Center for
Quantum Networks
NSF Engineering Research Center

Quantum Network Simulation Software

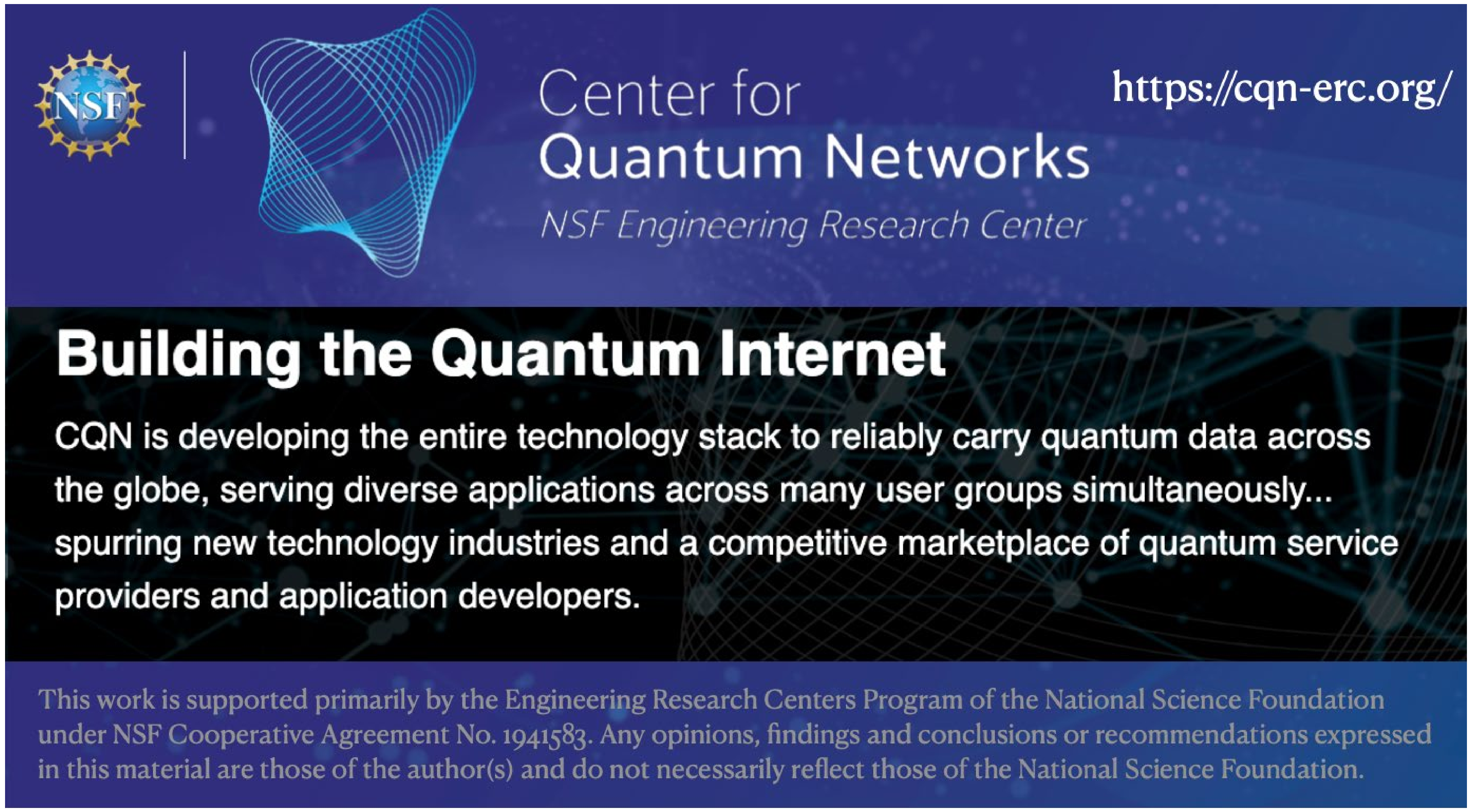
Instructor: Inès Montaño
Northern Arizona University

Co-Instructor: Jaime Diaz
Northern Arizona University

This work is supported primarily by the Engineering Research Centers Program of the National Science Foundation. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

CQN Winter School on Quantum Networks

Funded by National Science Foundation Grant #1941583



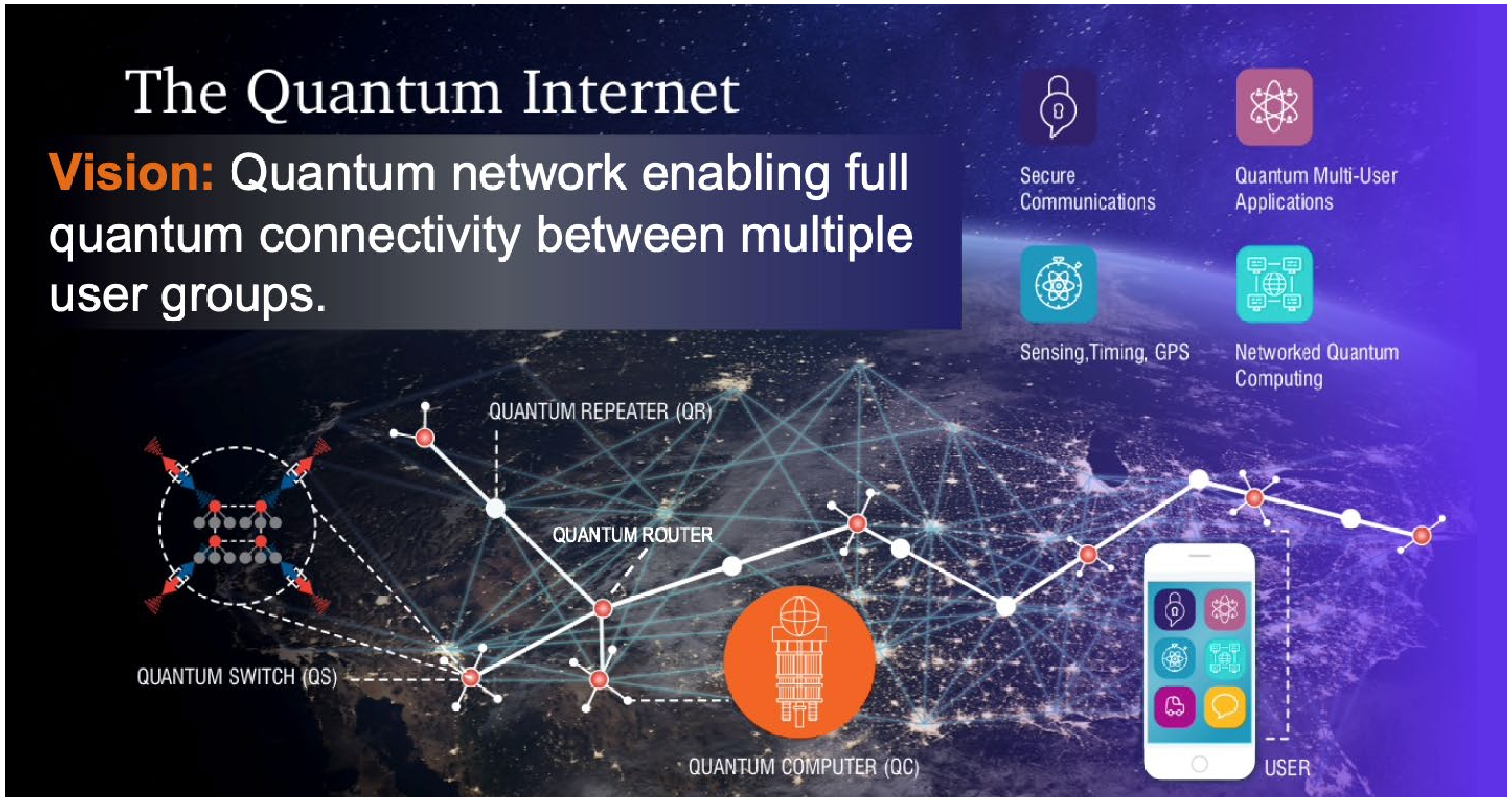
Center for
Quantum Networks
NSF Engineering Research Center

https://cqn-erc.org/

Building the Quantum Internet

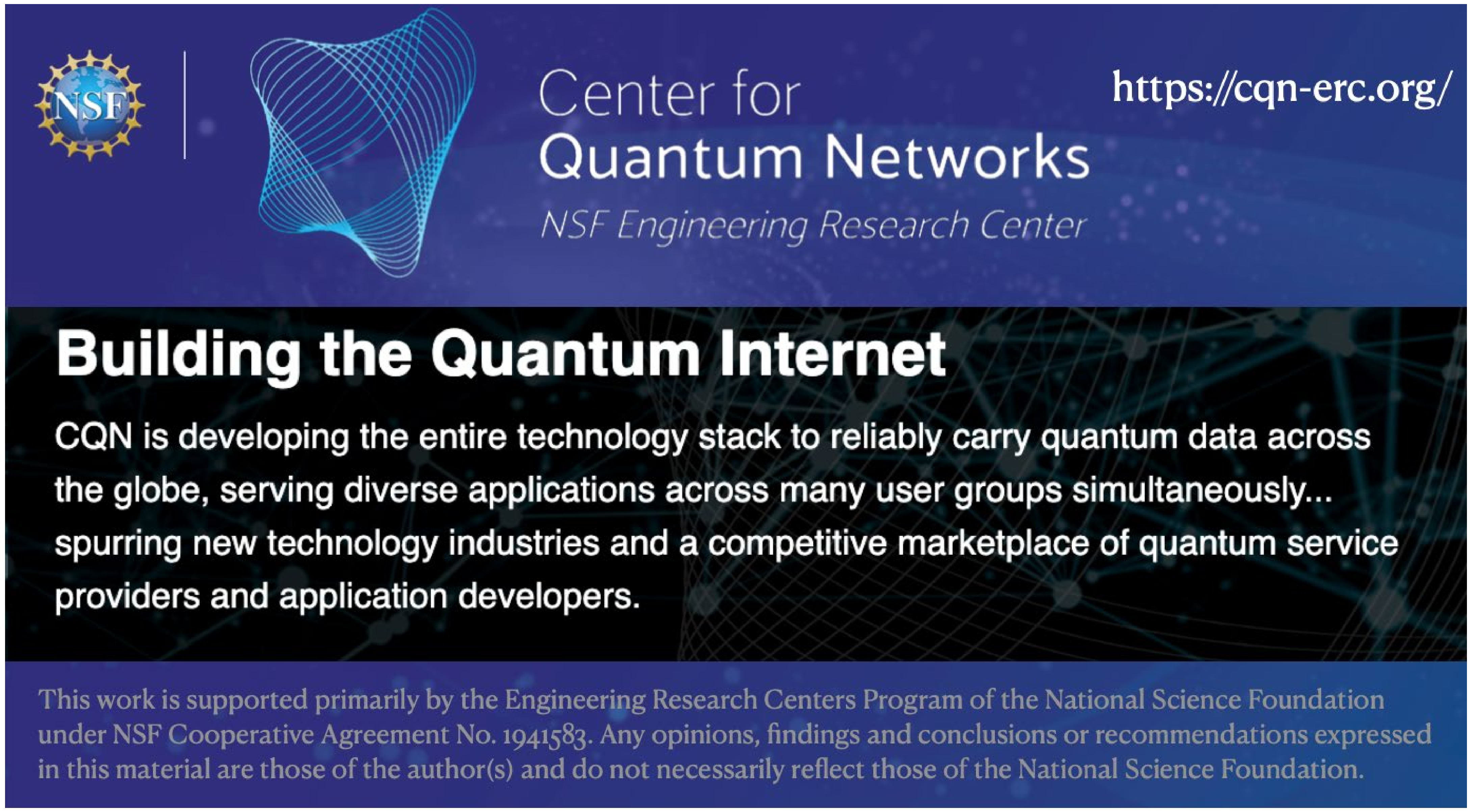
CQN is developing the entire technology stack to reliably carry quantum data across the globe, serving diverse applications across many user groups simultaneously... spurring new technology industries and a competitive marketplace of quantum service providers and application developers.

This work is supported primarily by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. 1941583. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.



The Quantum Internet

Vision: Quantum network enabling full quantum connectivity between multiple user groups.



Center for Quantum Networks

NSF Engineering Research Center

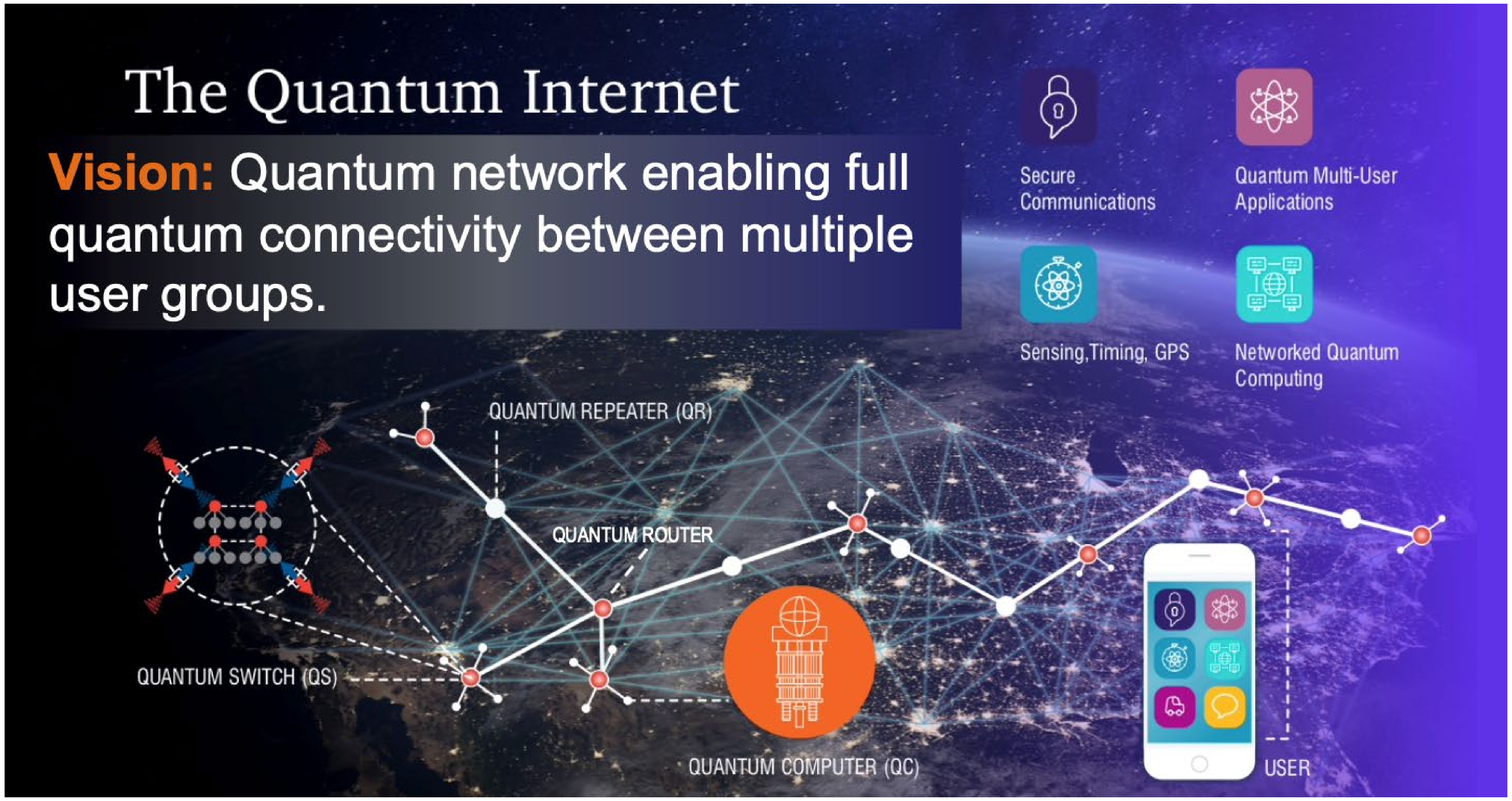
https://cqn-erc.org/

Building the Quantum Internet

CQN is developing the entire technology stack to reliably carry quantum data across the globe, serving diverse applications across many user groups simultaneously... spurring new technology industries and a competitive marketplace of quantum service providers and application developers.

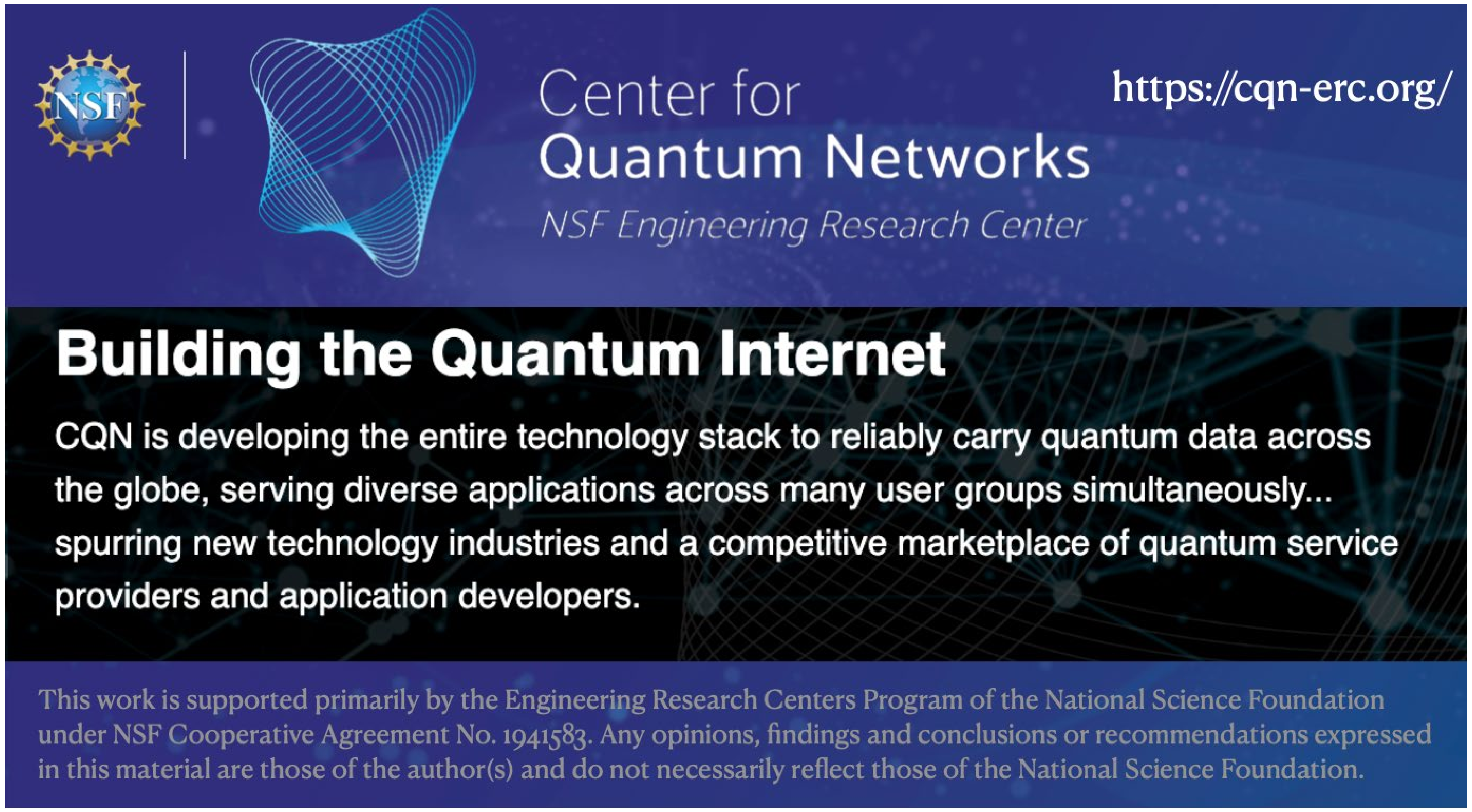
This work is supported primarily by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. 1941583. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

- Why
- What
- How



The Quantum Internet

Vision: Quantum network enabling full quantum connectivity between multiple user groups.



Center for Quantum Networks

NSF Engineering Research Center

https://cqn-erc.org/

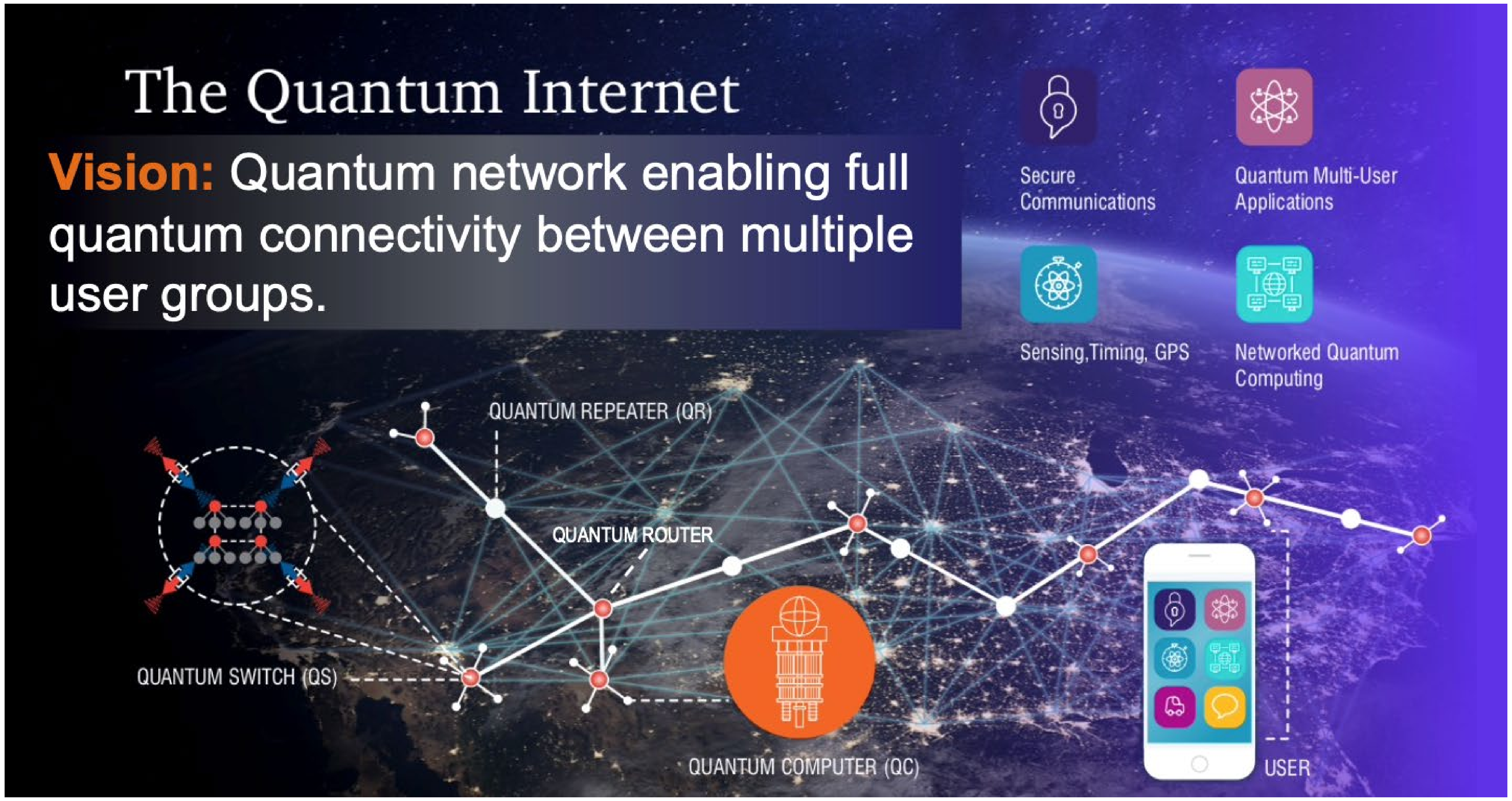
Building the Quantum Internet

CQN is developing the entire technology stack to reliably carry quantum data across the globe, serving diverse applications across many user groups simultaneously... spurring new technology industries and a competitive marketplace of quantum service providers and application developers.

