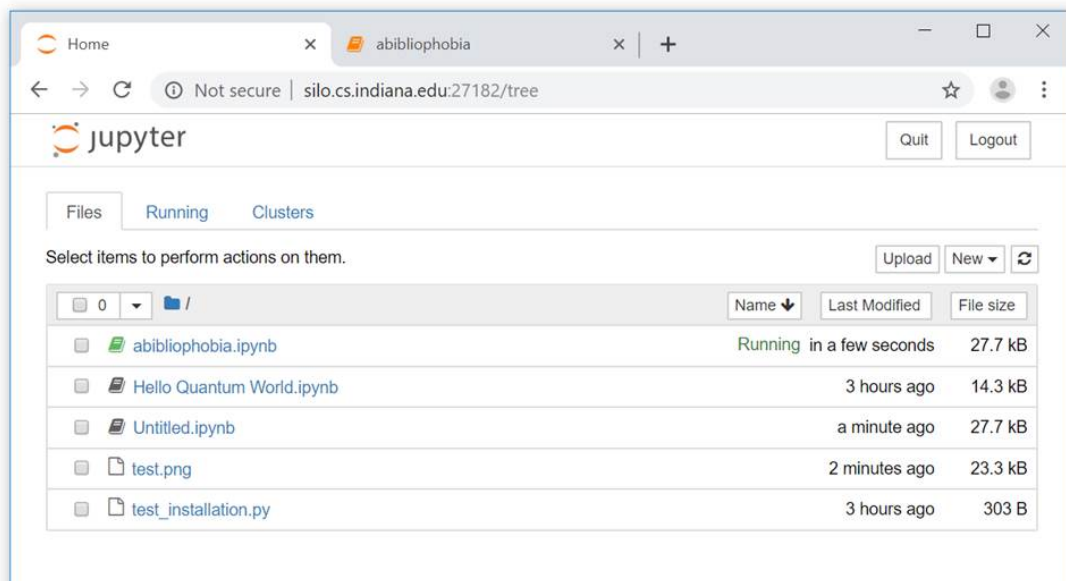


Figure 6: Verify that the file has been created.



```

In [3]: # This import registers the 3D projection, but is otherwise unused.
        from mpl_toolkits.mplot3d import Axes3D # noqa: F401 unused import

        import matplotlib.pyplot as plt
        from matplotlib import cm
        from matplotlib.ticker import LinearLocator, FormatStrFormatter
        import numpy as np

        fig = plt.figure()
        ax = fig.gca(projection='3d')

        # Make data.
        X = np.arange(-5, 5, 0.25)
        Y = np.arange(-5, 5, 0.25)
        X, Y = np.meshgrid(X, Y)
        R = np.sqrt(X**2 + Y**2)
        Z = np.sin(R)

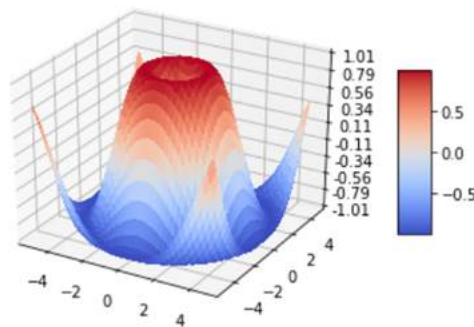
        # Plot the surface.
        surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                               linewidth=0, antialiased=False)

        # Customize the z axis.
        ax.set_zlim(-1.01, 1.01)
        ax.zaxis.set_major_locator(LinearLocator(10))
        ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

        # Add a color bar which maps values to colors.
        fig.colorbar(surf, shrink=0.5, aspect=5)

        plt.show()

```



```

In [ ]:

```

Figure 7: Try some other code (anything else you want) then save and exit.