This work is supported primarily by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. 1941583. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

- Why
- What
- How

- Applications
- Key components
- Challenges



The Quantum Internet

Vision: Quantum network enabling full quantum connectivity between multiple user groups.

Quantum Network Simulation Software

Quantum Network Simulators

(prob already not a complete list anymore...)

Quantum Network Simulators

(prob already not a complete list anymore...)

- Quantum-Network Explorer: https://www.quantum-network.com/

- QuISP: https://aqua.sfc.wide.ad.jp/quisp_website/
- SeQUeNce: https://github.com/sequence-toolbox/SeQUeNCe

- QuNetSim: https://github.com/tqsd/QuNetSim

- Squanch: https://github.com/att-innovate/squanch

- SimulaQron: https://github.com/SoftwareQuTech/SimulaQron

- NetSquid: https://netsquid.org/

Quantum Network Simulators

(prob already not a complete list anymore...)

- Quantum-Network Explorer: https://www.quantum-network.com/
- QuISP: https://aqua.sfc.wide.ad.jp/quisp_website/
- SeQUeNce: https://github.com/sequence-toolbox/SeQUeNCe
- QuNetSim: https://github.com/tqsd/QuNetSim
- Squanch: https://github.com/att-innovate/squanch
- SimulaQron: https://github.com/SoftwareQuTech/SimulaQron
- NetSquid: https://netsquid.org/

Aim to offer a way to test out 'things related to a quantum network'
- without having access to a physical quantum network !

(physical layer, protocols, control software, applications, performance prediction)

Quantum Network Simulators

- What one can do with ‘a’ quantum network simulator

Quantum Network Simulators

- What one can do with ‘a’ quantum network simulator
- **How to do SOMETHING with a quantum network simulator**

Quantum Network Simulators

- What one can do with ‘a’ quantum network simulator
- **How to do SOMETHING USEFUL with a quantum network simulator**

Quantum Network Simulators

- What one can do with ‘a’ quantum network simulator

- **How to do SOMETHING USEFUL with a quantum network simulator**

Who this short course is for…

Quantum Network Simulators

- What one can do with ‘a’ quantum network simulator

- **How to do SOMETHING USEFUL with a quantum network simulator**

Who this short course is for…

This short course is for anyone
interested in learning how to use a quantum network simulator.

Quantum Network Simulators

- What one can do with ‘a’ quantum network simulator
- **How to do SOMETHING USEFUL with a quantum network simulator**

Who this short course is for…

This short course is for anyone
interested in learning how to use a quantum network simulator.

Interested in using a quantum network simulator to:

- Explore concepts?
- Integrate it in your research?
- Or just have fun with it?

Quantum Network Simulators

- What one can do with ‘a’ quantum network simulator
- **How to do SOMETHING USEFUL with a quantum network simulator**

Who this short course is for…

This short course is for anyone
interested in learning how to use a quantum network simulator.

Interested in using a quantum network simulator to:

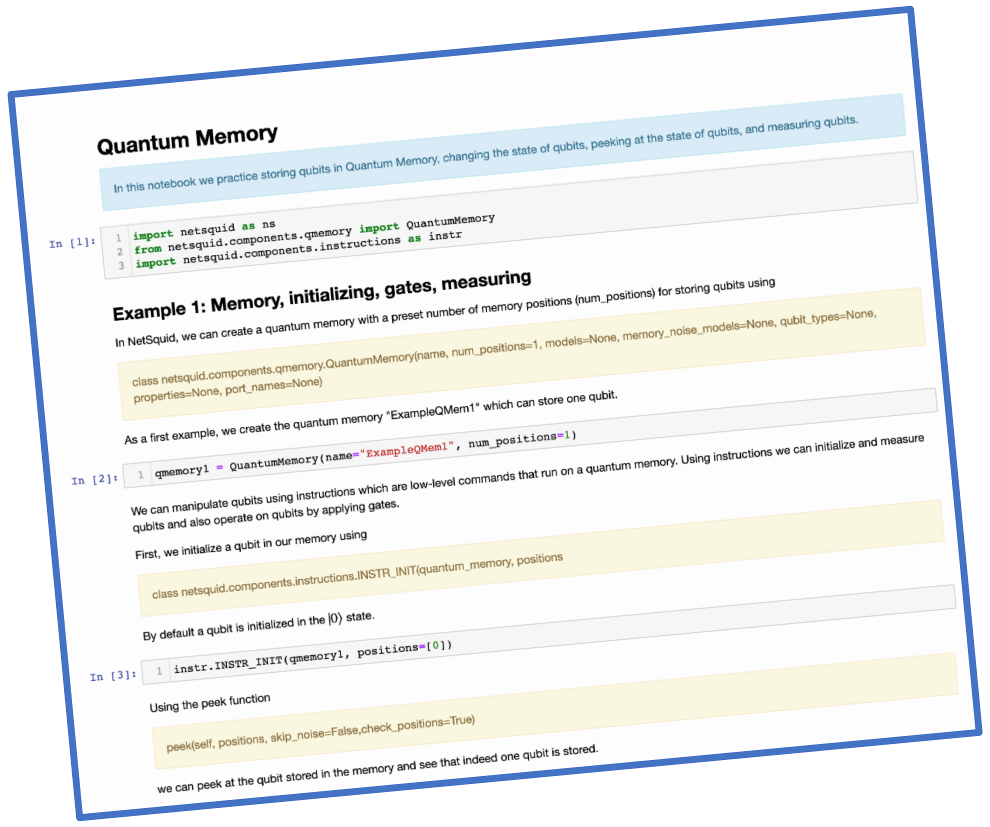
- Explore concepts?
- Integrate it in your research?
- Or just have fun with it?

Hopefully this short course can **help to get you started**
in doing **SOMETHING THAT YOU CONSIDER USEFUL** with a quantum network simulator

Simulation Deep Dive: Learning by Doing

Hands-on Explorations

Simulation Deep Dive: Learning by Doing



Quantum Memory

In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.

```
In [1]:  import netsquid as ns
         from netsquid.components.qmemory import QuantumMemory
         import netsquid.components.instructions as instr
```

Example 1: Memory, initializing, gates, measuring

In NetSquid, we can create a quantum memory with a preset number of memory positions (num_positions) for storing qubits using

class netsquid.components.qmemory.QuantumMemory(name, num_positions=1, models=None, memory_noise_models=None, qubit_types=None, properties=None, port_names=None)

As a first example, we create the quantum memory "ExampleQMem1" which can store one qubit.

```
In [2]:  qmemory1 = QuantumMemory(name="ExampleQMem1", num_positions=1)
```

We can manipulate qubits using instructions which are low-level commands that run on a quantum memory. Using instructions we can initialize and measure qubits and also operate on qubits by applying gates.

First, we initialize a qubit in our memory using

class netsquid.components.instructions.INSTR_INIT(quantum_memory, positions

By default a qubit is initialized in the $|0\rangle$ state.

```
In [3]:  instr.INSTR_INIT(qmemory1, positions=[0])
```

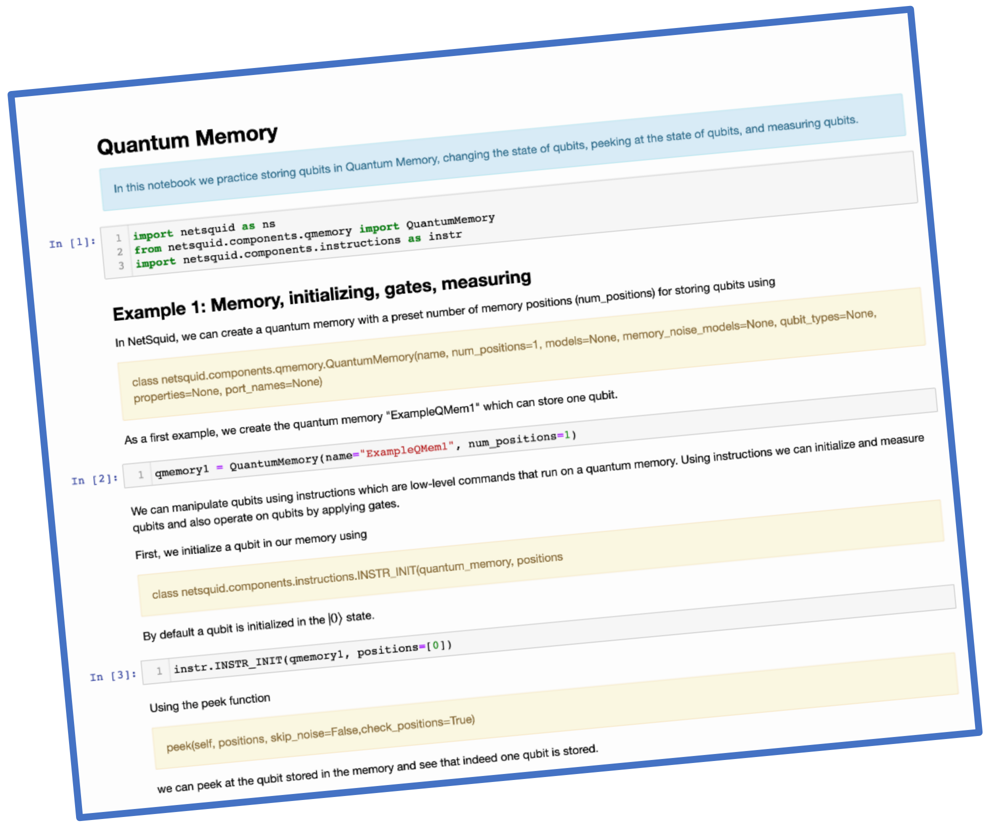
Using the peek function

peek(self, positions, skip_noise=False,check_positions=True)

we can peek at the qubit stored in the memory and see that indeed one qubit is stored.

Hands-on Explorations

Simulation Deep Dive: Learning by Doing



Quantum Memory

In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.

```
In [1]:
import netsquid as ns
from netsquid.components.qmemory import QuantumMemory
import netsquid.components.instructions as instr
```

Example 1: Memory, initializing, gates, measuring

In NetSquid, we can create a quantum memory with a preset number of memory positions (num_positions) for storing qubits using

class netsquid.components.qmemory.QuantumMemory(name, num_positions=1, models=None, memory_noise_models=None, qubit_types=None, properties=None, port_names=None)

As a first example, we create the quantum memory "ExampleQMem1" which can store one qubit.

```
In [2]:
qmemory1 = QuantumMemory(name="ExampleQMem1", num_positions=1)
```

We can manipulate qubits using instructions which are low-level commands that run on a quantum memory. Using instructions we can initialize and measure qubits and also operate on qubits by applying gates.

First, we initialize a qubit in our memory using

class netsquid.components.instructions.INSTR_INIT(quantum_memory, positions

By default a qubit is initialized in the $|0\rangle$ state.

```
In [3]:
instr.INSTR_INIT(qmemory1, positions=[0])
```

Using the peek function

peek(self, positions, skip_noise=False,check_positions=True)

we can peek at the qubit stored in the memory and see that indeed one qubit is stored.

Hands-on Explorations

- Introduce you to key building blocks of quantum networks/ quantum network simulators
- Provide you with opportunity to explore and try-out material
- Introduce you to material of increasing complexity
- Let you simulate a teleportation network protocol in a Quantum Network

How to get the most out of this course…

How to get the most out of this course…

- Be engaged!
- Work with the provided notebooks
- Try to apply the material
- **Ideally: work with others, discuss your questions etc.**

How to get the most out of this course…

- Be engaged!
- Work with the provided notebooks
- Try to apply the material
- **Ideally: work with others, discuss your questions etc.**

Poll:

Do you prefer to work alone on your own?
or
Are you interested in working in a breakout room so you can discuss with others?

Simulation Deep Dive: Learning by Doing

Hands-on Explorations

Hopefully this short course can **help to get you started**
in doing **SOMETHING THAT YOU CONSIDER USEFUL** with a quantum network simulator

Quantum Network Simulators

(prob already not a complete list anymore...)

- Quantum-Network Explorer: https://www.quantum-network.com/
- QuISP: https://aqua.sfc.wide.ad.jp/quisp_website/
- SeQUeNce: https://github.com/sequence-toolbox/SeQUeNCe
- QuNetSim: https://github.com/tqsd/QuNetSim
- Squanch: https://github.com/att-innovate/squanch
- SimulaQron: https://github.com/SoftwareQuTech/SimulaQron
- NetSquid: https://netsquid.org/

Aim to offer a way to test out ‘things related to a quantum network’
- without having access to a physical quantum network !

(physical layer, protocols, control software, applications, performance prediction)

Quantum Network Simulators

(prob already not a complete list anymore...)

- Quantum-Network Explorer: https://www.quantum-network.com/
- QuISP: https://aqua.sfc.wide.ad.jp/quisp_website/
- SeQUeNce: https://github.com/sequence-toolbox/SeQUeNCe
- QuNetSim: https://github.com/tqsd/QuNetSim
- Squanch: https://github.com/att-innovate/squanch
- SimulaQron: https://github.com/SoftwareQuTech/SimulaQron
- NetSquid: https://netsquid.org/

Aim to offer a way to test out 'things related to a quantum network'
- without having access to a physical quantum network !

(physical layer, protocols, control software, applications, performance prediction)

Quantum Network Simulators

(prob already not a complete list anymore...)

- Quantum-Network Explorer: https://www.quantum-network.com/
- QuISP: https://aqua.sfc.wide.ad.jp/quisp_website/
- SeQUeNce: https://github.com/sequence-toolbox/SeQUeNCe
- QuNetSim: https://github.com/tqsd/QuNetSim
- Squanch: https://github.com/att-innovate/squanch
- SimulaQron: https://github.com/SoftwareQuTech/SimulaQron
- NetSquid: https://netsquid.org/

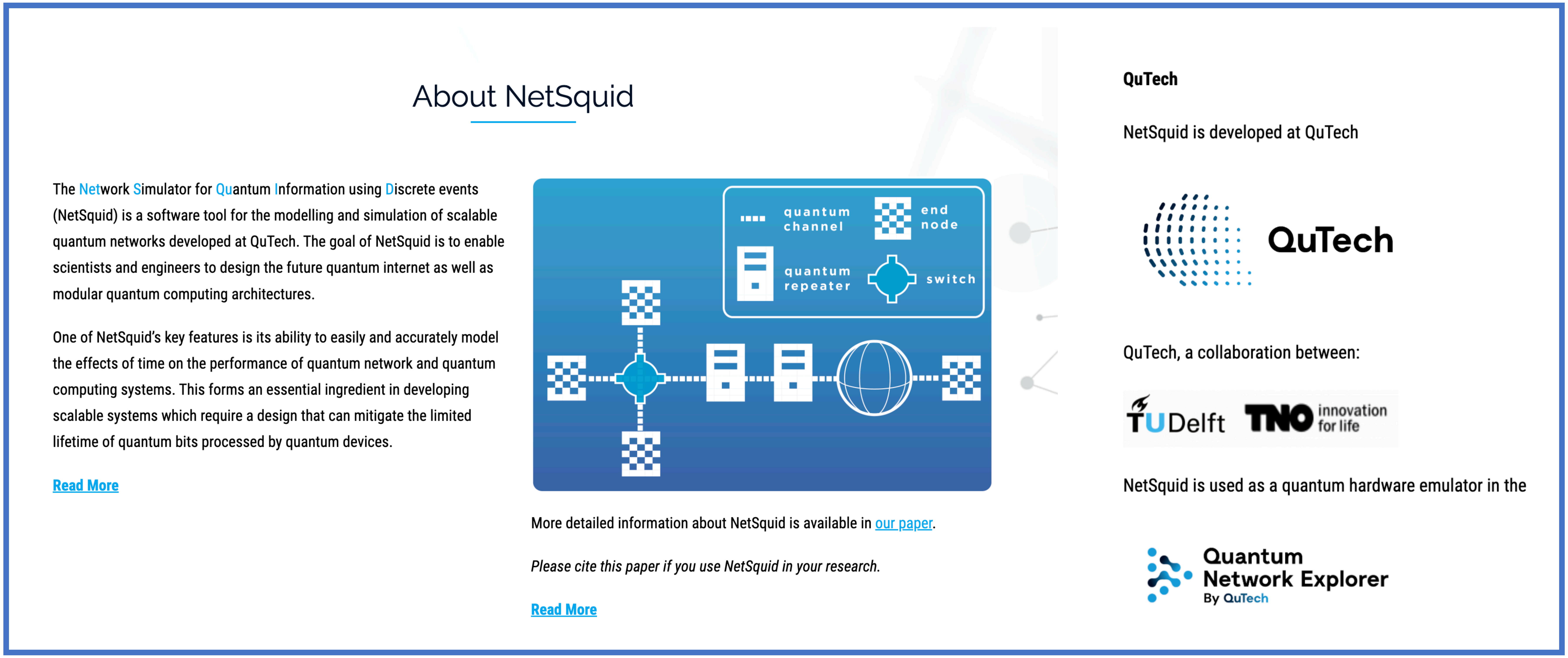
Official Disclaimer:
(this does NOT mean the others aren’t great!)

Aim to offer a way to test out ‘things related to a quantum network’
- without having access to a physical quantum network !

(physical layer, protocols, control software, applications, performance prediction)

For this short course we will use NetSquid.

https://netsquid.org/



About NetSquid

The Network Simulator for Quantum Information using Discrete events (NetSquid) is a software tool for the modelling and simulation of scalable quantum networks developed at QuTech. The goal of NetSquid is to enable scientists and engineers to design the future quantum internet as well as modular quantum computing architectures.

One of NetSquid's key features is its ability to easily and accurately model the effects of time on the performance of quantum network and quantum computing systems. This forms an essential ingredient in developing scalable systems which require a design that can mitigate the limited lifetime of quantum bits processed by quantum devices.

Read More

More detailed information about NetSquid is available in our paper.

Please cite this paper if you use NetSquid in your research.

Read More

QuTech

NetSquid is developed at QuTech

QuTech

QuTech, a collaboration between:

TU Delft TNO innovation for life

NetSquid is used as a quantum hardware emulator in the

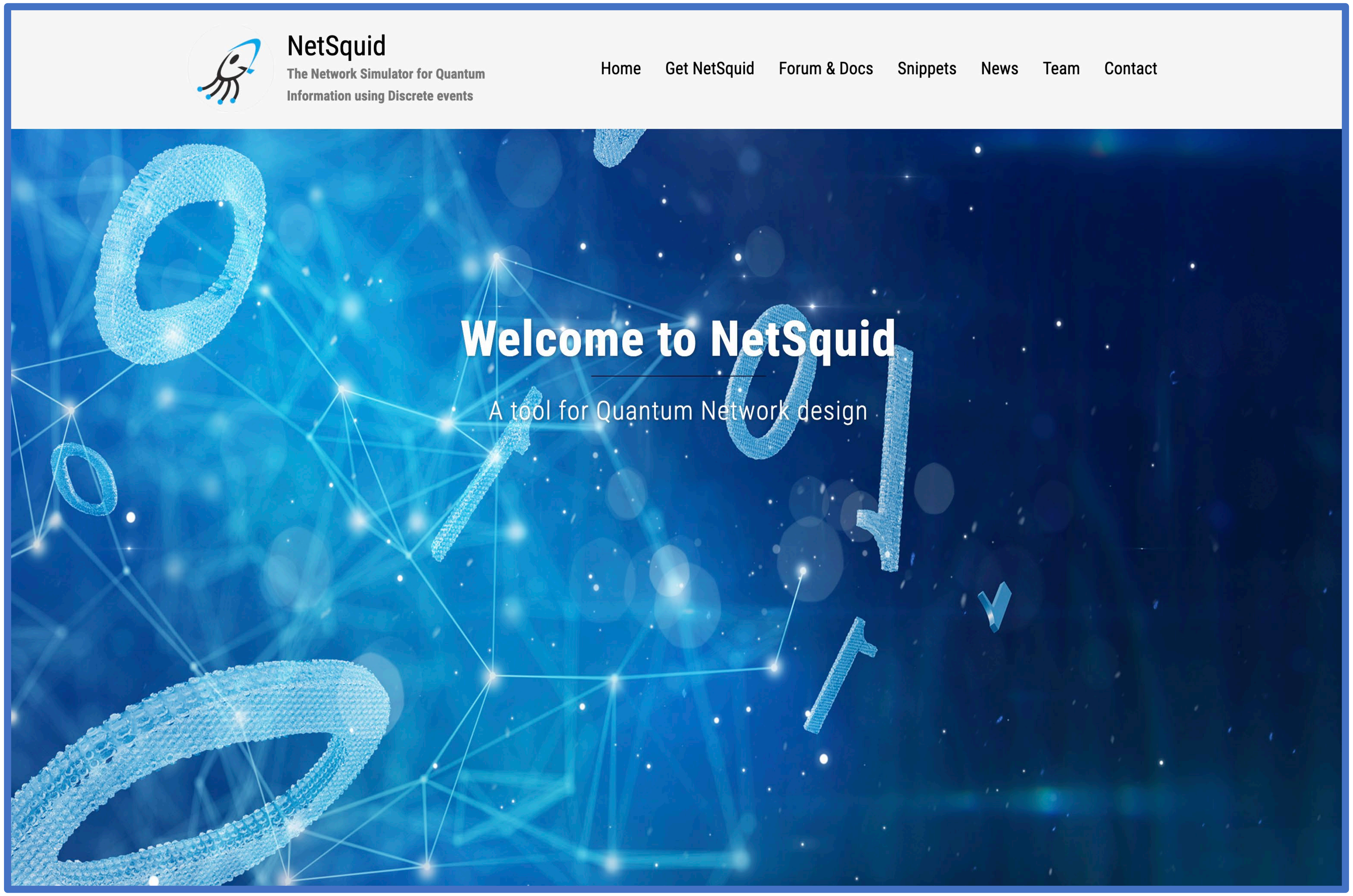
Quantum Network Explorer By QuTech

* Screenshots from https://netsquid.org

Official Disclaimer:
(this does NOT mean the others aren't great!)