Meanwhile on silo we have:

```
(MoranQC) -bash-4.2$ pwd
/u/dgerman
(MoranQC) -bash-4.2$ cd qc-6w2-2019/
(MoranQC) -bash-4.2$ cd Mastering-Quantum-Computing-with-IBM-QX/
(MoranQC) -bash-4.2$ cd Chapter01
(MoranQC) -bash-4.2$ jupyter notebook --port=27182 --ip=silo.cs.indiana.edu --no-browser
[I 21:35:14.024 NotebookApp] JupyterLab extension loaded from /l/python3.6.6/lib/python3.6/site[...]
[I 21:35:14.024 NotebookApp] JupyterLab application directory is /nfs/nfs5-insecure/home/insecu[...]
[I 21:35:14.026 NotebookApp] Serving notebooks from local directory: /nfs/nfs6/home/dgerman/qc-[...]
[I 21:35:14.026 NotebookApp] The Jupyter Notebook is running at:
[I 21:35:14.026 NotebookApp] http://silo.cs.indiana.edu:27182/?token=7fd4a3f77f58390bb82d2f66b2[...]
[I 21:35:14.026 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice[...])
[C 21:35:14.026 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://silo.cs.indiana.edu:27182/?token=7fd4a3f77f58390bb82d2f66b26a5b922233f43d7a2448e[...]
[I 21:39:51.304 NotebookApp] 302 GET /?token=7fd4a3f77f58390bb82d2f66b26a5b922233f43d7a2448ee ([...]
[I 21:43:47.147 NotebookApp] Creating new notebook in
[I 21:43:50.048 NotebookApp] Kernel started: 0e673cad-af2a-4c0e-a5bf-c561d6877ad0
[I 21:43:52.348 NotebookApp] Adapting to protocol v5.1 for kernel 0e673cad-af2a-4c0e-a5bf-c561d[...]
[I 21:45:50.109 NotebookApp] Saving file at /Untitled.ipynb
[I 21:49:50.234 NotebookApp] Saving file at /Untitled.ipynb
[W 21:51:11.491 NotebookApp] 404 GET /api/contents/abibliophobia.ipynb?type=notebook&content=0&[...]
[W 21:51:11.491 NotebookApp] No such file or directory: abibliophobia.ipynb
[W 21:51:11.491 NotebookApp] 404 GET /api/contents/abibliophobia.ipynb?type=notebook&content=0&[...]
[I 21:51:11.676 NotebookApp] Uploading file to /abibliophobia.ipynb
[I 21:53:58.283 NotebookApp] Saving file at /abibliophobia.ipynb
[I 21:55:01.361 NotebookApp] Saving file at /abibliophobia.ipynb
[I 21:55:14.142 NotebookApp] Starting buffering for 0e673cad-af2a-4c0e-a5bf-c561d6877ad0:3e382b[...]
^C[I 21:55:49.419 NotebookApp] interrupted
Serving notebooks from local directory: /nfs/nfs6/home/dgerman/qc-6w2-2019/Mastering-Quantum-Co[...]
1 active kernel
The Jupyter Notebook is running at:
http://silo.cs.indiana.edu:27182/?token=7fd4a3f77f58390bb82d2f66b26a5b922233f43d7a2448ee
Shutdown this notebook server (y/[n])? y
[C 21:55:51.947 NotebookApp] Shutdown confirmed
[I 21:55:51.947 NotebookApp] Shutting down 1 kernel
[I 21:55:52.248 NotebookApp] Kernel shutdown: 0e673cad-af2a-4c0e-a5bf-c561d6877ad0
(MoranQC) -bash-4.2$ ls
abibliophobia.ipynb Hello Quantum World.ipynb test_installation.py test.png Untitled.ipynb
(MoranQC) -bash-4.2$
```

Now you can stop the server, deactivate the virtual environment and log out.

## 2 Qubits, Gates, Circuits

Let's now<sup>7</sup> look at Chapter 02. We logged out, went did something else, now we log in again. What do we need to do? Four things, in order: (a) first

---

<sup>7</sup>There's a little more to Chapter 01, but we'll address that soon.

we need to load the right Python, (b) then we need to activate the virtual environment, (c) then we need to go into the folder for Chapter 02 and (d) finally, start the server:

```
-bash-4.2$ module load python/3
-bash-4.2$ source MoranQC/bin/activate
(MoranQC) -bash-4.2$ cd qc-6w2-2019/
(MoranQC) -bash-4.2$ cd Mastering-Quantum-Computing-with-IBM-QX/
(MoranQC) -bash-4.2$ cd Chapter02
(MoranQC) -bash-4.2$ ls
SimulatingQubits.ipynb
(MoranQC) -bash-4.2$ jupyter notebook --port=27182 --ip=silo.cs.indiana.edu --no-browser
[I 02:21:49.901 NotebookApp] JupyterLab extension loaded from /l/python3.6.6/lib/python3.6/site[...]
[I 02:21:49.901 NotebookApp] JupyterLab application directory is /nfs/nfs5-insecure/home/insecu[...]
[I 02:21:49.903 NotebookApp] Serving notebooks from local directory: /nfs/nfs6/home/dgerman/qc-[...]
[I 02:21:49.903 NotebookApp] The Jupyter Notebook is running at:
[I 02:21:49.903 NotebookApp] http://silo.cs.indiana.edu:27182/?token=644301f85586c1e353baf198c5[...]
[I 02:21:49.903 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice[...
[...
[I 02:21:58.699 NotebookApp] 302 GET /?token=644301f85586c1e353baf198c54f775048f32e8409de7406 [...]
[W 02:22:05.315 NotebookApp] Notebook SimulatingQubits.ipynb is not trusted
[I 02:22:06.041 NotebookApp] Kernel started: 7ab79f93-9410-44f8-a02d-4df47028bbe9
[I 02:22:07.007 NotebookApp] Adapting to protocol v5.1 for kernel 7ab79f93-9410-44f8-a02d-4df4[...]
[I 02:24:06.567 NotebookApp] Saving file at /SimulatingQubits.ipynb
[...]
```

So when we get online we find the code from Chapter 02 ready. Press Run to run each one of the cells in the notebook. We get something like this:

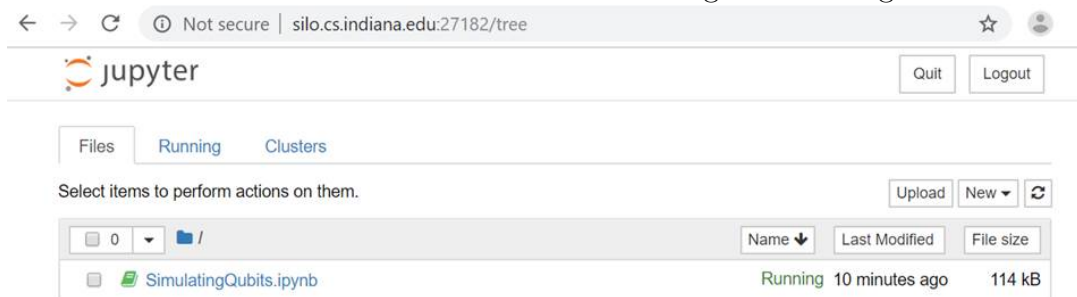



Figure 8: Start the notebook in Chapter 02 and that's what you see when you connect online. You are however stuck in that folder and can only access what's inside. Sub-folders are accessible (there aren't any here) but not the parent folder(s).

← → ↻ ⓘ Not secure | silo.cs.indiana.edu:27182/notebooks/SimulatingQubits.ipynb ☆

**Jupyter** SimulatingQubits (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

⏏ + ⌂ ↶ ↷ ⏏ Run ⏏ Code

---

In [1]: `%matplotlib inline  
import numpy as np`

### Simulating a Qubit

In [2]: `zero_qubit=np.matrix('1; 0')  
one_qubit=np.matrix('0; 1')`

In [3]: `def zero_to_one_qubit(percentage_zero,percentage_one):  
 if not percentage_zero+percentage_one==100 or percentage_zero<0 or percentage  
 return np.sqrt(percenta`

In [4]: `fifty_fifty_qubit=zero_to_one_qubit(50,50)  
ten_ninety_qubit=zero_to_one_qubit(10,90)  
print("fifty_fifty",fifty_fifty_qubit)  
print("ten_ninety",ten_ninety_qubit)`

```
fifty_fifty [[0.70710678]
             [0.70710678]]
ten_ninety  [[0.31622777]
             [0.9486833 ]]
```

### Three Different Representations of Qubits

In [5]: `def qubit(percentage_first,percentage_second,basis_first,basis_second):  
 if not percentage_first+percentage_second==100 or percentage_first<0 or percer  
 return sqrt(percenta`

In [6]: `plus_qubit=1/np.sqrt(2)*np.matrix('1; 1')  
minus_qubit=1/np.sqrt(2)*np.matrix('1; -1')`

In [7]: `clockwisearrow_qubit=1/np.sqrt(2)*np.matrix([[1],[np.complex(0,1)]])  
counterclockwisearrow_qubit=1/np.sqrt(2)*np.matrix([[1],[-np.complex(0,1)]])`

Figure 9: Using Python code to simulate various representations of qubits and their representations. This code is explained/developed in Chapter 02.

The image shows a Jupyter Notebook window titled "SimulatingQubits (autosaved)". The interface includes a top bar with the Jupyter logo, a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), and a status bar (Trusted, Python 3). Below the menu bar is a toolbar with icons for file operations and execution. The notebook content is divided into two cells. The first cell, labeled "In [8]:", contains a Python function definition for calculating Bloch sphere coordinates. The second cell, labeled "In [9]:", contains a series of print statements that call the function for various qubit states. The output of the second cell is displayed below the code, showing the coordinates for each state.

### The Bloch Sphere

```

In [8]: def get_bloch_coordinates(qubit):
        def get_x_bloch(qubit):
            qubit_x_basis=1./np.sqrt(2)*np.matrix('1 1; 1 -1')*qubit
            prob_zero_qubit=(qubit_x_basis.item(0)*qubit_x_basis.item(0).conjugate()).r
            prob_one_qubit=(qubit_x_basis.item(1)*qubit_x_basis.item(1).conjugate()).r
            return prob_zero_qubit-prob_one_qubit

        def get_y_bloch(qubit):
            qubit_y_basis=1./np.sqrt(2)*np.matrix('1 1; 1 -1')*np.matrix([[1,0],[0,-1]])
            prob_zero_qubit=(qubit_y_basis.item(0)*qubit_y_basis.item(0).conjugate()).r
            prob_one_qubit=(qubit_y_basis.item(1)*qubit_y_basis.item(1).conjugate()).r
            return prob_zero_qubit-prob_one_qubit

        def get_z_bloch(qubit):
            qubit_z_basis=qubit
            prob_zero_qubit=(qubit_z_basis.item(0)*qubit_z_basis.item(0).conjugate()).r
            prob_one_qubit=(qubit_z_basis.item(1)*qubit_z_basis.item(1).conjugate()).r
            return prob_zero_qubit-prob_one_qubit
        return (get_x_bloch(qubit),get_y_bloch(qubit),get_z_bloch(qubit))

In [9]: print('|"0"> coordinates are:',get_bloch_coordinates(zero_qubit))
        print('|"1"> coordinates are:',get_bloch_coordinates(one_qubit))
        print('|"+"> coordinates are:',get_bloch_coordinates(plus_qubit))
        print('|"-"> coordinates are:',get_bloch_coordinates(minus_qubit))
        print('|"⌚"> coordinates are:',get_bloch_coordinates(clockwisearrow_qubit))
        print('|"⌚"> coordinates are:',get_bloch_coordinates(counterclockwisearrow_qubit))

        |"0"> coordinates are: (0.0, 0.0, 1)
        |"1"> coordinates are: (0.0, 0.0, -1)
        |"+"> coordinates are: (0.9999999999999996, 0.0, 0.0)
        |"-"> coordinates are: (-0.9999999999999996, 0.0, 0.0)
        |"⌚"> coordinates are: (0.0, 0.9999999999999996, 0.0)
        |"⌚"> coordinates are: (0.0, -0.9999999999999996, 0.0)