For this short course we will use NetSquid.

https://netsquid.org/



NetSquid

The Network Simulator for Quantum Information using Discrete events

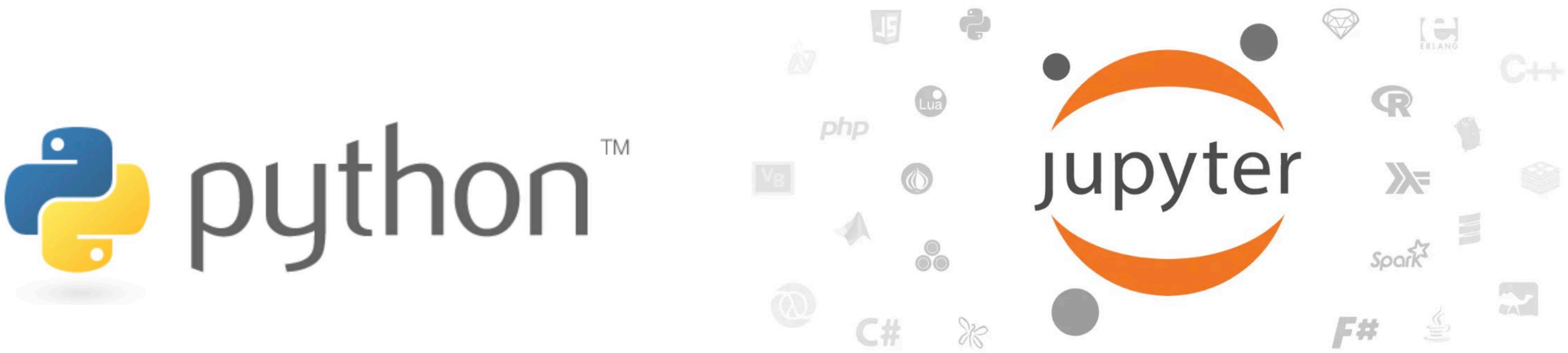
Welcome to NetSquid

A tool for Quantum Network design

* Screenshots from https://netsquid.org

Official Disclaimer:
(this does NOT mean the others aren't great!)

Simulations are in Python and will be run on a virtual machine using Jupyter Notebooks.



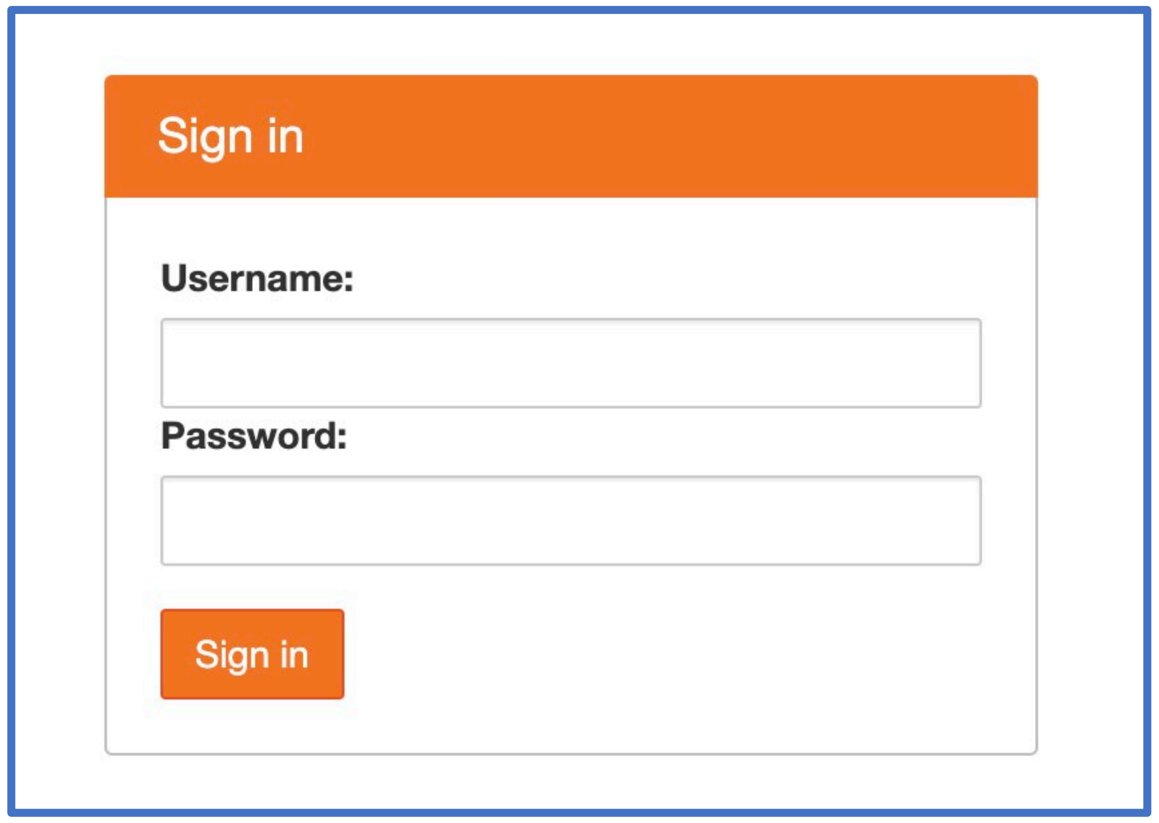
Jupyter Notebook

- Powerful tool to integrate code and output in single document
- Allows to combine code, output, text, equations, images

You will be working with notebooks online – in your browser.

To access the notebooks:

- Please go to https://miracqn.stonedwarf5.net/



Sign in

Username:

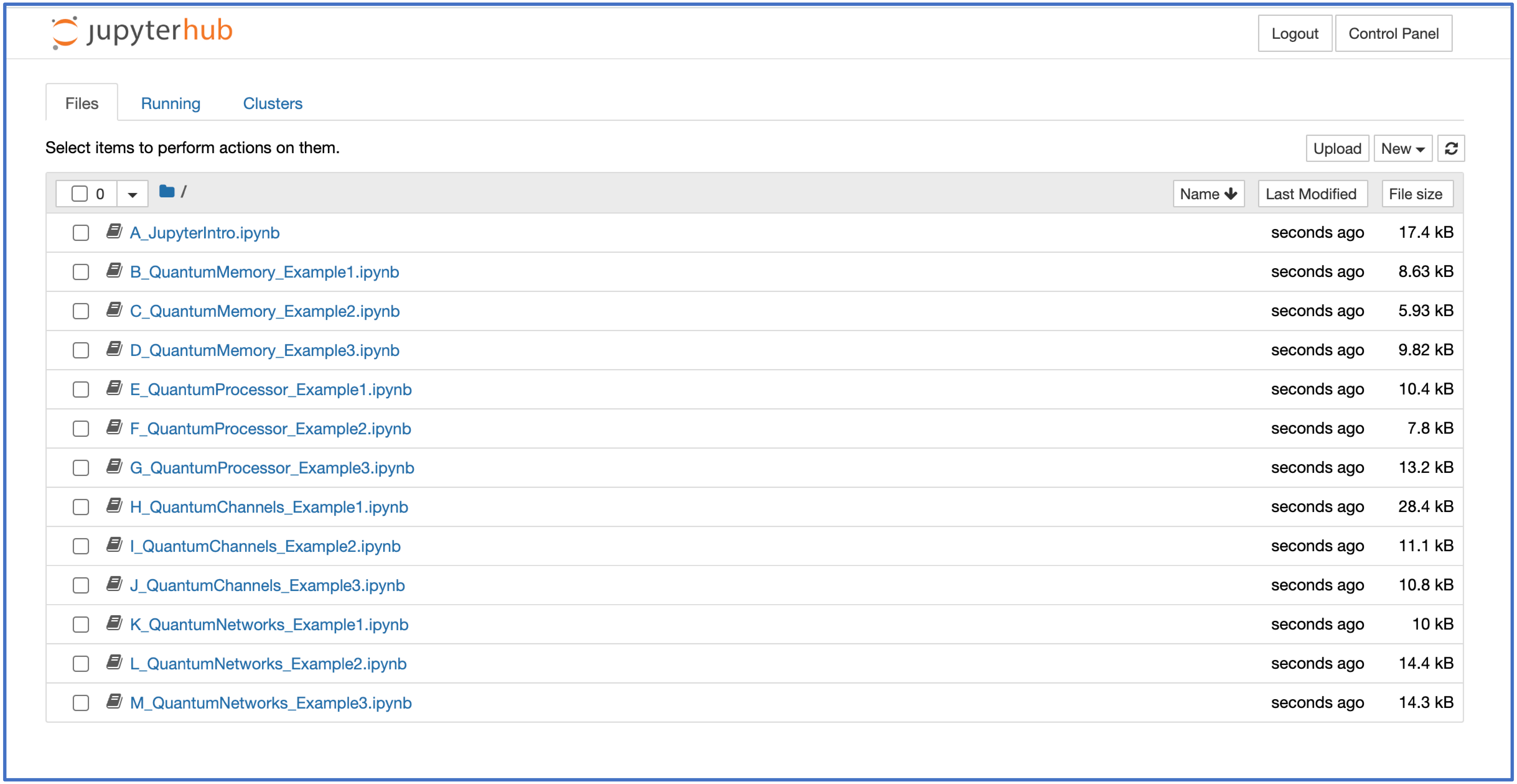
Password:

- Accounts are setup and ready to go (user1, user2, ..)
- Password: shortcourse8

Listen for your account info, then log onto the server, please.

Please write down your account!

What you should see

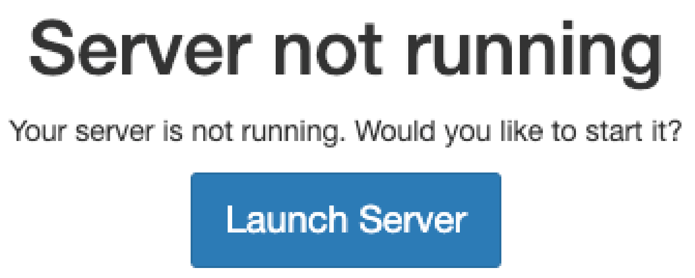


Select items to perform actions on them.

□ 0 /

	Name	Last Modified	File size
□	A_JupyterIntro.ipynb	seconds ago	17.4 kB
□	B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□	C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□	D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□	E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□	F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
□	G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
□	H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□	I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□	J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□	K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□	L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□	M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

If you ever see this: →
Just click launch server.

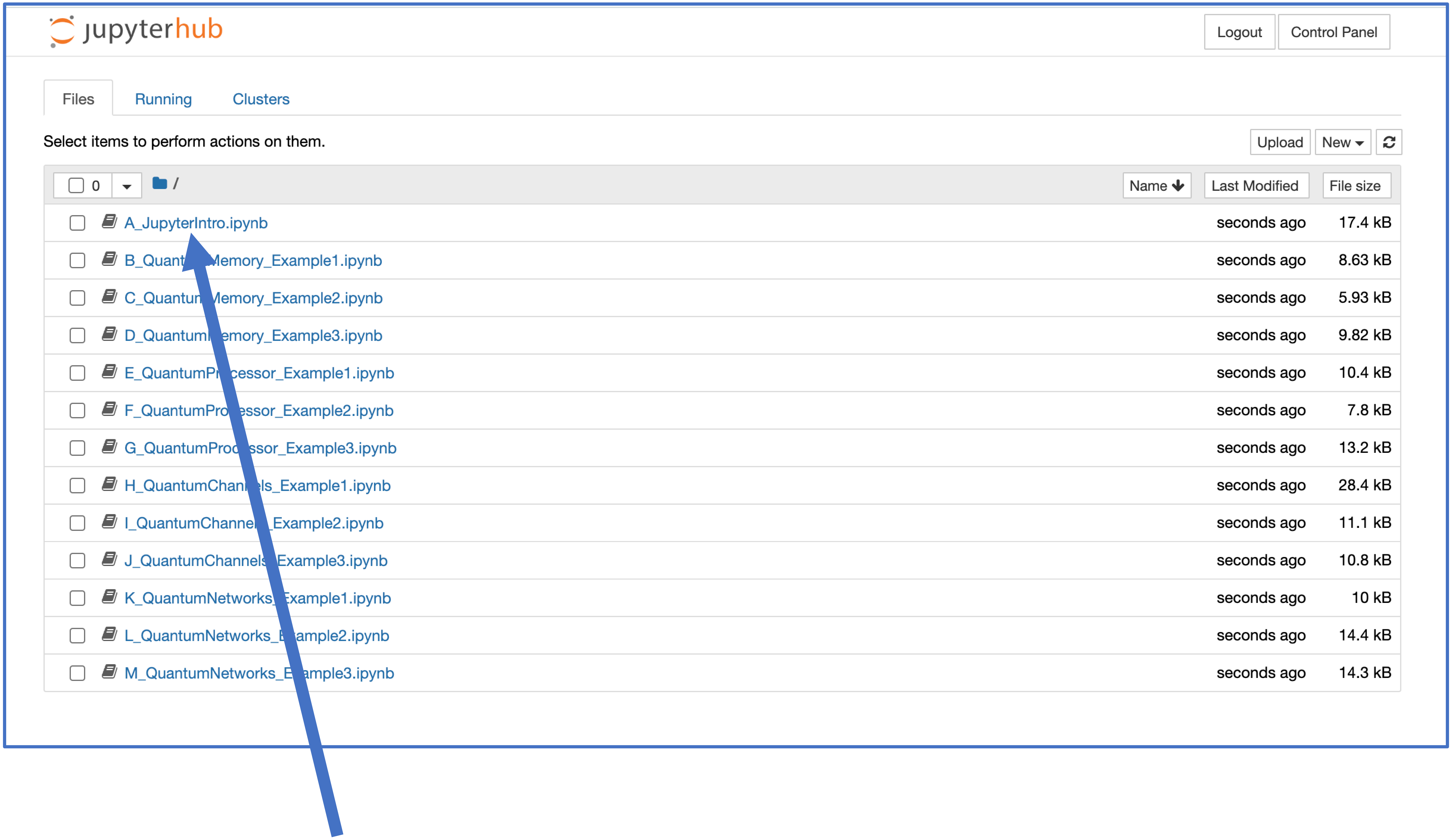


Server not running

Your server is not running. Would you like to start it?

Center for Quantum Networks NSF-ERC

What you should see

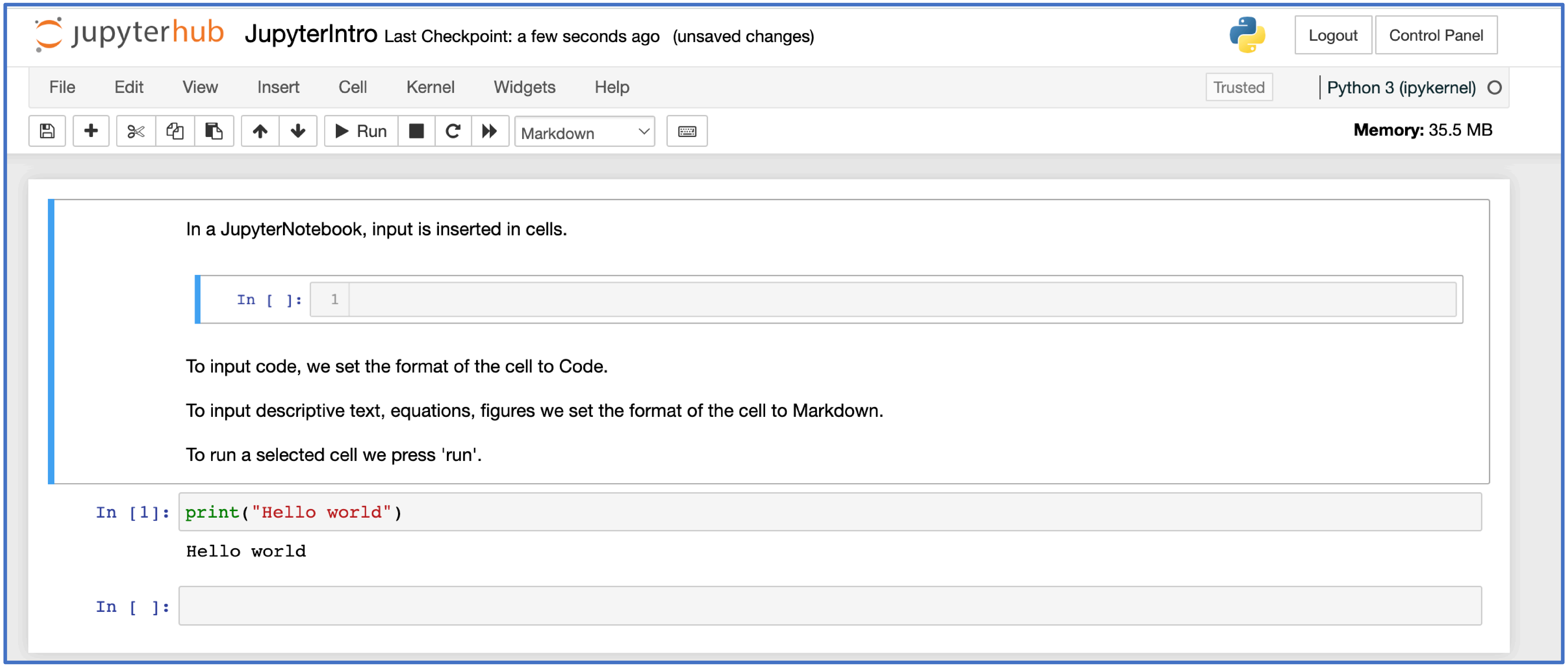


Select items to perform actions on them.

Name	Last Modified	File size
□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□ D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□ E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□ F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
□ G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
□ H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□ I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Click on A_JupyterIntro to open it.

Try it out!



In a JupyterNotebook, input is inserted in cells.

```
In [ ]:
```

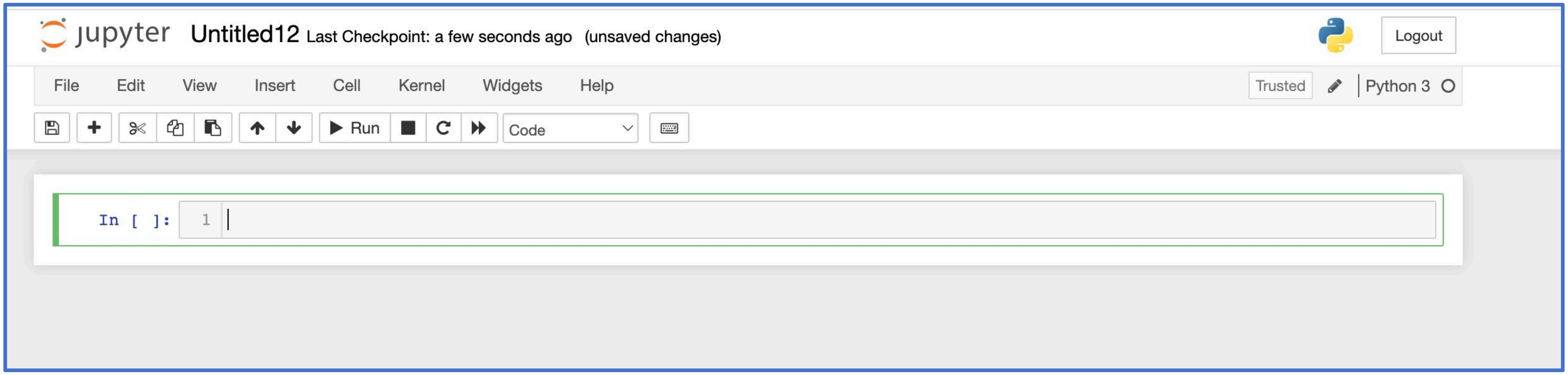
To input code, we set the format of the cell to Code.

To input descriptive text, equations, figures we set the format of the cell to Markdown.

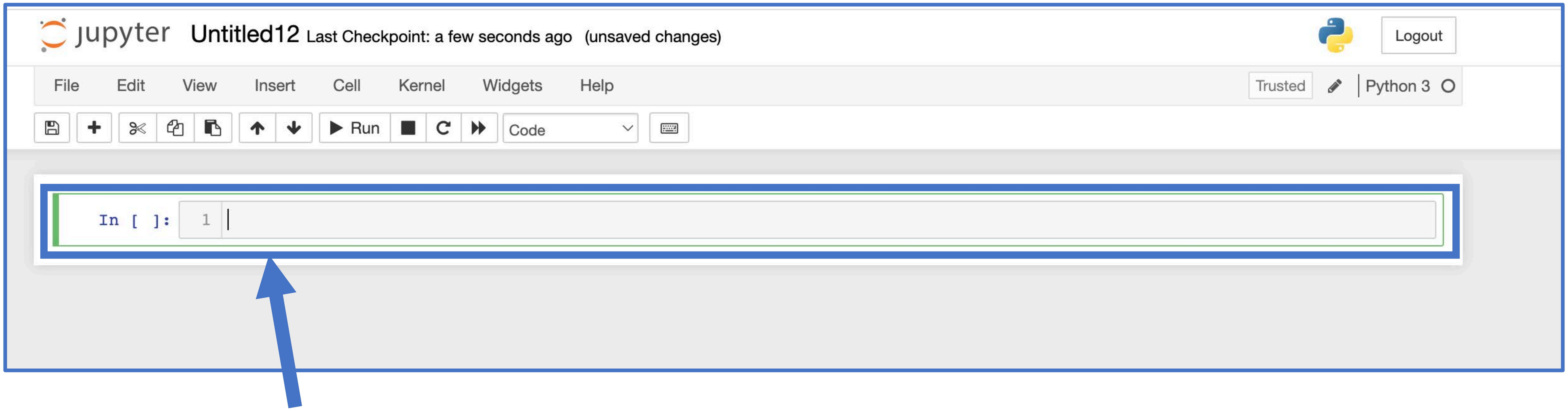
To run a selected cell we press 'run'.

```
In [1]: print("Hello world")

        Hello world

In [ ]:
```

Quick intro to Jupyter Notebooks

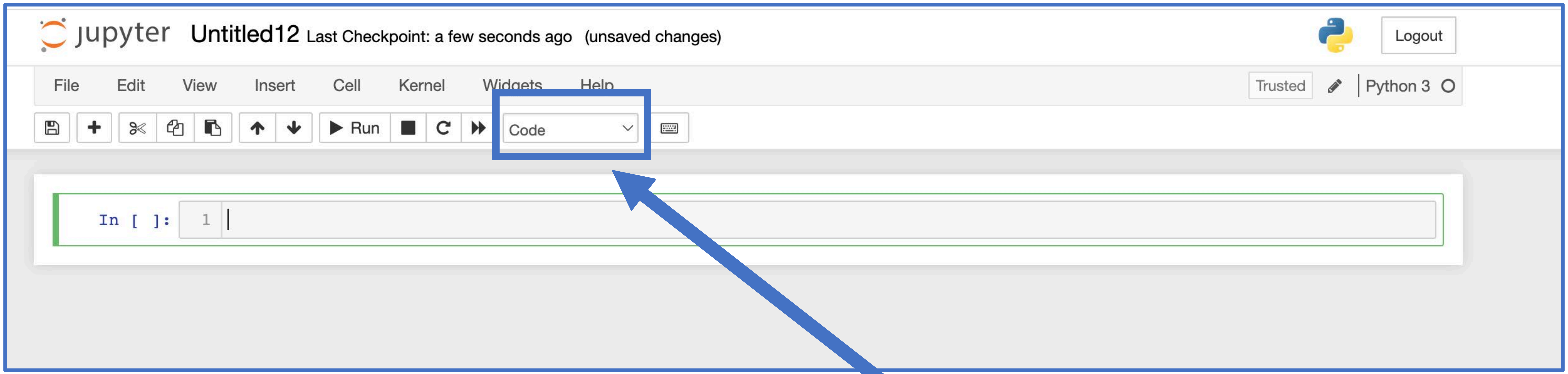


Quick intro to Jupyter Notebooks

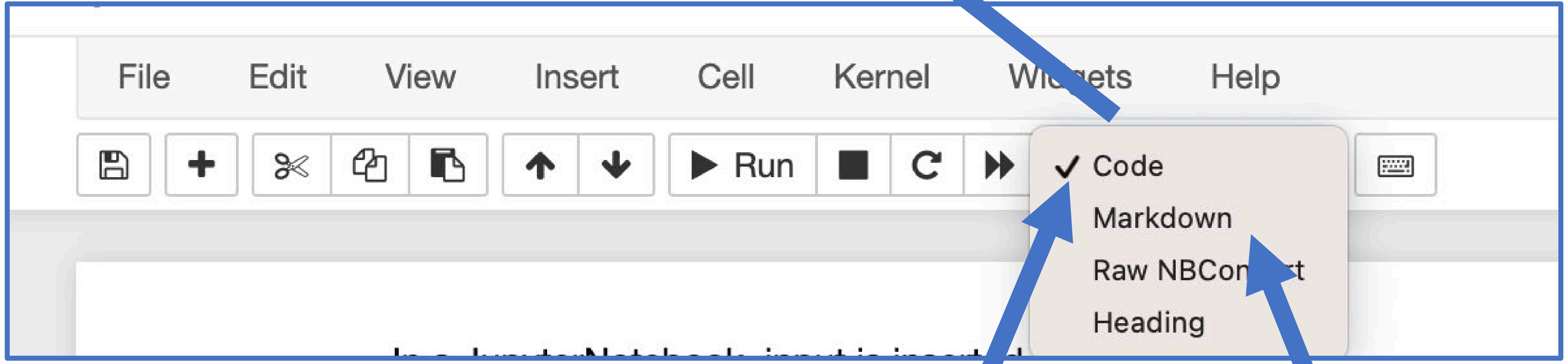


- Input is inserted in cells

Quick intro to Jupyter Notebooks

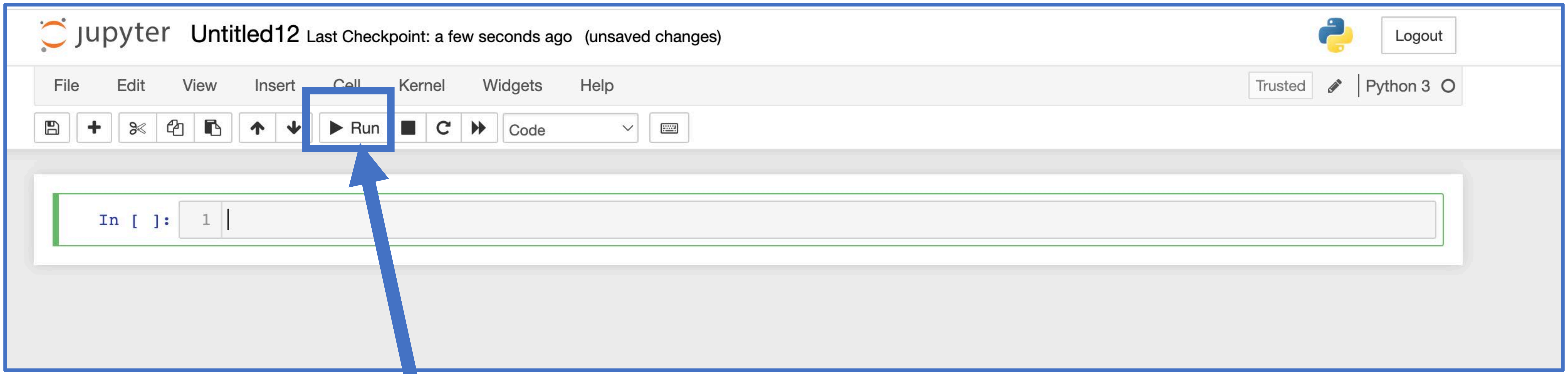


- Input is inserted in cells



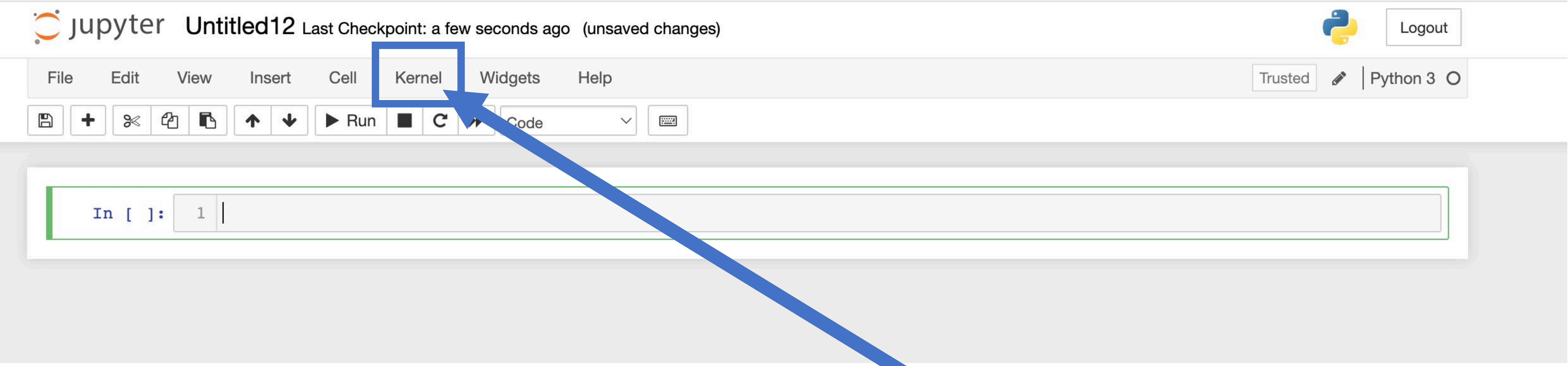
- To input code, we set the format of the cell to **Code**
- To input text, equations, figures etc., we set the format to **Markdown**

Quick intro to Jupyter Notebooks

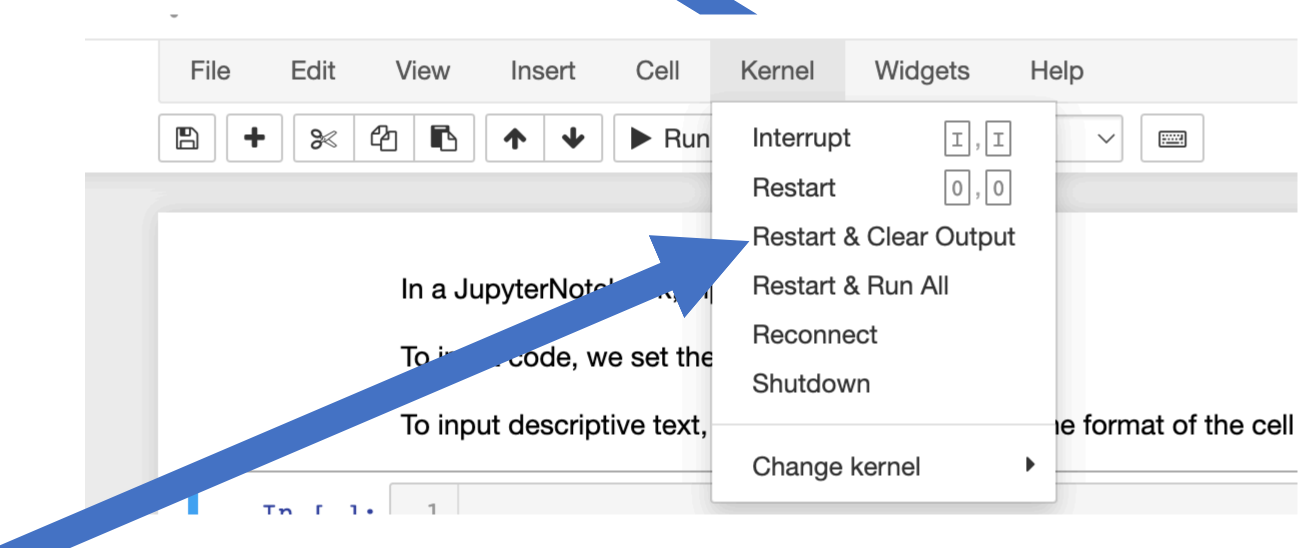


- Input is inserted in cells
- To input code, we set the format of the cell to **Code**
- To input text, equations, figures etc., we set the format to **Markdown**
- To run a selected cell

Quick intro to Jupyter Notebooks



- Input is inserted in cells
- To input code, we set the format of the cell to **Code**
- To input text, equations, figures etc., we set the format to **Markdown**
- To run a selected cell



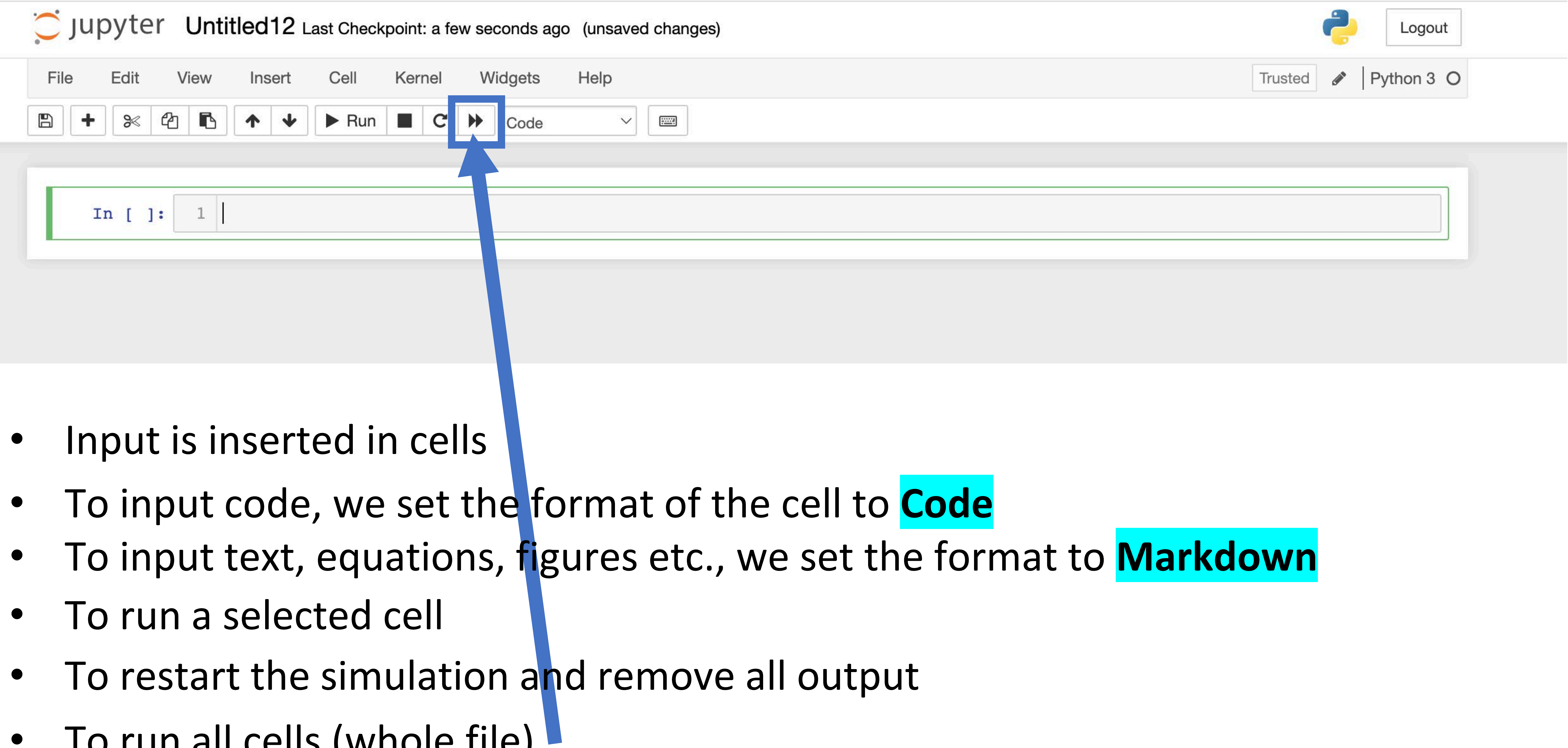
In a JupyterNote

To i code, we set the

To input descriptive text, e format of the cell

- To restart the simulation and remove all output

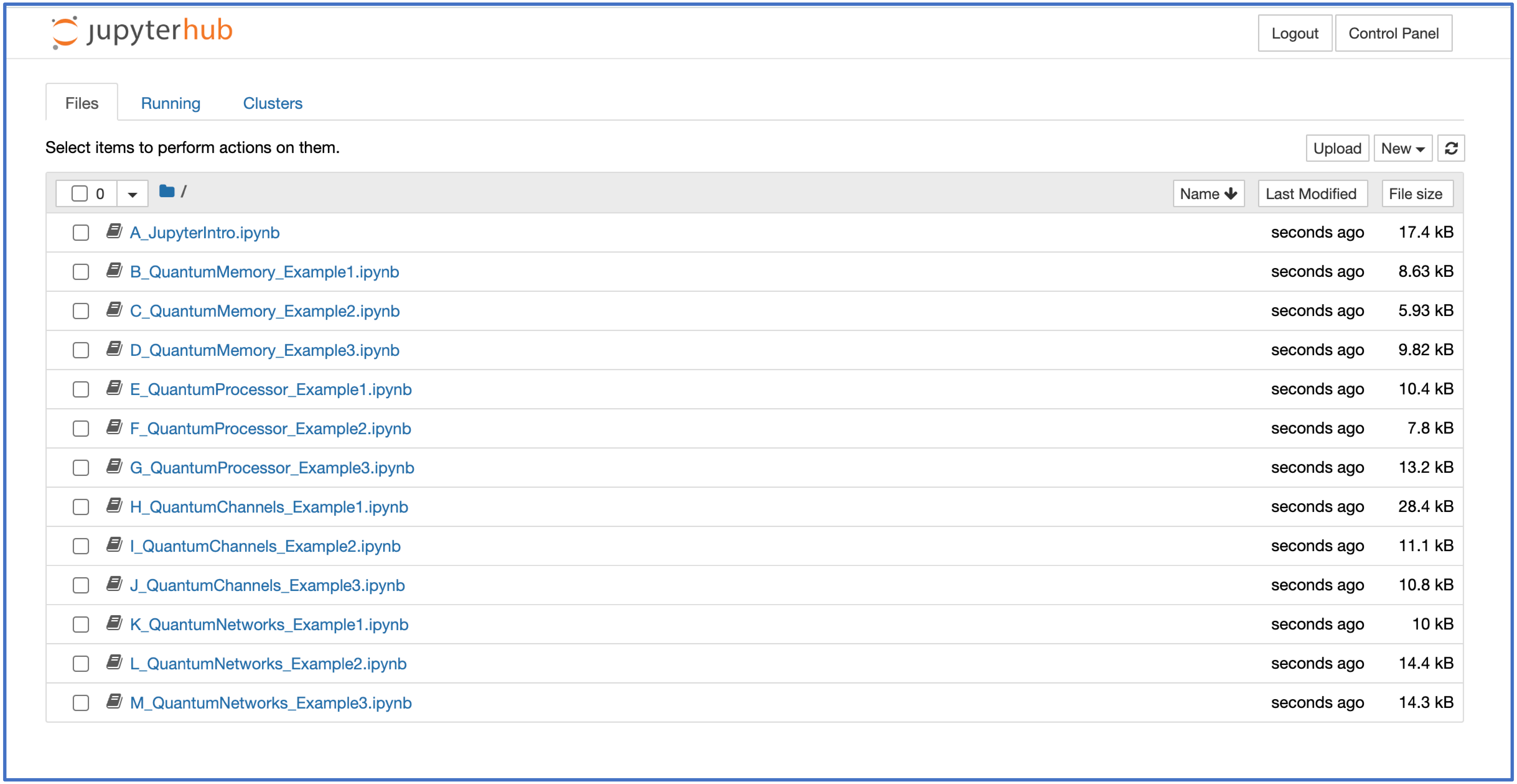
Quick intro to Jupyter Notebooks



- Input is inserted in cells
- To input code, we set the format of the cell to **Code**
- To input text, equations, figures etc., we set the format to **Markdown**
- To run a selected cell
- To restart the simulation and remove all output
- To run all cells (whole file)

Simulation Deep Dive: Learning by Doing

Hands-on Explorations

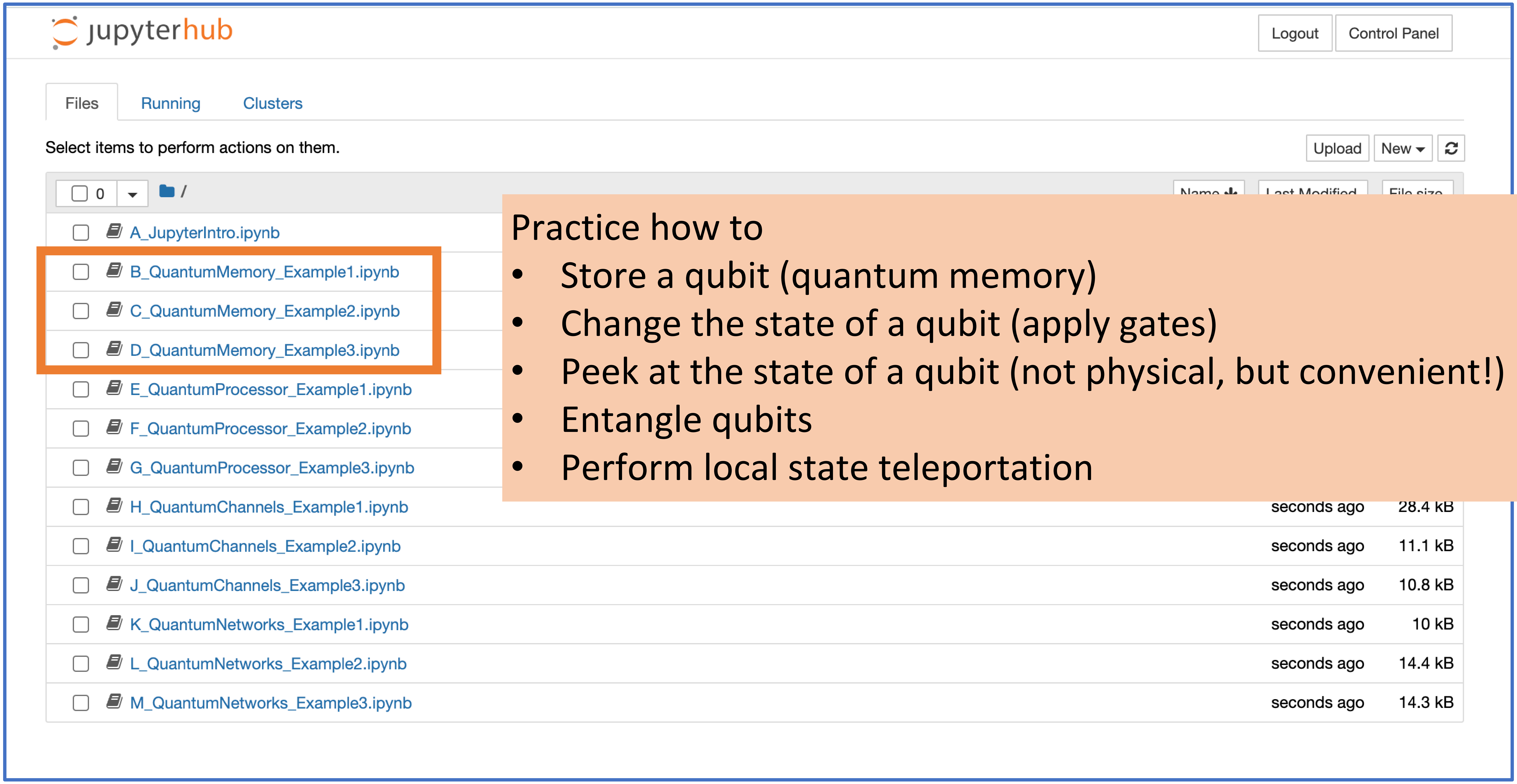


Select items to perform actions on them.