```

Figure 10: Learning to visualize a single cubit on the Bloch sphere.

```

In [10]: def plot_bloch(qubit,color='b',ax=None):
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
if not ax:
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # draw sphere
    u, v = np.mgrid[0:2*np.pi:20j, 0:np.pi:10j]
    x = np.cos(u)*np.sin(v)
    y = np.sin(u)*np.sin(v)
    z = np.cos(v)
    ax.plot_wireframe(x, y, z, color="k",alpha=.1)
    ax.grid(False)

    coordinates=get_bloch_coordinates(qubit)
    ax.quiver([0],[0],[0],[coordinates[0]],[coordinates[1]],[coordinates[2]],length=1)
    ax.set_xlim([-1,1])
    ax.set_ylim([-1,1])
    ax.set_zlim([-1,1])
    ax.set_xlabel('x: |"-> to |"+>')
    ax.set_ylabel('y: |"0"> to |"1">')
    ax.set_zlabel('z: |"1"> to |"0">')
    ax.view_init(azim=20)
    return ax

# Plotting all of our basis qubits, colors and orientation match the textbook figure
ax=plot_bloch(zero_qubit,color='xkcd:red')
plot_bloch(one_qubit,color='xkcd:orange',ax=ax)
plot_bloch(plus_qubit,color='xkcd:yellow',ax=ax)
plot_bloch(minus_qubit,color='xkcd:green',ax=ax)
plot_bloch(clockwisearrow_qubit,color='xkcd:blue',ax=ax)
plot_bloch(counterclockwisearrow_qubit,color='xkcd:purple',ax=ax)

# Now plotting a qubit that is 10% |"0"> and 90% |"1"> in turquoise
plot_bloch(zero_to_one_qubit(10,90),color="xkcd:turquoise",ax=ax)

```

Out[10]: <matplotlib.axes.\_subplots.Axes3DSubplot at 0x7f3a2075db38>

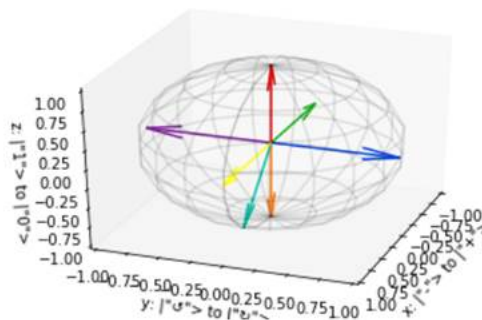


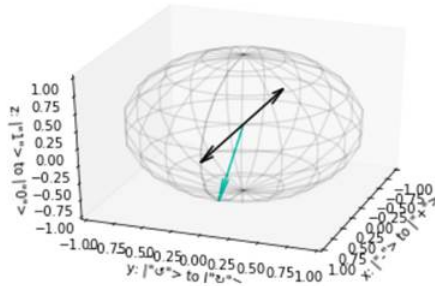
Figure 11: Visualizing qubits in 3D plots.



## Measuring a qubit on the Bloch Sphere

```
In [11]: ax=plot_bloch(plus_qubit,color='xkcd:black')
          plot_bloch(minus_qubit,color='xkcd:black',ax=ax)
          plot_bloch(ten_ninety_qubit,color="xkcd:turquoise",ax=ax)

Out[11]: <matplotlib.axes._subplots.Axes3DSubplot at 0x7f3a0f5ef4a8>
```



```
In [ ]:
```

Figure 12: We can visualize the measurement process on the Bloch sphere.

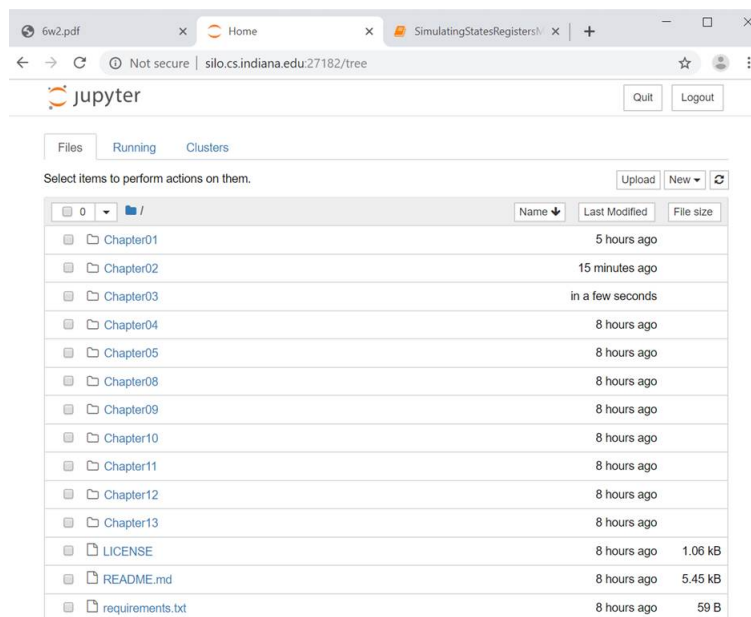


Figure 13: Stop server, start it in parent folder, now you can access all code.