□ 0 /	Name	Last Modified	File size
□ A_JupyterIntro.ipynb		seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb		seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb		seconds ago	5.93 kB
□ D_QuantumMemory_Example3.ipynb		seconds ago	9.82 kB
□ E_QuantumProcessor_Example1.ipynb		seconds ago	10.4 kB
□ F_QuantumProcessor_Example2.ipynb		seconds ago	7.8 kB
□ G_QuantumProcessor_Example3.ipynb		seconds ago	13.2 kB
□ H_QuantumChannels_Example1.ipynb		seconds ago	28.4 kB
□ I_QuantumChannels_Example2.ipynb		seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb		seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb		seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb		seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb		seconds ago	14.3 kB

Simulation Deep Dive: Learning by Doing

Hands-on Explorations



Select items to perform actions on them.

□ 0 /

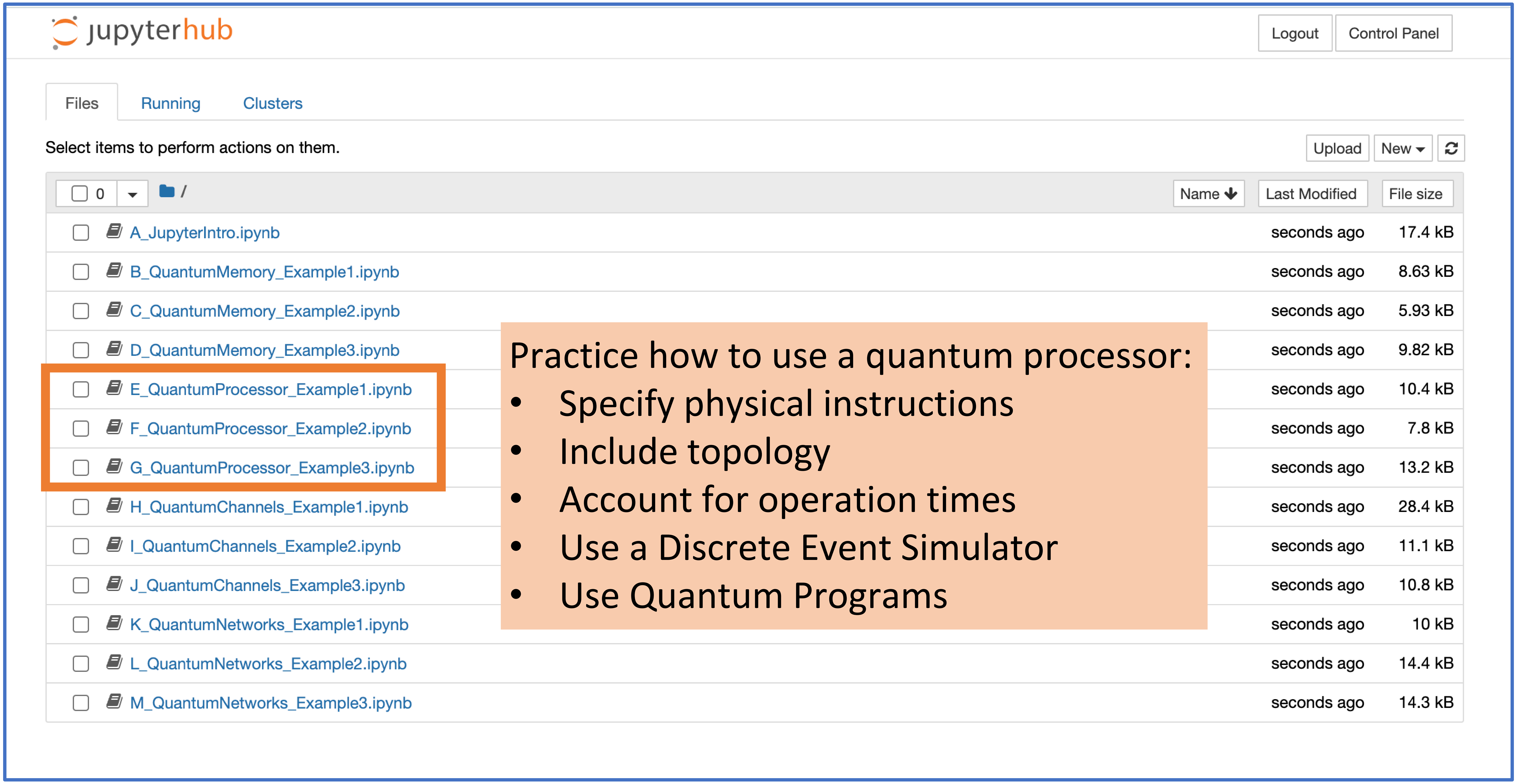
	Name	Last Modified	File size
□	A_JupyterIntro.ipynb		
□	B_QuantumMemory_Example1.ipynb		
□	C_QuantumMemory_Example2.ipynb		
□	D_QuantumMemory_Example3.ipynb		
□	E_QuantumProcessor_Example1.ipynb		
□	F_QuantumProcessor_Example2.ipynb		
□	G_QuantumProcessor_Example3.ipynb		
□	H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□	I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□	J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□	K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□	L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□	M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Practice how to

- Store a qubit (quantum memory)
- Change the state of a qubit (apply gates)
- Peek at the state of a qubit (not physical, but convenient!)
- Entangle qubits
- Perform local state teleportation

Simulation Deep Dive: Learning by Doing

Hands-on Explorations



Select items to perform actions on them.

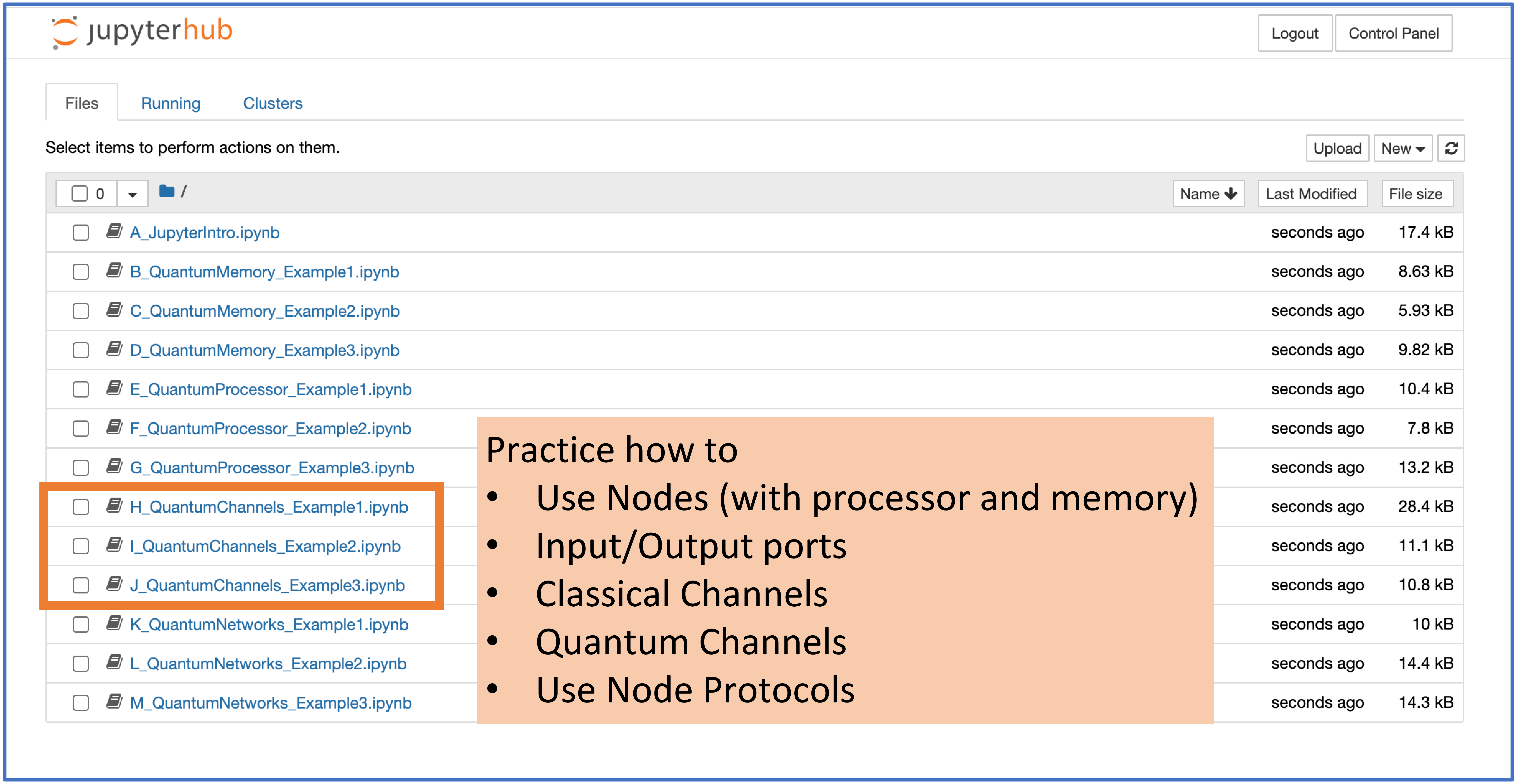
Name	Last Modified	File size
□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□ D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□ E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□ F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
□ G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
□ H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□ I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Practice how to use a quantum processor:

- Specify physical instructions
- Include topology
- Account for operation times
- Use a Discrete Event Simulator
- Use Quantum Programs

Simulation Deep Dive: Learning by Doing

Hands-on Explorations



Select items to perform actions on them.

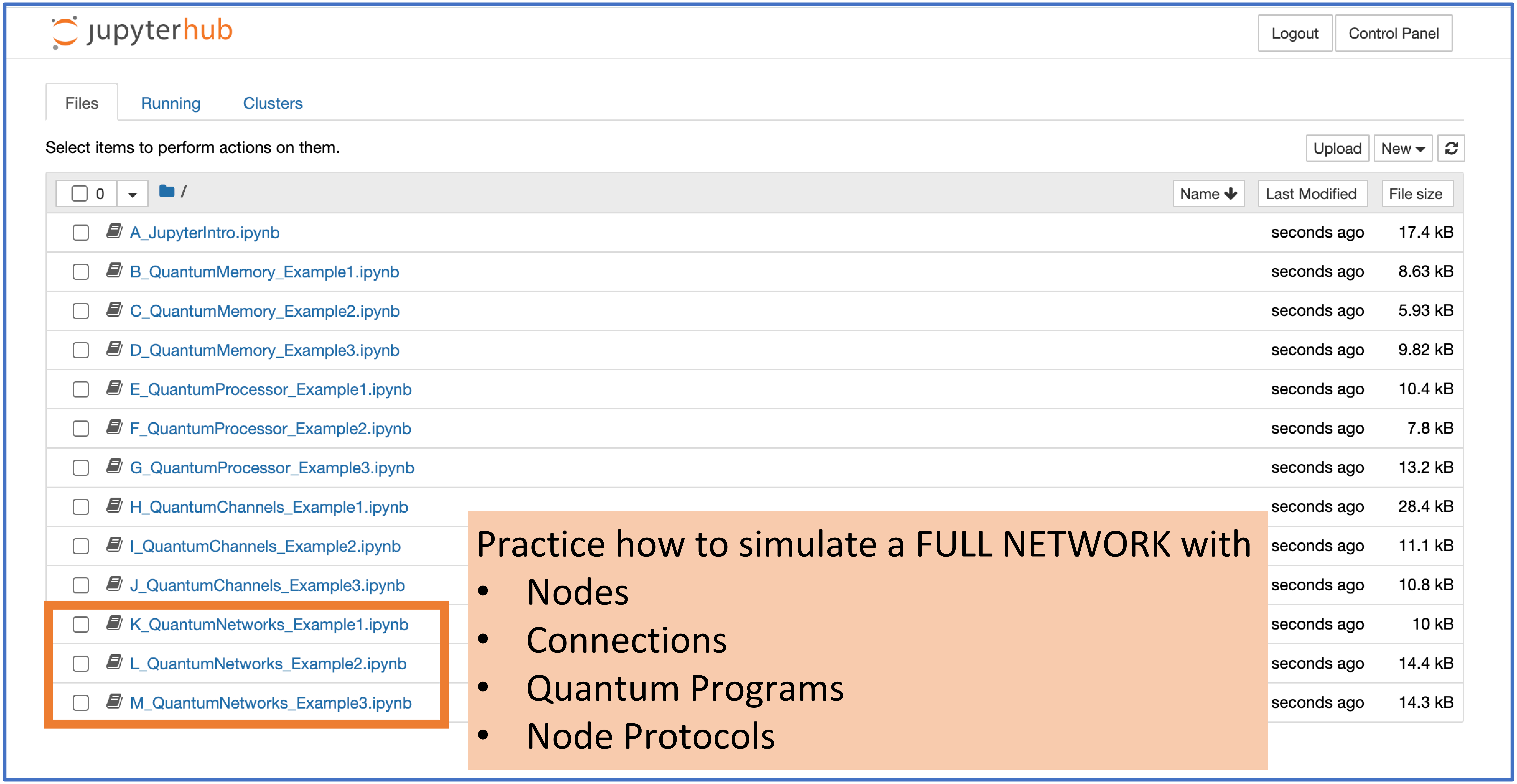
Name	Last Modified	File size
□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□ D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□ E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□ F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
□ G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
□ H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□ I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Practice how to

- Use Nodes (with processor and memory)
- Input/Output ports
- Classical Channels
- Quantum Channels
- Use Node Protocols

Simulation Deep Dive: Learning by Doing

Hands-on Explorations



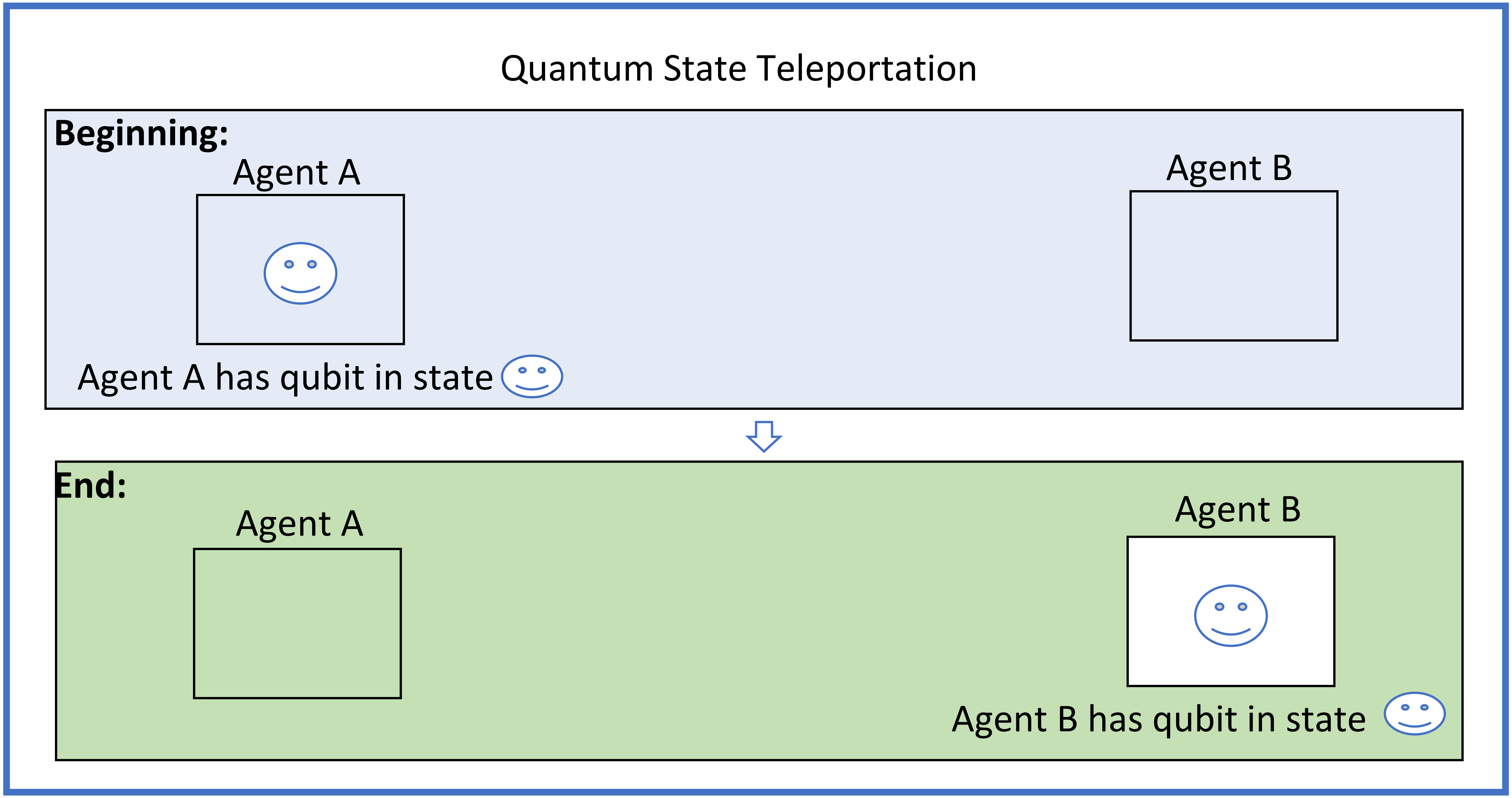
Select items to perform actions on them.

Name	Last Modified	File size
A_JupyterIntro.ipynb	seconds ago	17.4 kB
B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Practice how to simulate a FULL NETWORK with

- Nodes
- Connections
- Quantum Programs
- Node Protocols

Quantum Network Simulators - Example application:



Quantum State Teleportation

Beginning:

Agent A

Agent B

Agent A has qubit in state ☺

End:

Agent A

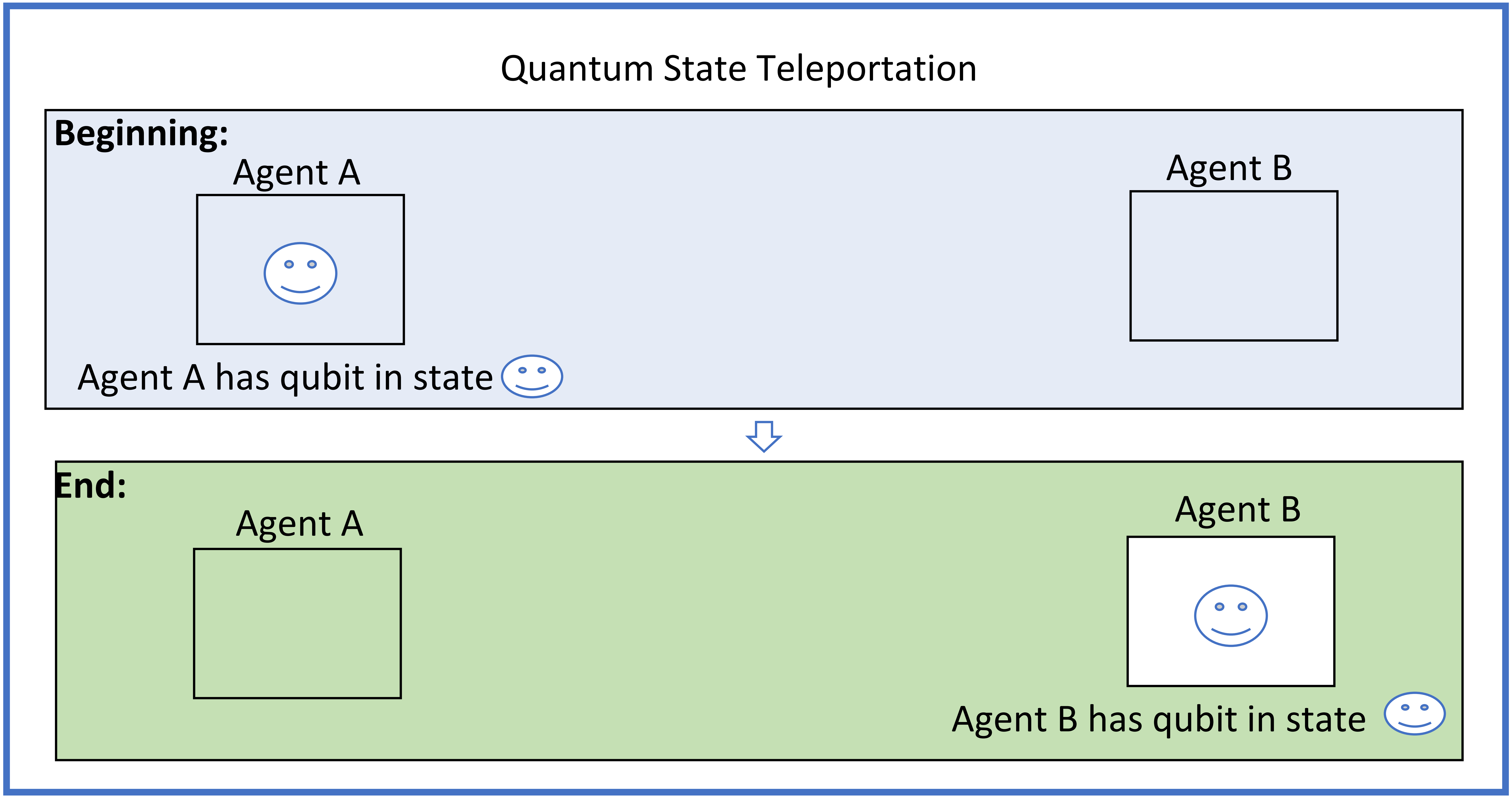
Agent B

Agent B has qubit in state ☺

Center for Quantum Networks *NSF-ERC*

Quantum State Teleportation - Refresher

Goal: Teleport state of qubit from Agent A to Agent B



Quantum State Teleportation

Beginning:

Agent A

Agent B

Agent A has qubit in state ☺

End:

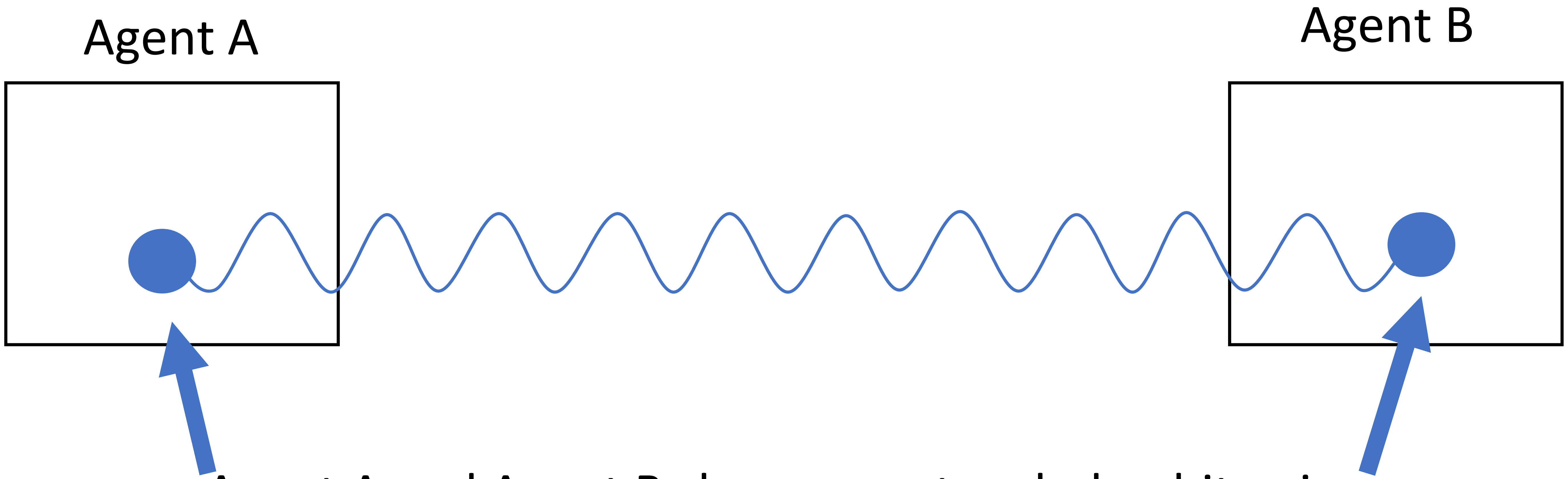
Agent A

Agent B

Agent B has qubit in state ☺

Quantum State Teleportation - Refresher

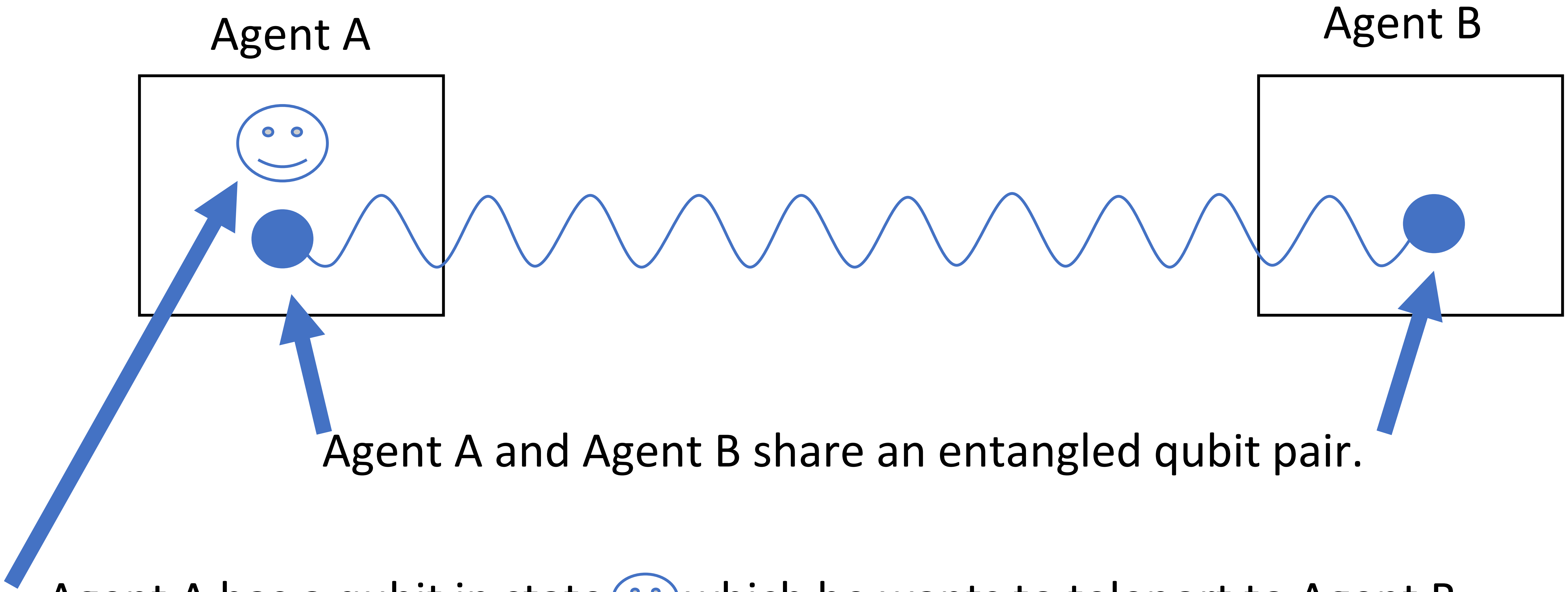
Goal: Teleport state of qubit from Agent A to Agent B



Agent A and Agent B share an entangled qubit pair.

Quantum State Teleportation - Refresher

Goal: Teleport state of qubit from Agent A to Agent B

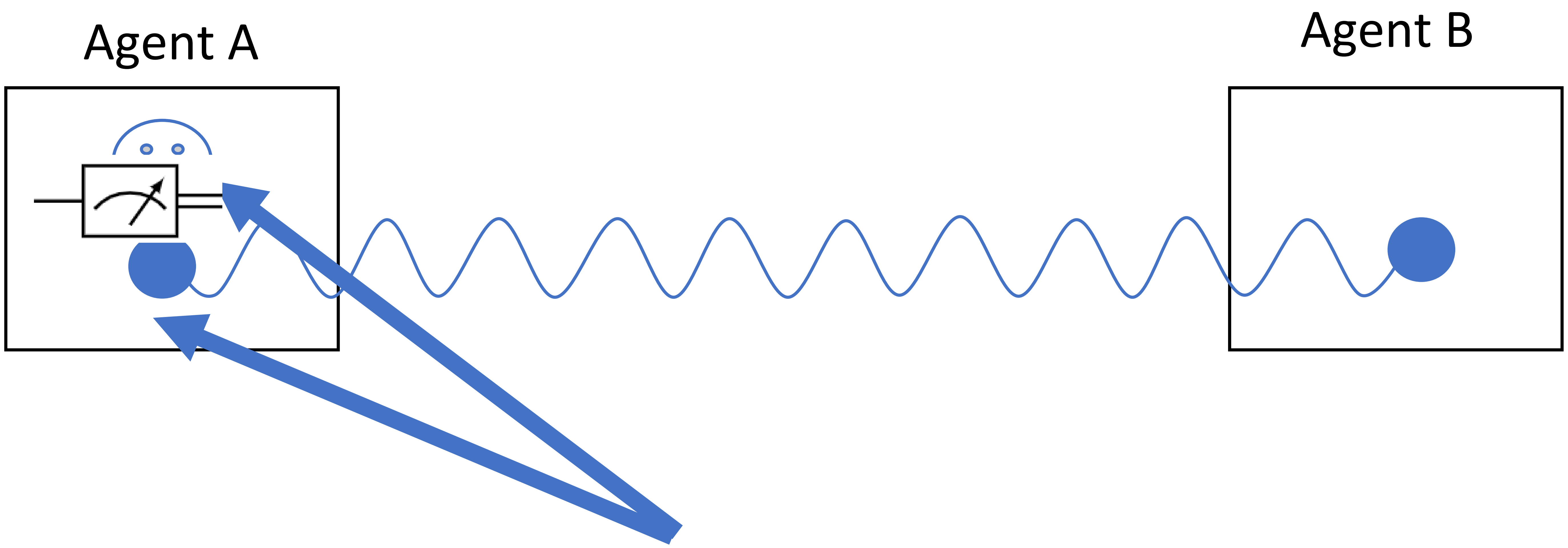


Agent A and Agent B share an entangled qubit pair.

Agent A has a qubit in state ☺ which he wants to teleport to Agent B.

Quantum State Teleportation - Refresher

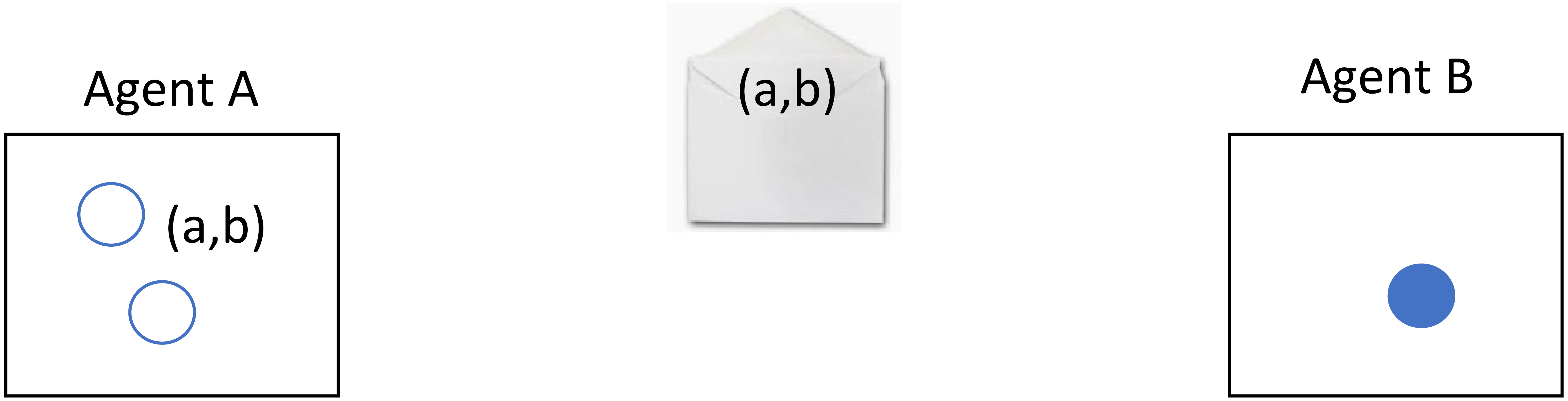
Goal: Teleport state of qubit from Agent A to Agent B



Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

Quantum State Teleportation - Refresher

Goal: Teleport state of qubit from Agent A to Agent B

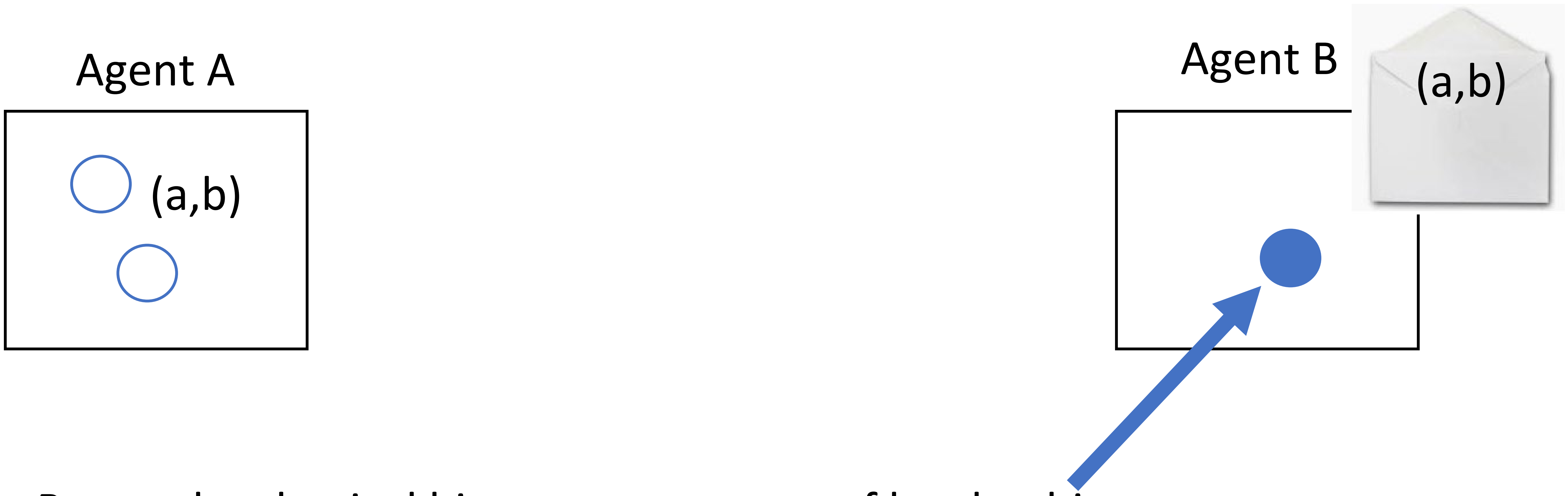


Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits. ☑

Quantum State Teleportation - Refresher

Goal: Teleport state of qubit from Agent A to Agent B



Agent A

(a,b)

Agent B

(a,b)

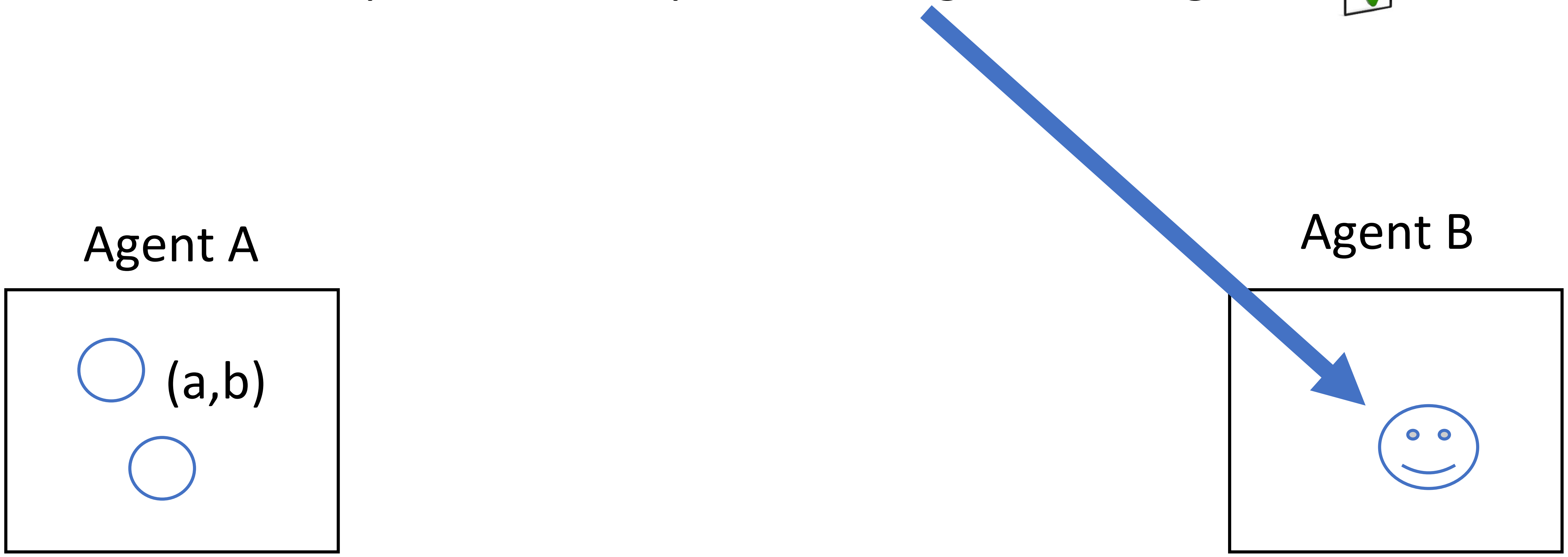
Step 3: Agent B uses the classical bits to correct state of local qubit.

Step 2: Agent A send the measurement outcomes to Agent B (classical bits). ☑

Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits. ☑

Quantum State Teleportation - Refresher

Goal: Teleport state of qubit from Agent A to Agent B ☑



Step 3: Agent B uses the classical bits to correct state of local qubit. ☑

Step 2: Agent A send the measurement outcomes to Agent B (classical bits). ☑

Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits. ☑

Quantum State Teleportation - Refresher

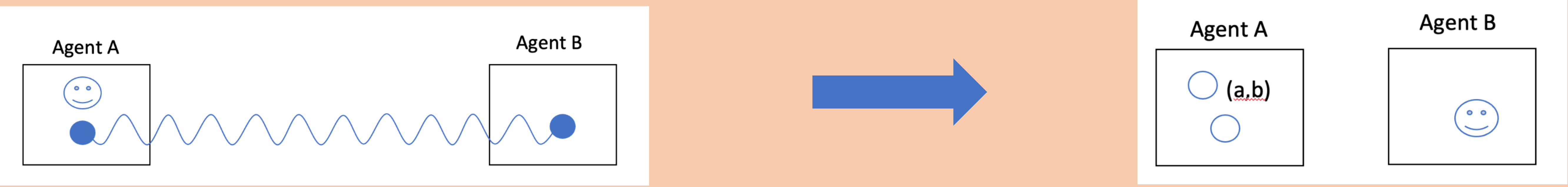
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

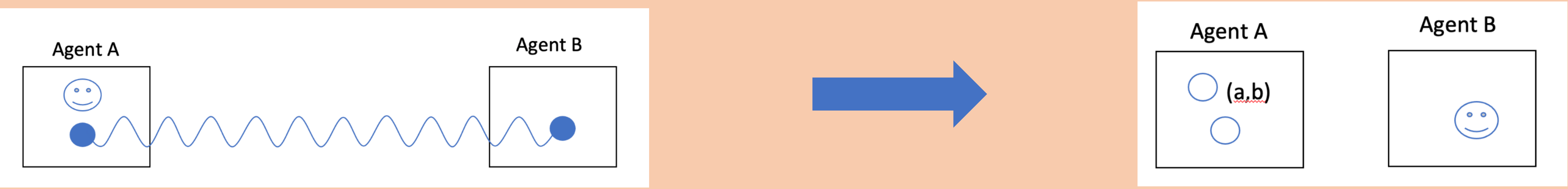
Step 3: Agent B uses the classical bits to correct state of local qubit.



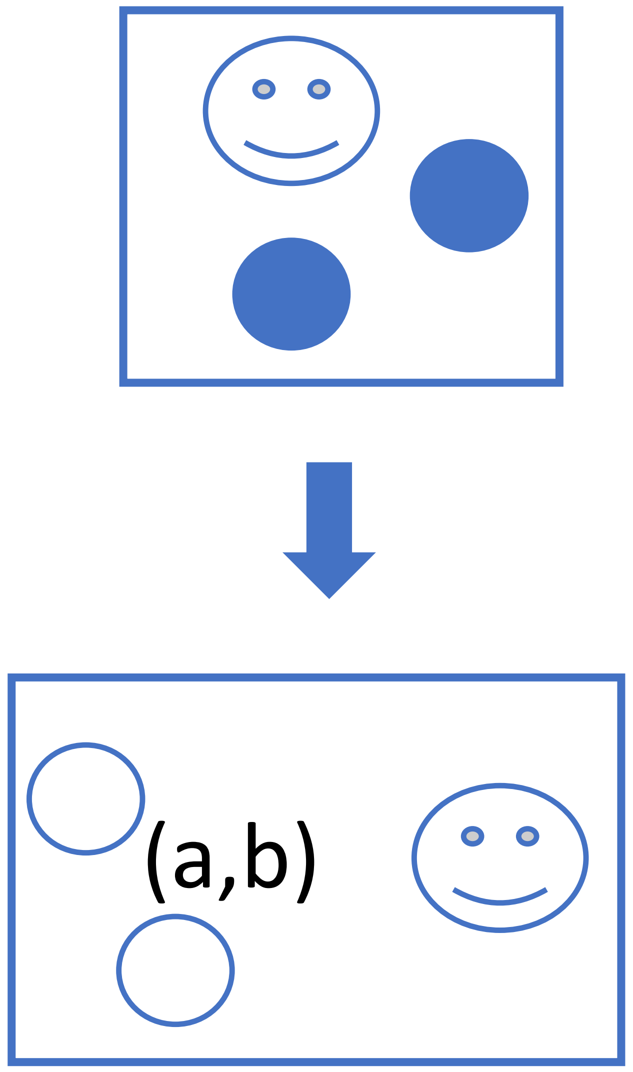
Quantum State Teleportation - Refresher

Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).
Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit



Quantum State Teleportation - Refresher

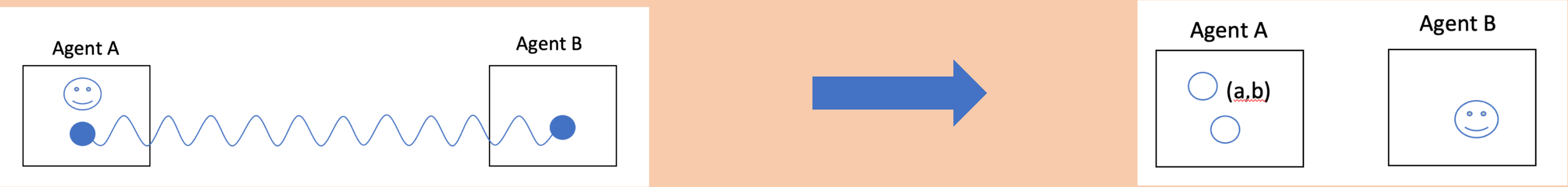
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