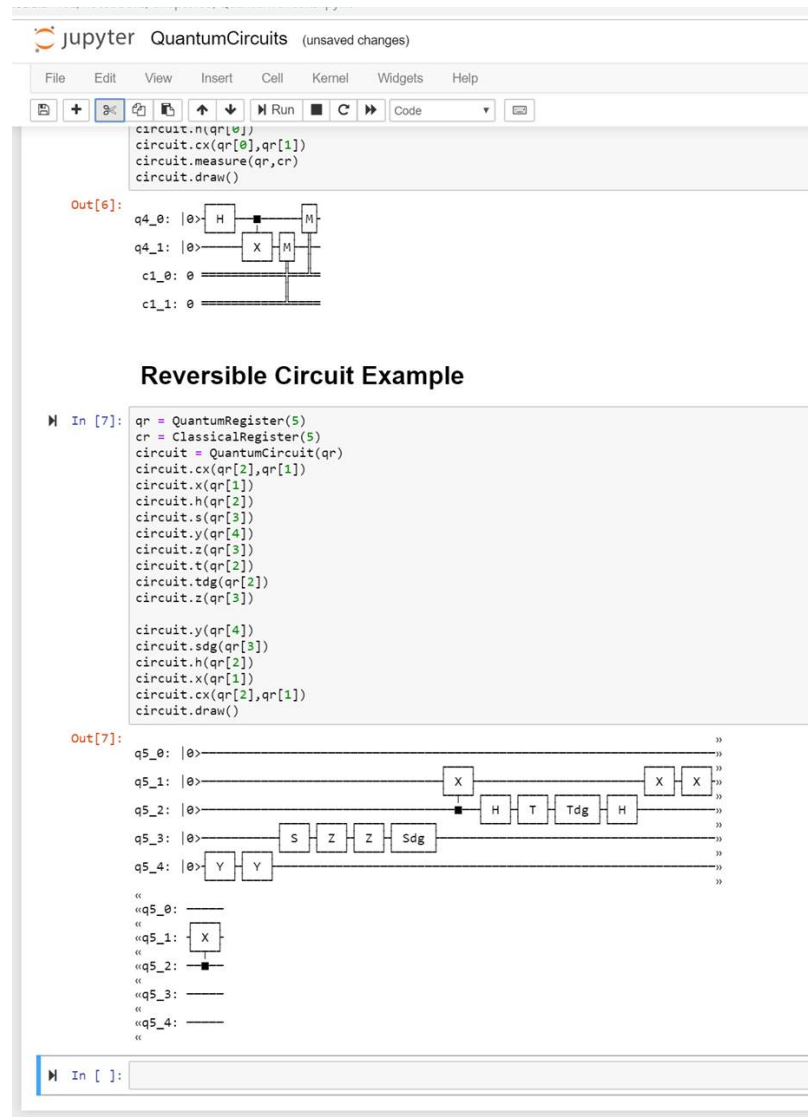


Figure 14: Quantum computing is reversible, which is remarkable because some of the logical operations we might perform in a classical computer can't be reversed. This example is not in the text per se, just in the notebook.

As it turns out all code in Chapters 02-05 works. There's no code in Chapters 06 and 07. Let's try to have the first 7 chapters under control so we can discuss them when we meet next time, most likely on Friday July 5 (tentatively, since we can also meet on Wednesday, July 3).



### 3 First Exam

Our first exam is next week, Friday July 5, 2019 in Luddy Hall (IF 2010). It will cover chapter 2-7 in Moran. We start Chapter 08 the week after.

```
In [11]: %matplotlib inline
# Importing standard Qiskit libraries and configuring account
from qiskit import QuantumCircuit, execute, Aer, IBMQ, ClassicalRegister, QuantumRegister
from qiskit.compiler import transpile, assemble
from qiskit.tools.jupyter import *
from qiskit.visualization import *
# Loading your IBM Q account(s)
IBMQ.load_accounts()
backend = IBMQ.get_backend('ibmq_qasm_simulator')

In [12]: import time
q = QuantumRegister(5)
c = ClassicalRegister(5)
qc = QuantumCircuit(q, c)
qc.measure(q, c)
job_exp = execute(qc, backend=backend)
```

Figure 15: Hello Quantum World! We need to clarify how one can connect remotely via token and/or URL. However, until then, this approach is also possible. But although this was working well at 2pm when Andrew came it wasn't working any more at 6:15pm when we discussed it with Lukas. So, this is (as of now not a big deal but over the next two days) something for me to work out and clarify.



(a) Andrew K Weissman



(b) Yifan Chen



(c) Lukas Cavar

Figure 16: I end this draft with the pics of the three students enrolled.

## 4 Loose Ends: Week 02

As mentioned earlier there was a problem with the user agreement: there was no way to agree to it. However I was logging in with my regular gmail account. Once I logged in with my Google @IU account I was presented with a user agreement that had a checkbox that I could check.

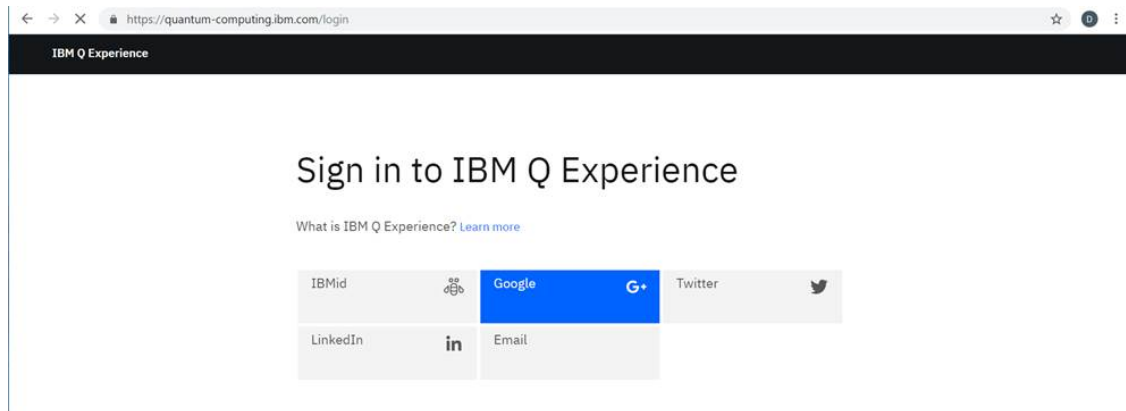


Figure 17: Log in with your Google @IU account for best results.