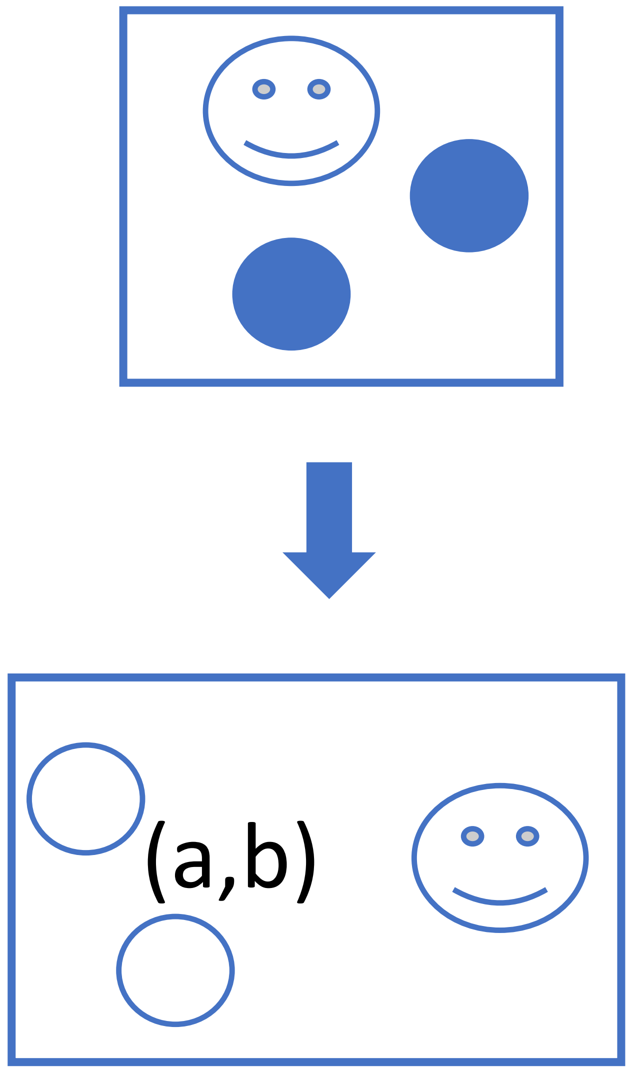
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

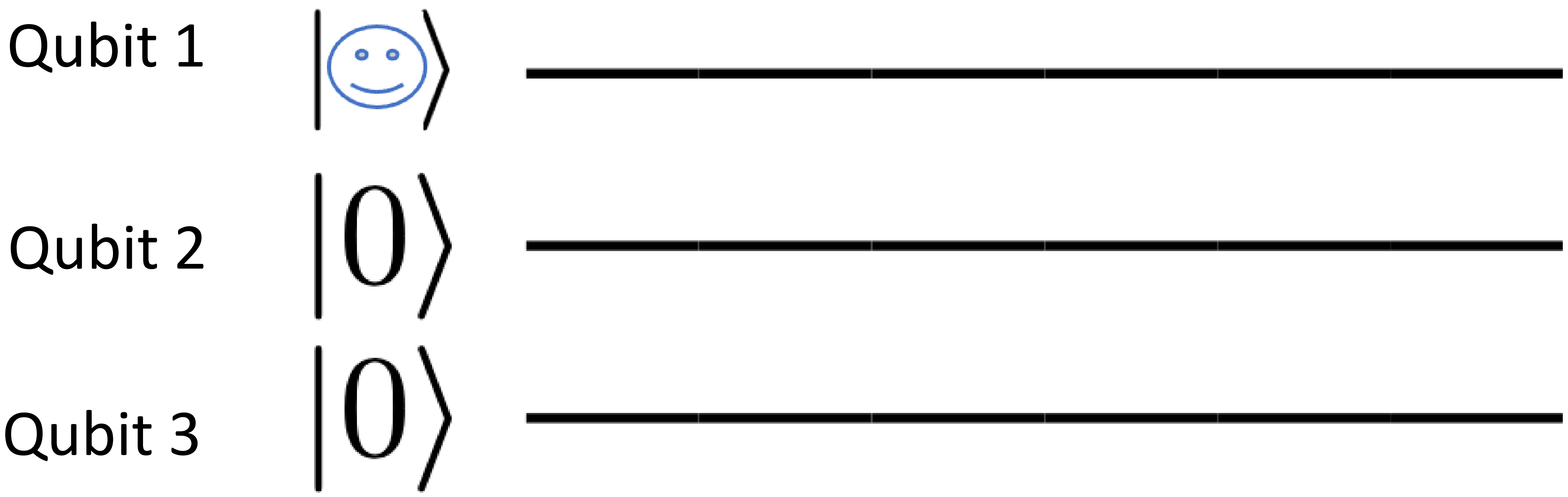
Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit



Step 1: Start with three qubits



Quantum State Teleportation - Refresher

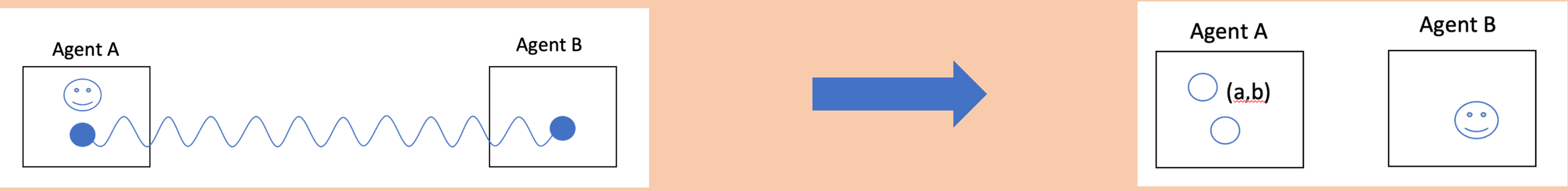
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

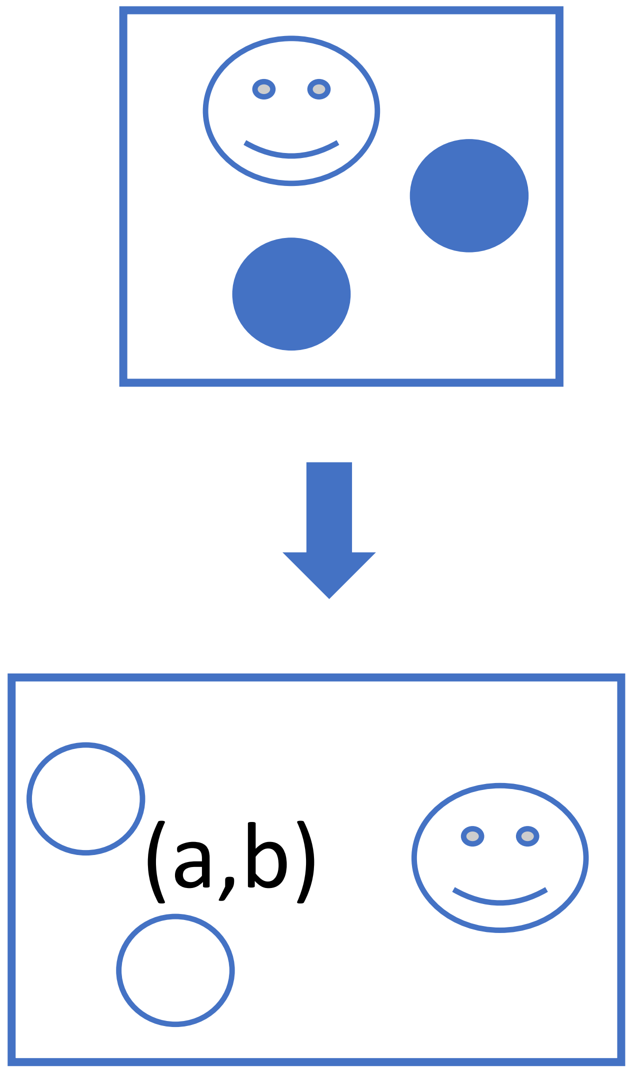
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

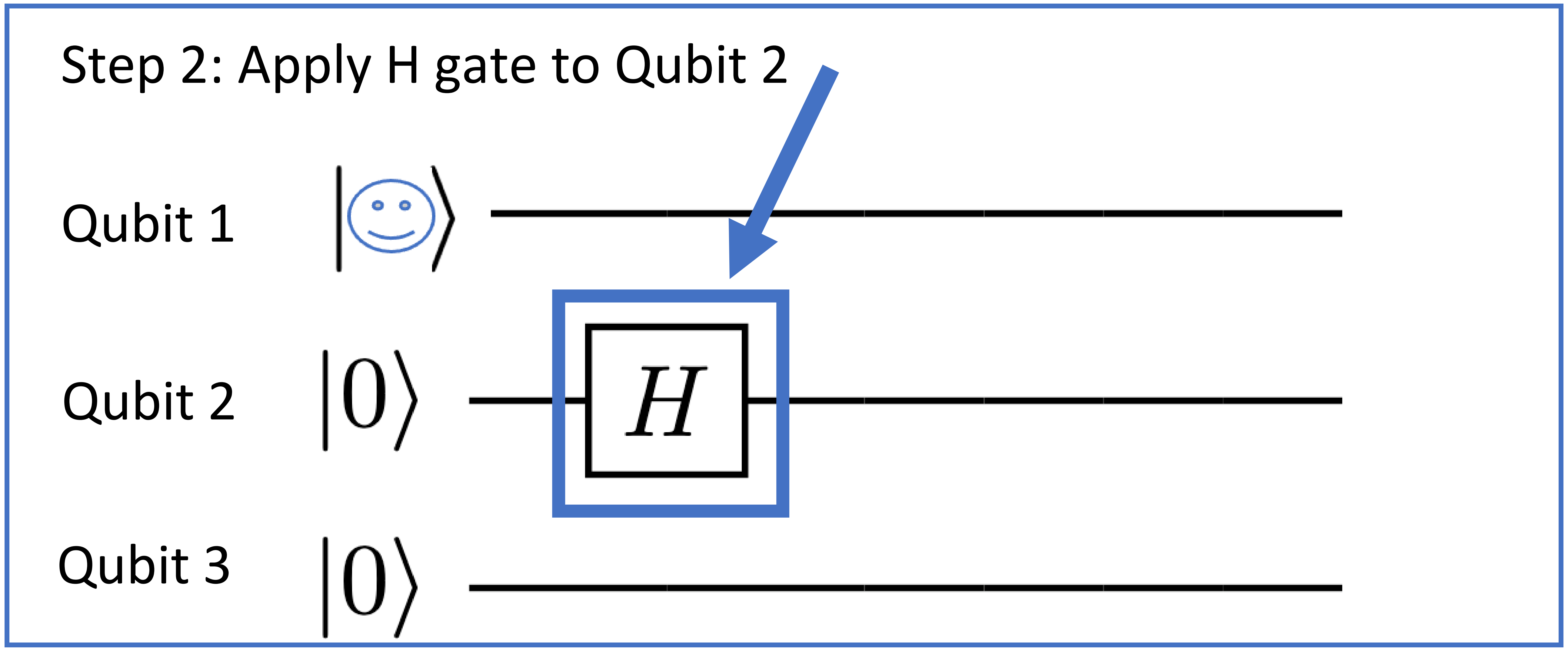
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit

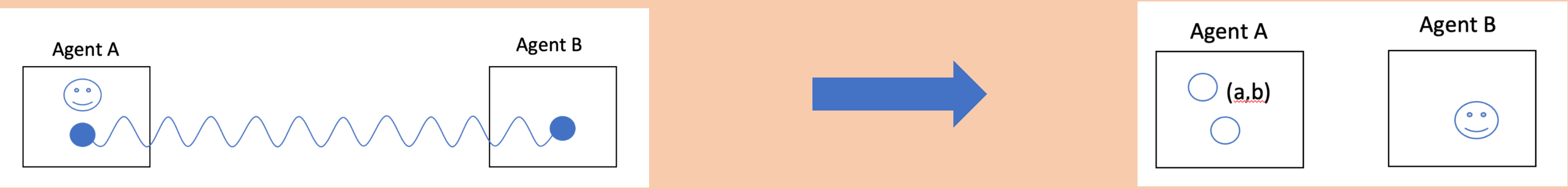



Step 2: Apply H gate to Qubit 2

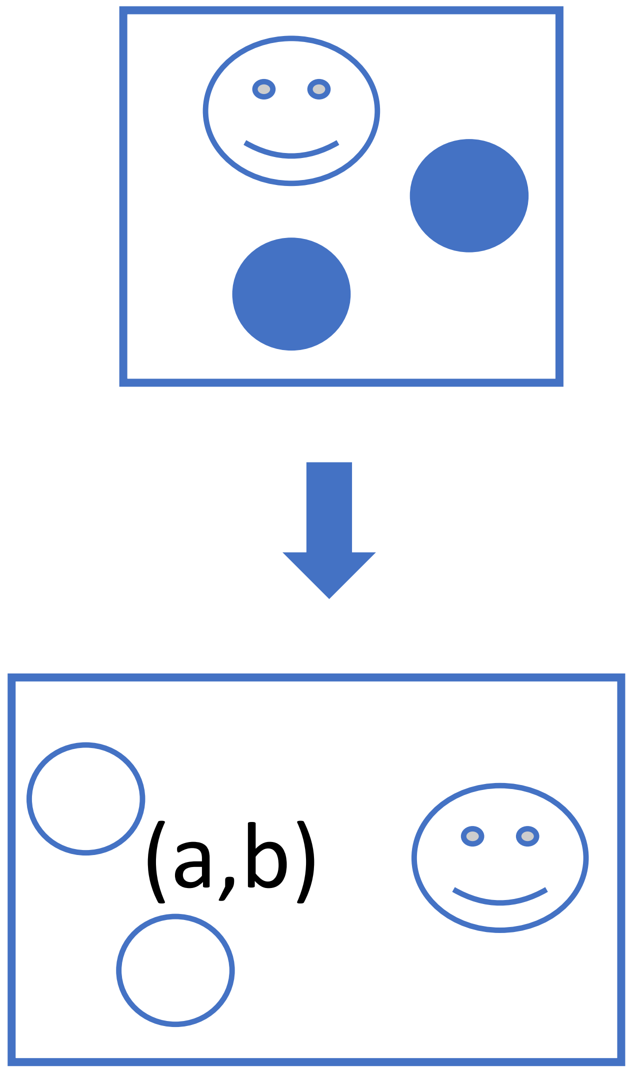
Quantum State Teleportation - Refresher

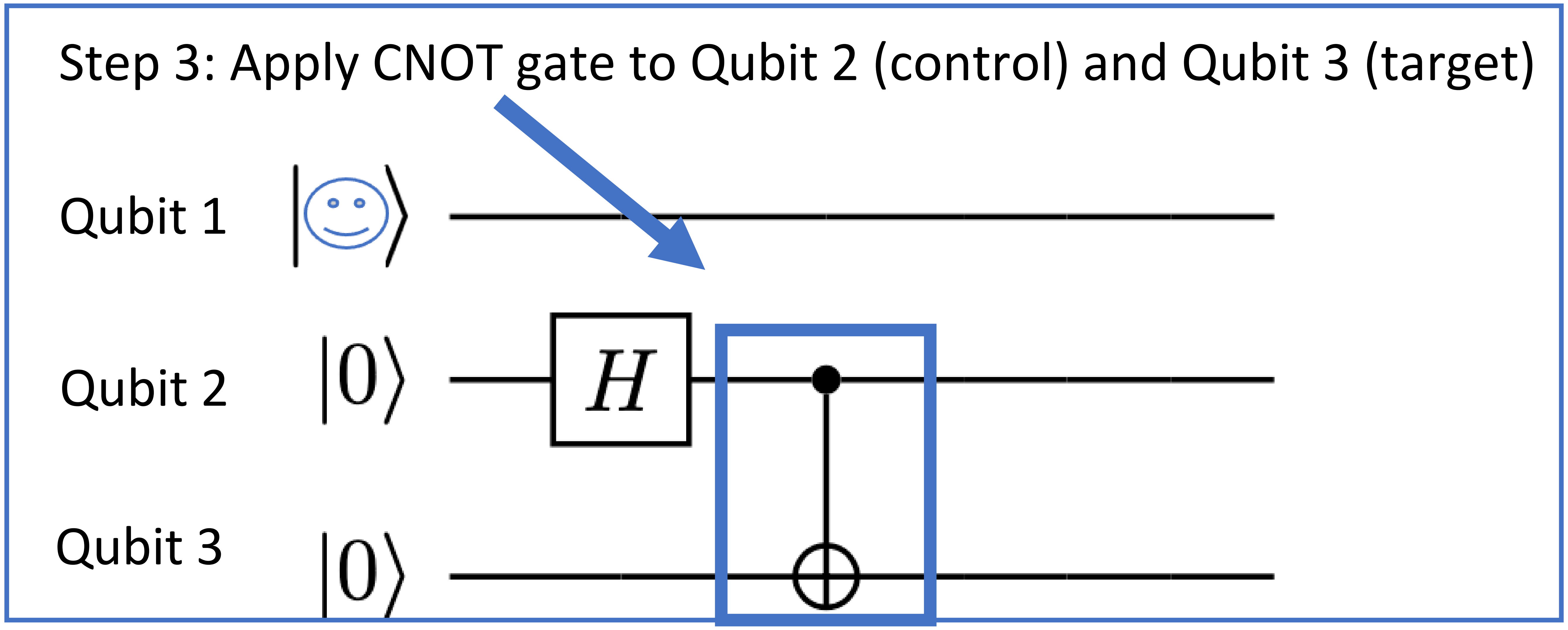
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).
Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit




Step 3: Apply CNOT gate to Qubit 2 (control) and Qubit 3 (target)

Center for Quantum Networks
NSF-ERC

Quantum State Teleportation - Refresher

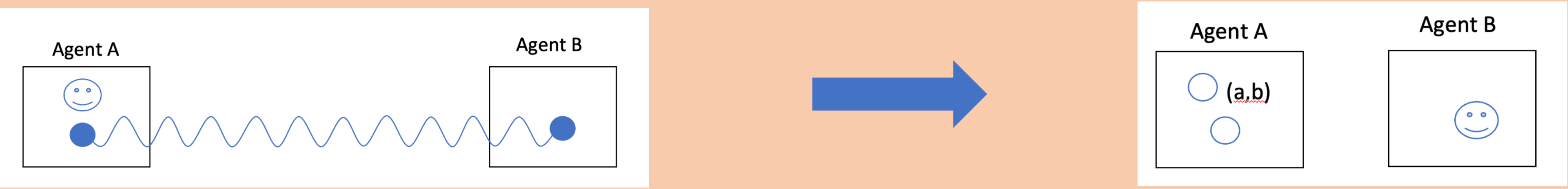
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

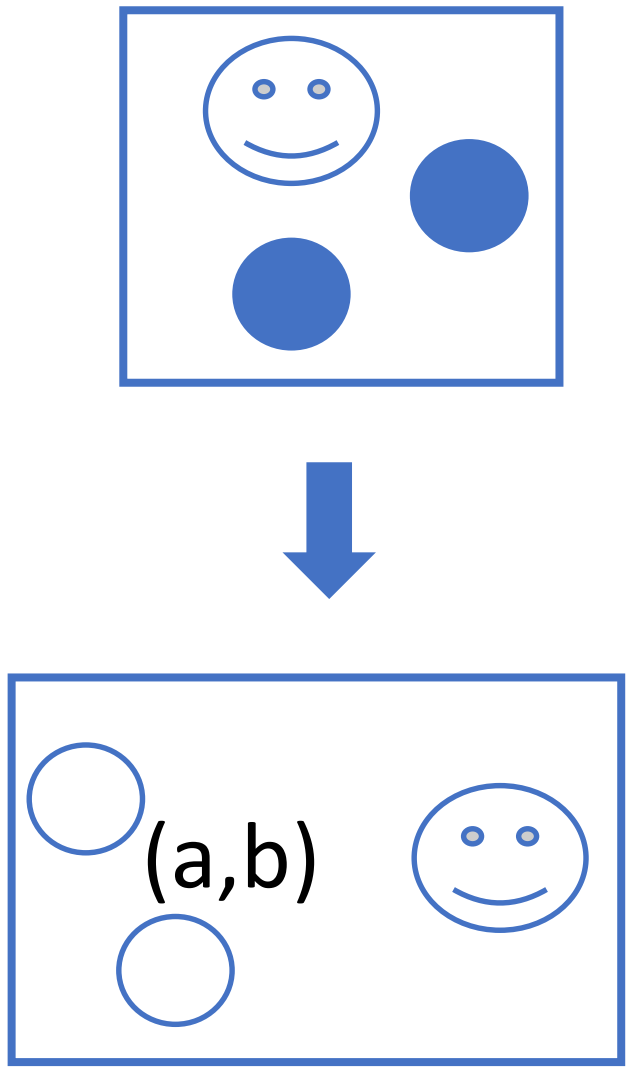
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

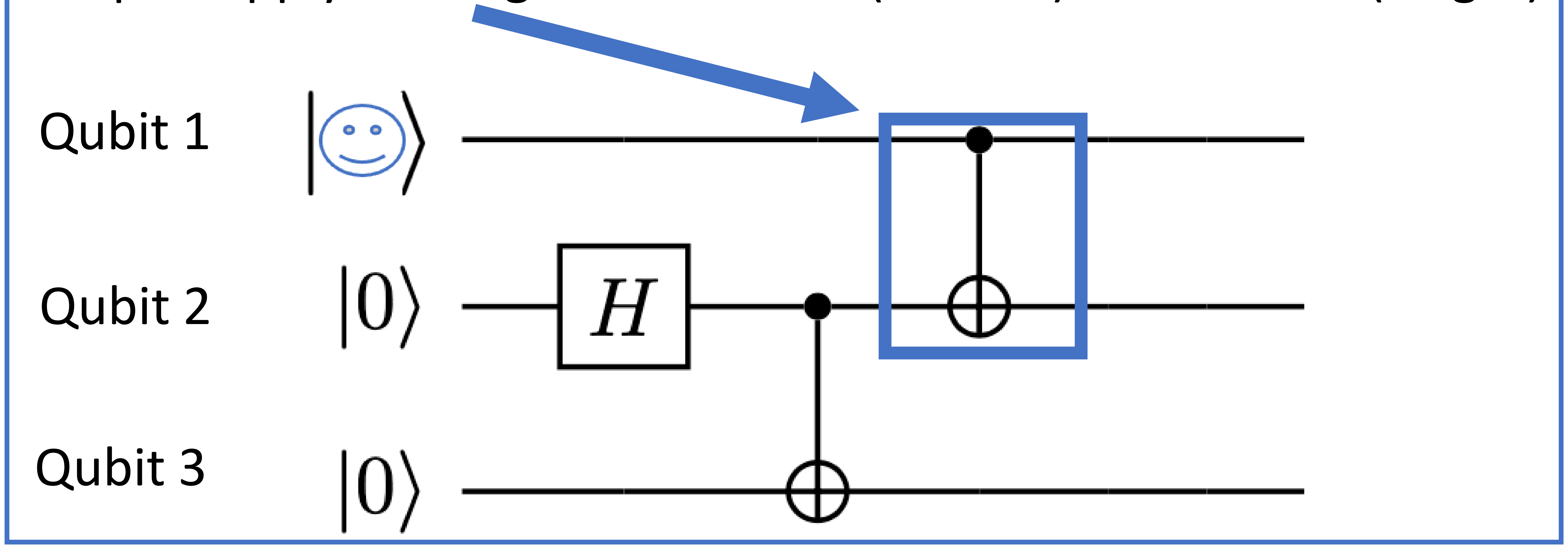
Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit