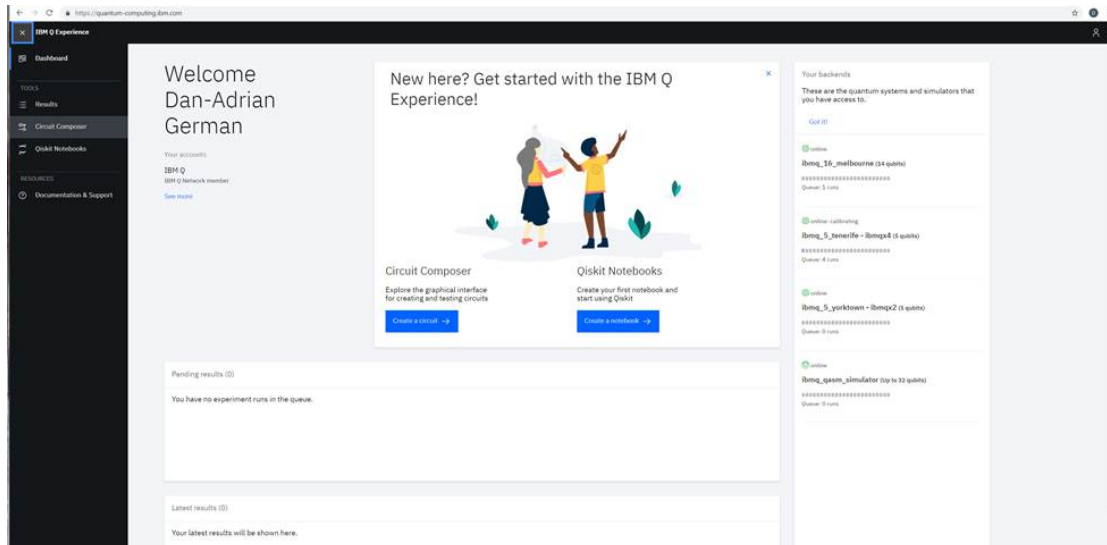


Figure 18: Since you're reading this online (it would be ridiculous to print it, right?!) please magnify to your heart's content. Here I choose, following the steps on page 15, the Composer tab.

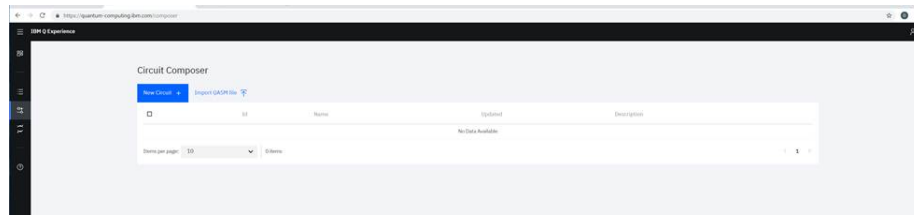


Figure 19: Now you can ask for a New Circuit (the blue button).

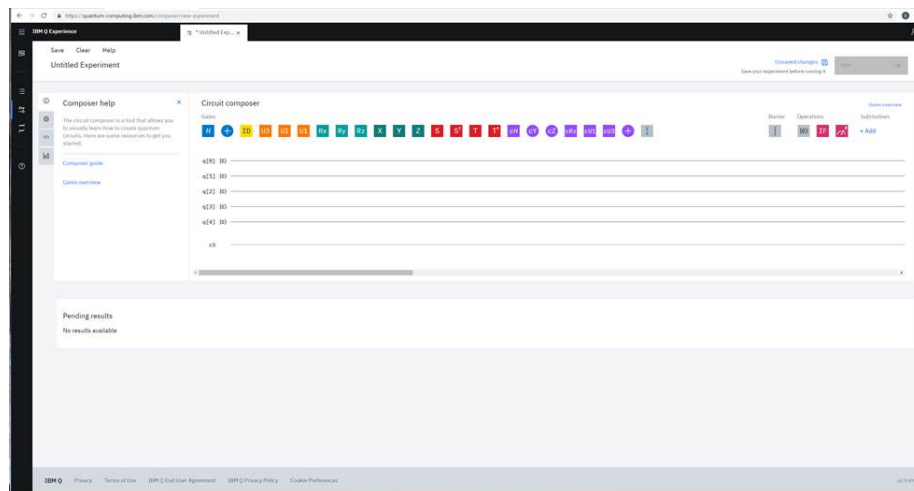


Figure 20: You are now ready to compose a new circuit.

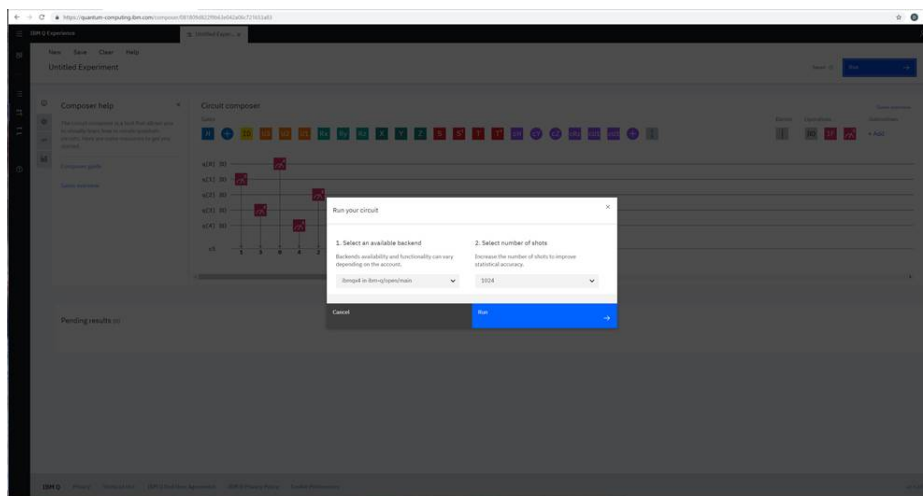


Figure 21: Create then run the new circuit as in the book.

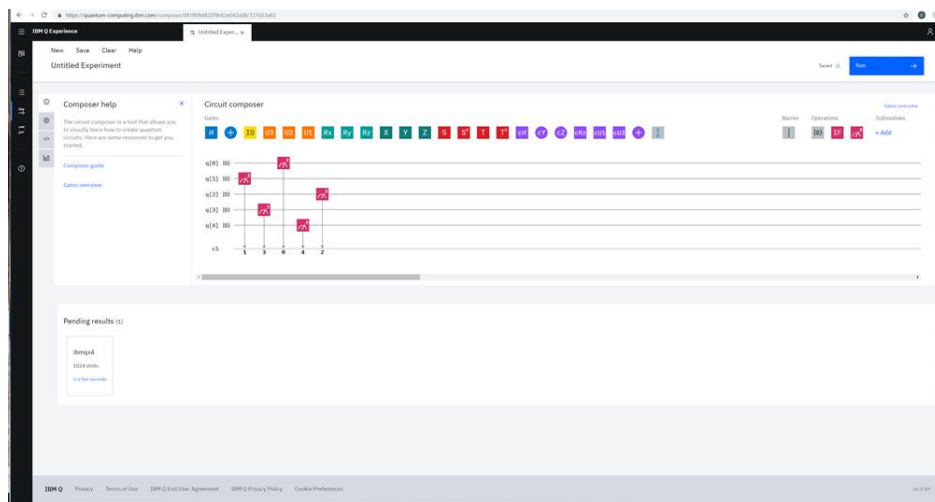


Figure 22: Check out the “in a few minutes” message posted in the left lower quadrant of the screen. Keep your eyes on it because it will change.

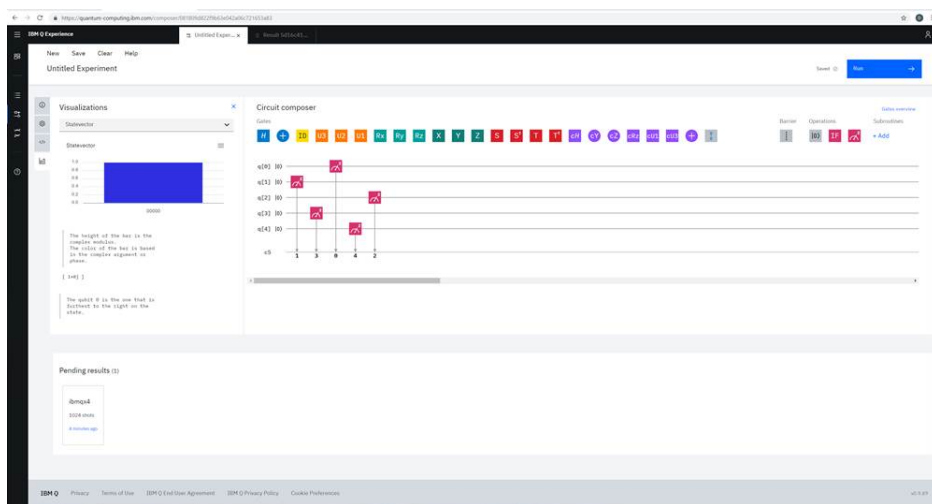


Figure 23: Look at the results as soon as they come back.

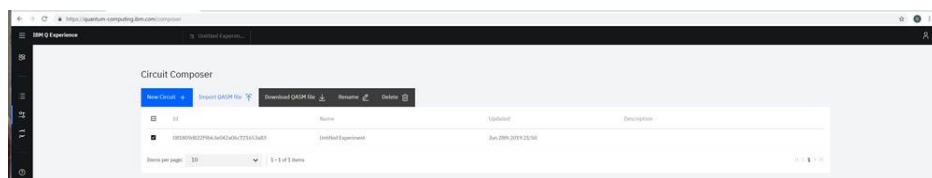


Figure 24: If you want to delete your experiment.

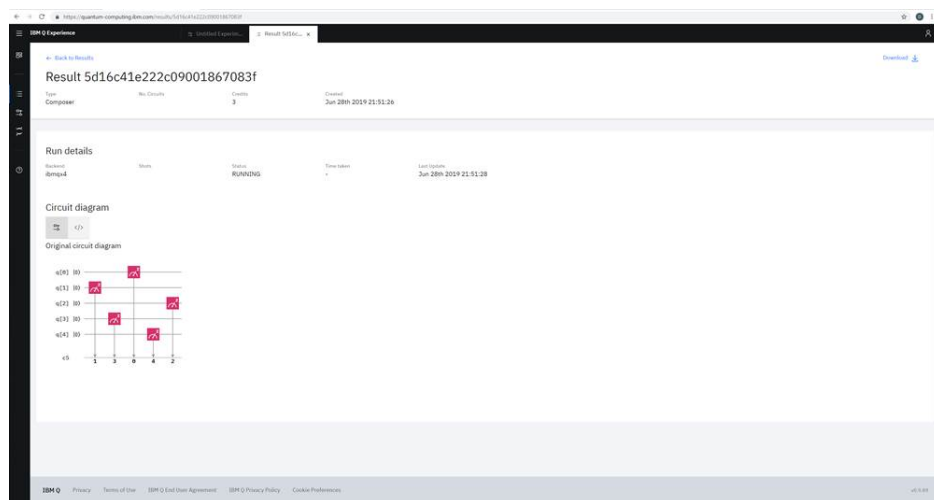


Figure 25: A little bit earlier, as the experiment was still running.

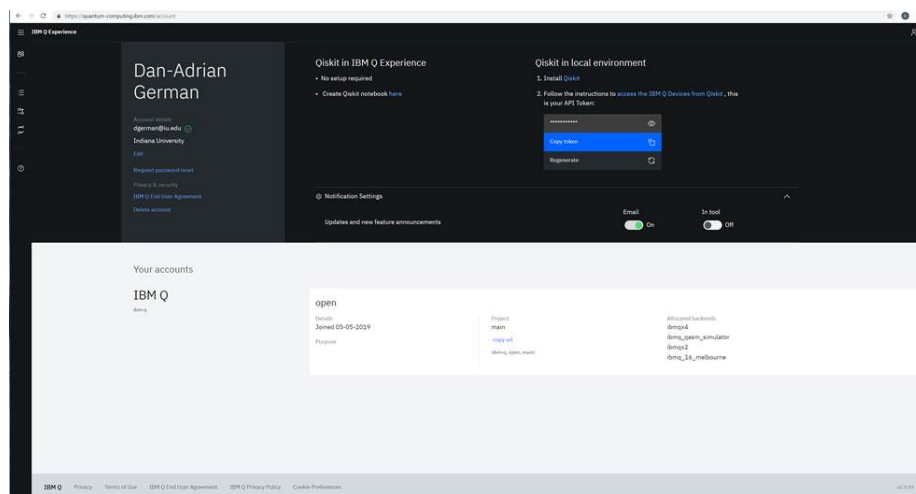



Figure 26: Finally make sure you confirm/validate your e-mail address.


So what does this mean? It means that now you have absolutely all the tools to get the first seven chapters done (and start the eighth). Over the next few days I will summarize that information here, for your review. As a matter of fact here's an update of something Andrew and I tried<sup>8</sup> several times (but could not get it to work) during our Thursday 6/27 appointments around 3pm. However as can be seen below<sup>9</sup> everything seems to be working now.

<sup>8</sup><https://github.com/Qiskit/qiskit-ibmq-provider>

<sup>9</sup><https://qiskit.org/documentation/faq.html>



Hello Quantum World
Last Checkpoint: Last Thursday at 2:57 PM (autosaved)


Logout

File Edit View Insert Cell Kernel Widgets Help
Not Trusted bookkernel

In [12]:
%matplotlib inline
import qiskit

### Put in your API Token and Register

In [13]:
from qiskit import IBMQ
# Authenticate an account and add for use during this session. Replace string argument with your private token.
IBMQ.save\_account('7b2971bba6c936e05eec09ffaf0dbfec9ba5cc4d845504adbe042aeaf221b143cb3275a43c2c90e69f45c9d04892495928065321e4605c3')
# To store your credentials locally you can run:
# IBMQ.save\_account("INSERT\_YOUR\_API\_TOKEN\_HERE")

### List the available backends and pick one

In [14]:
print(IBMQ.backends()) # remote IBM backends
from qiskit import Aer
print(Aer.backends()) # local backends

In [18]:
# Pick an available backend
backend = Aer.get\_backend('qasm\_simulator') # if this isn't available pick a backend whose name contains '\_qasm\_simulator' from t

### Run your first quantum program

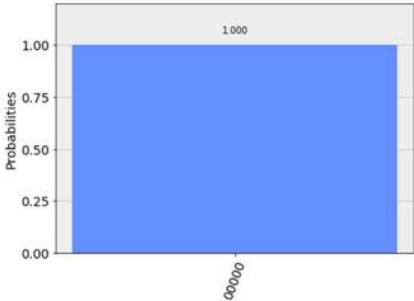
This program starts with an input of the quantum equivalent of five "0" bits, five quantum "0" bits. It then performs no action before it outputs the classical equivalent of the quantum bits. Since they haven't changed, the output should be 00000.

In [19]:
import time
q = qiskit.QuantumRegister(5)
c = qiskit.ClassicalRegister(5)
qc = qiskit.QuantumCircuit(q, c)
qc.measure(q, c)
job\_exp = qiskit.execute(qc, backend=backend)

### Seeing the results

Here we will get output 00000 with 100% probability as our output

In [20]:
from qiskit.tools.visualization import plot\_histogram
plot\_histogram(job\_exp.result().get\_counts(qc))

Out[20]:


In [ ]:

Figure 27: Last test from Chapter 01 we could not run last week (runs now).