Step 4: Apply CNOT gate to Qubit 1 (control) and Qubit 2 (target)



Quantum State Teleportation - Refresher

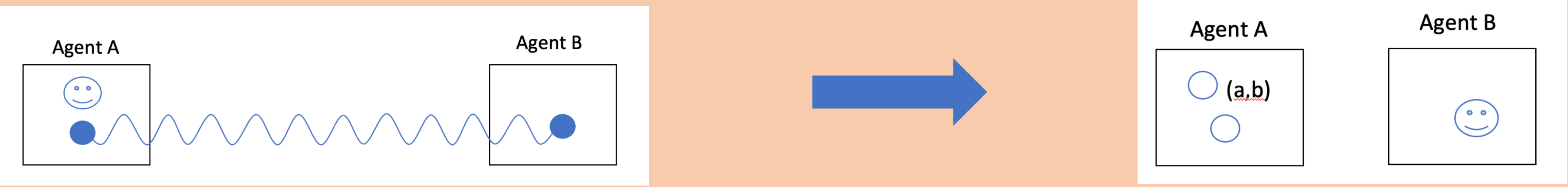
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

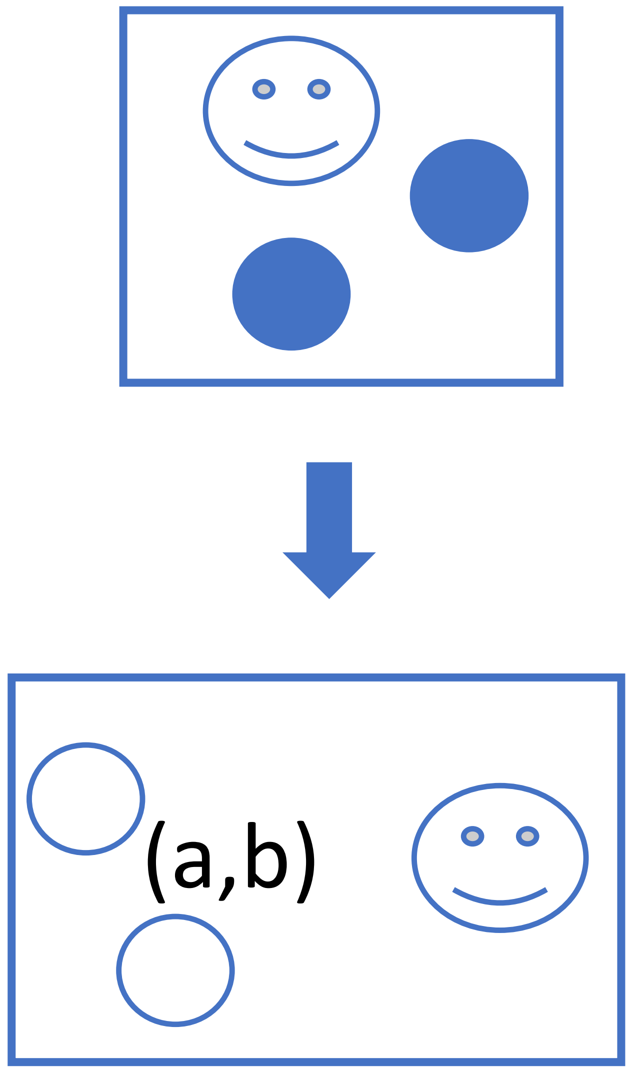
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

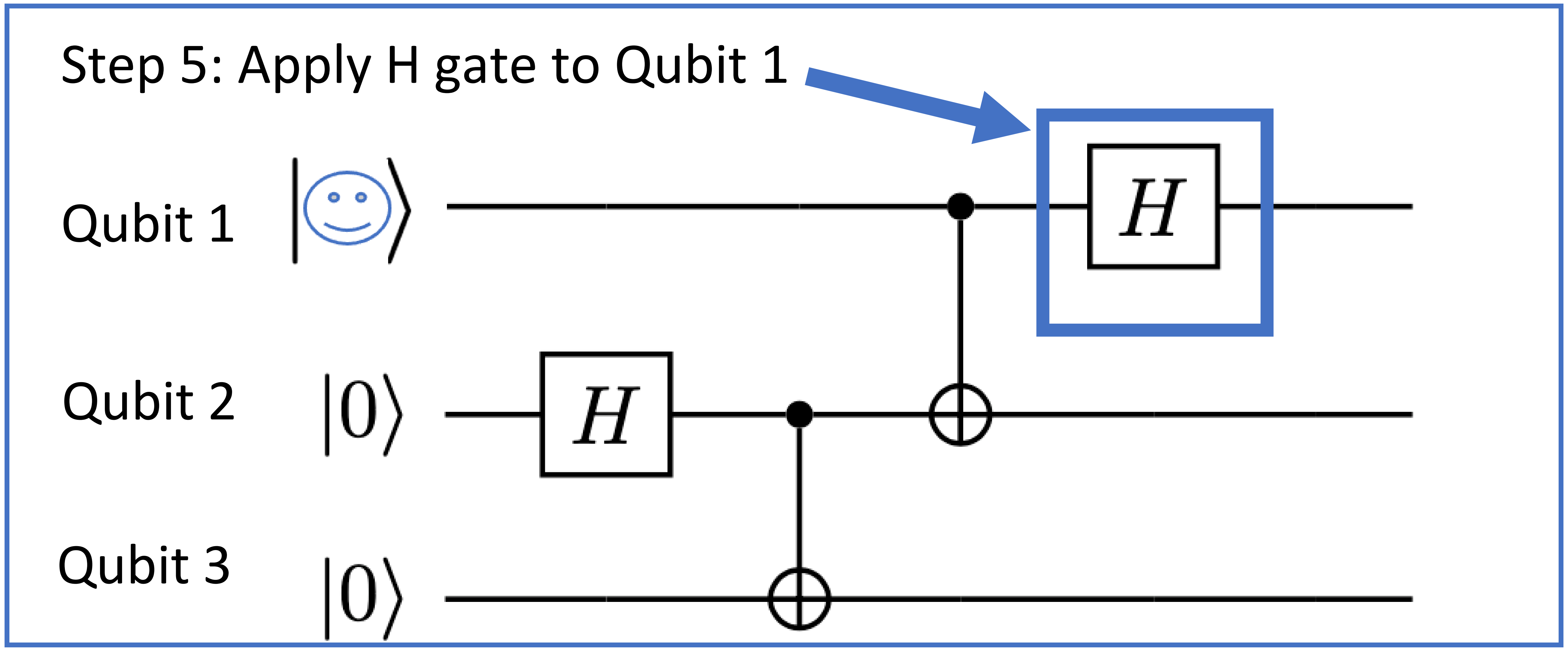
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit





Step 5: Apply H gate to Qubit 1

Quantum State Teleportation - Refresher

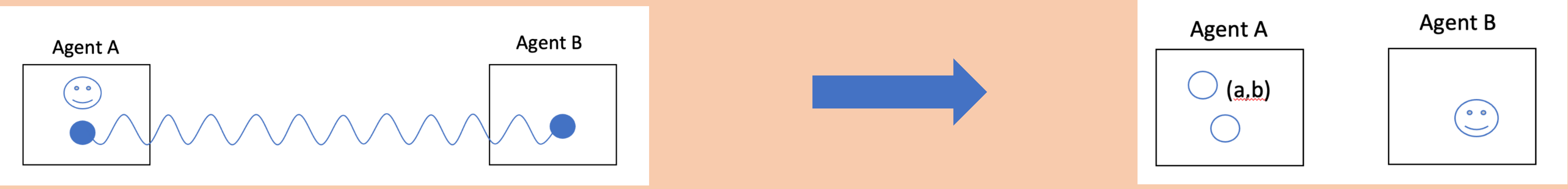
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

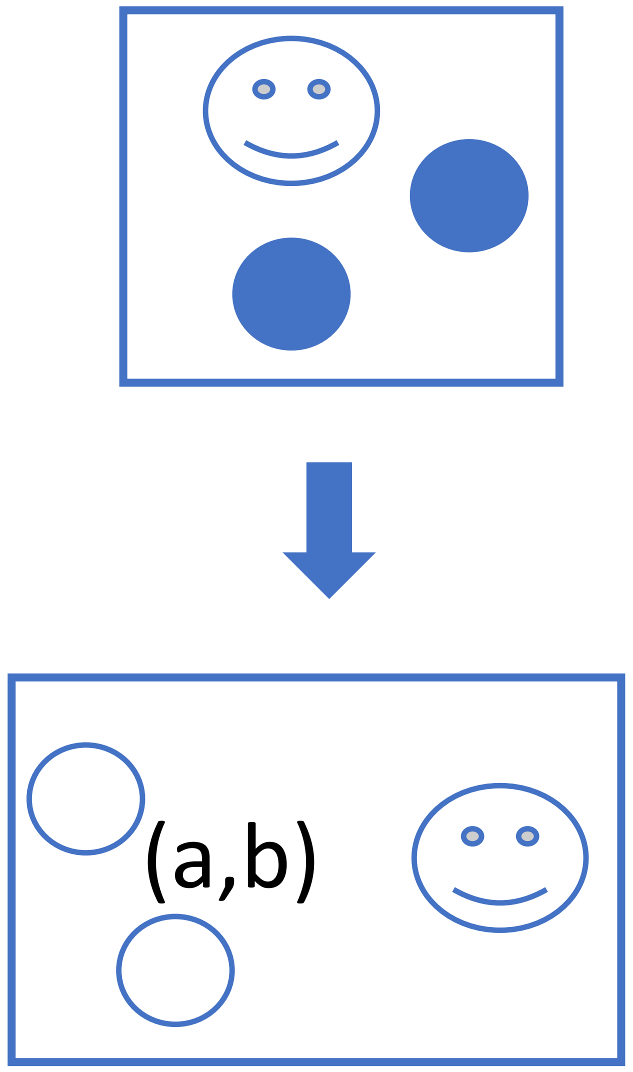
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

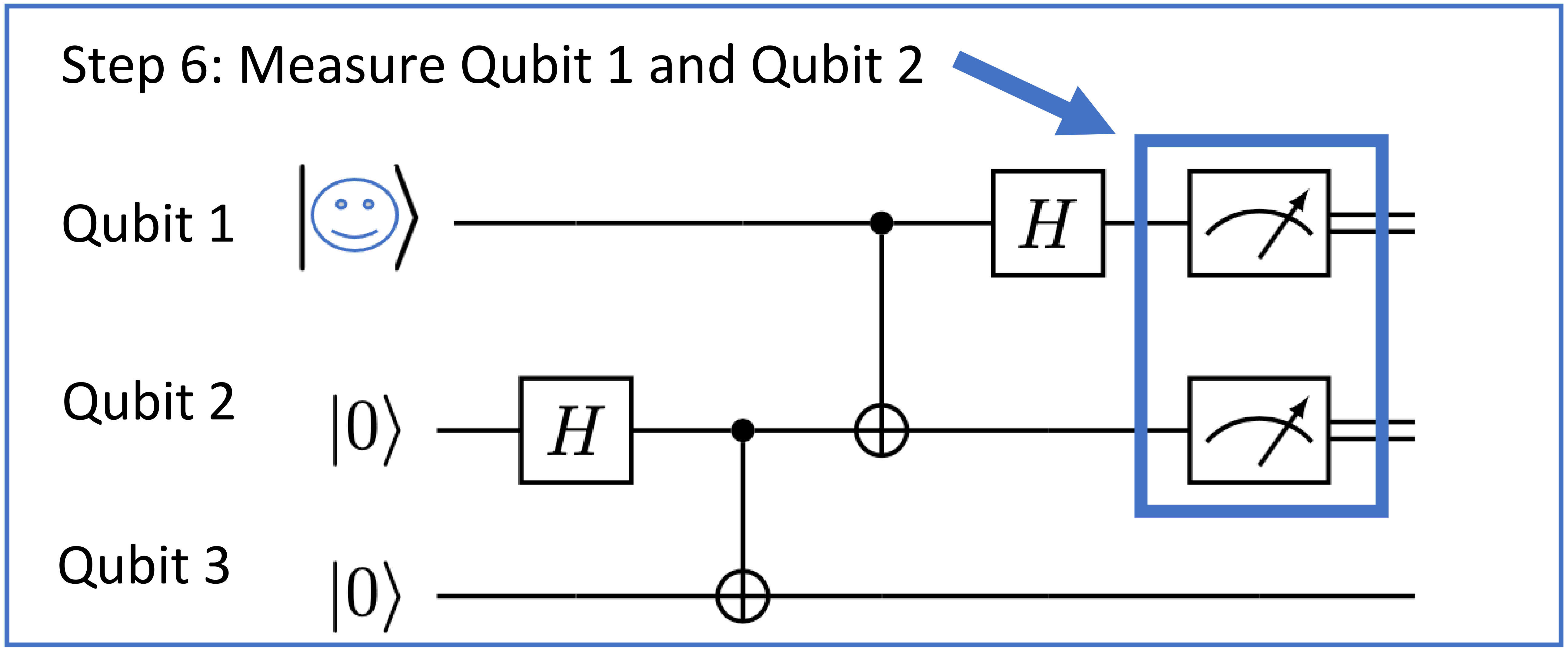
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit





Step 6: Measure Qubit 1 and Qubit 2

Center for Quantum Networks *NSF-ERC*

Quantum State Teleportation - Refresher

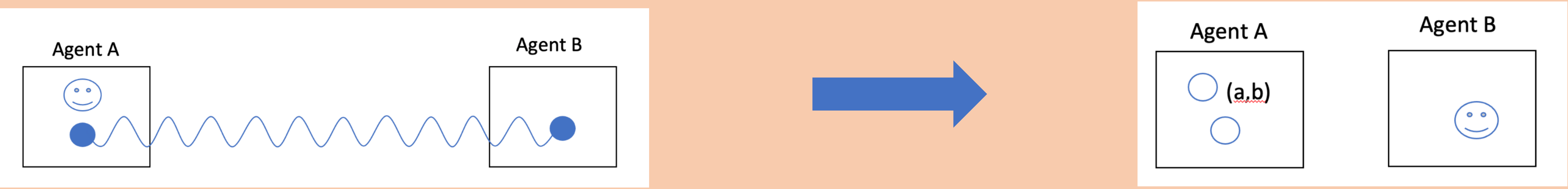
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

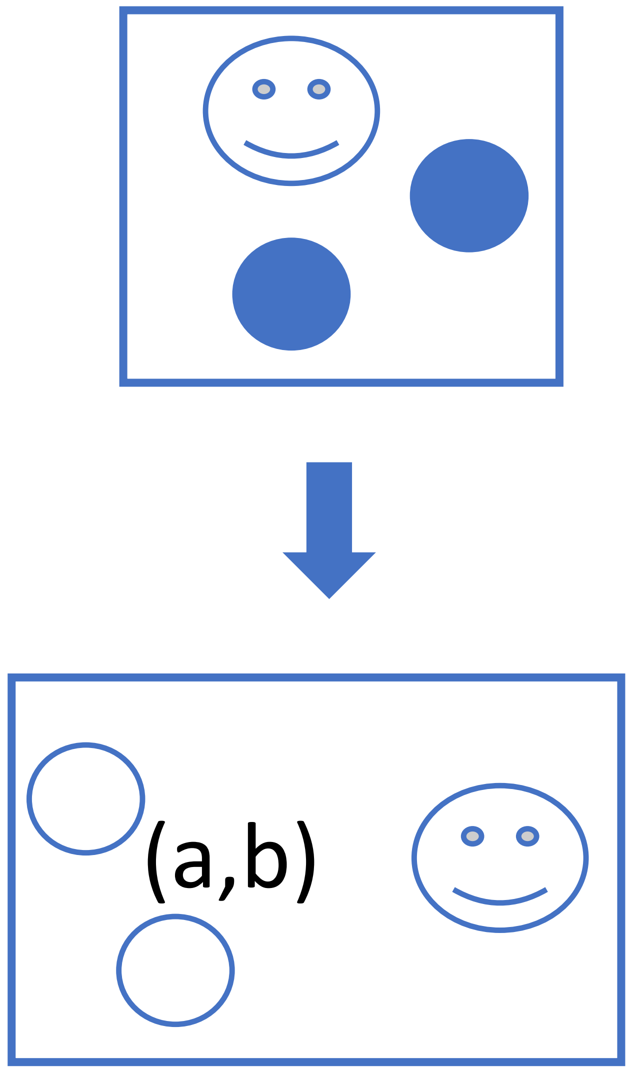
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

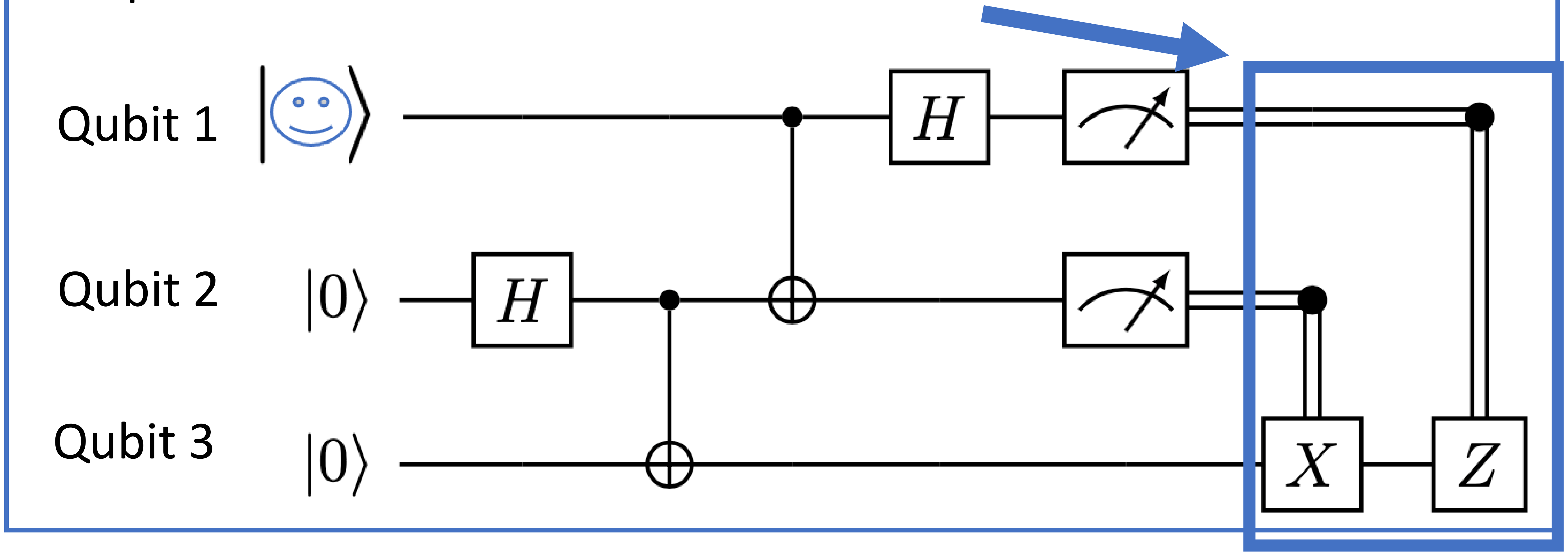
Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit



Step 7: Use measurement results to decide if need to correct state



Quantum State Teleportation - Refresher

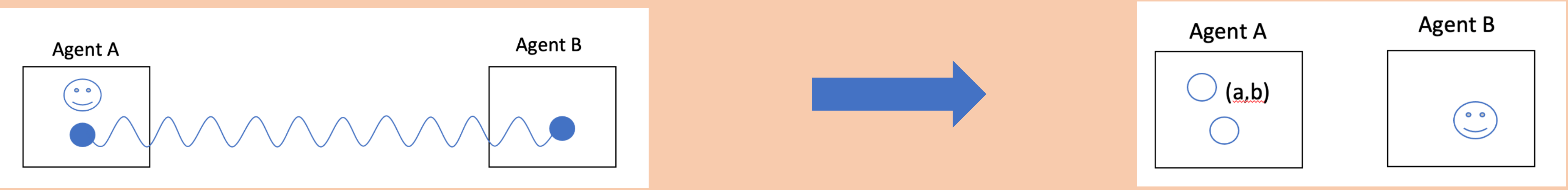
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

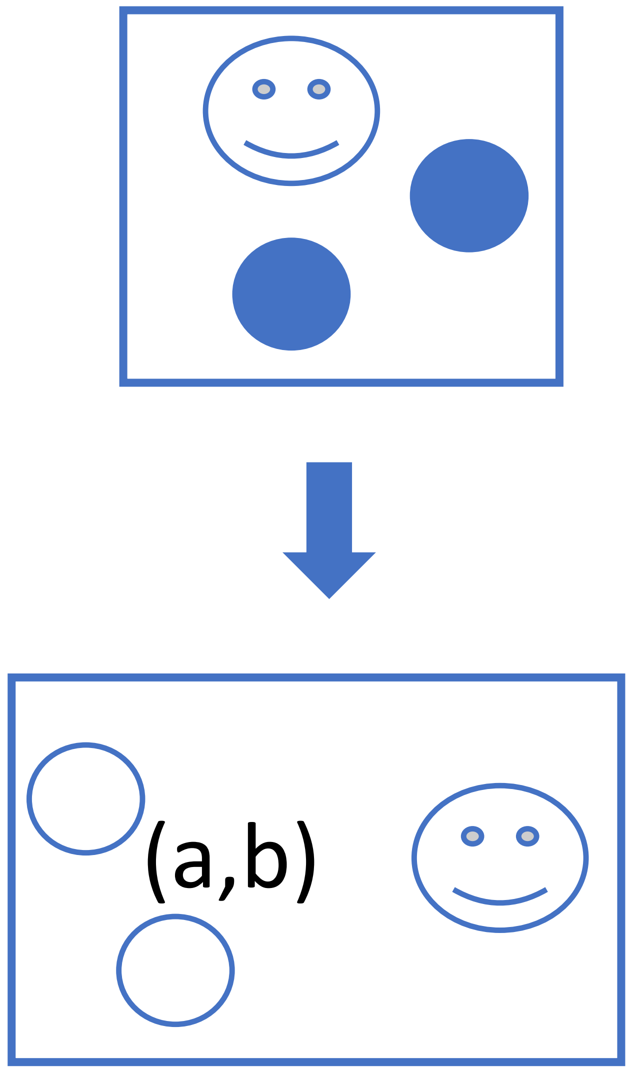
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

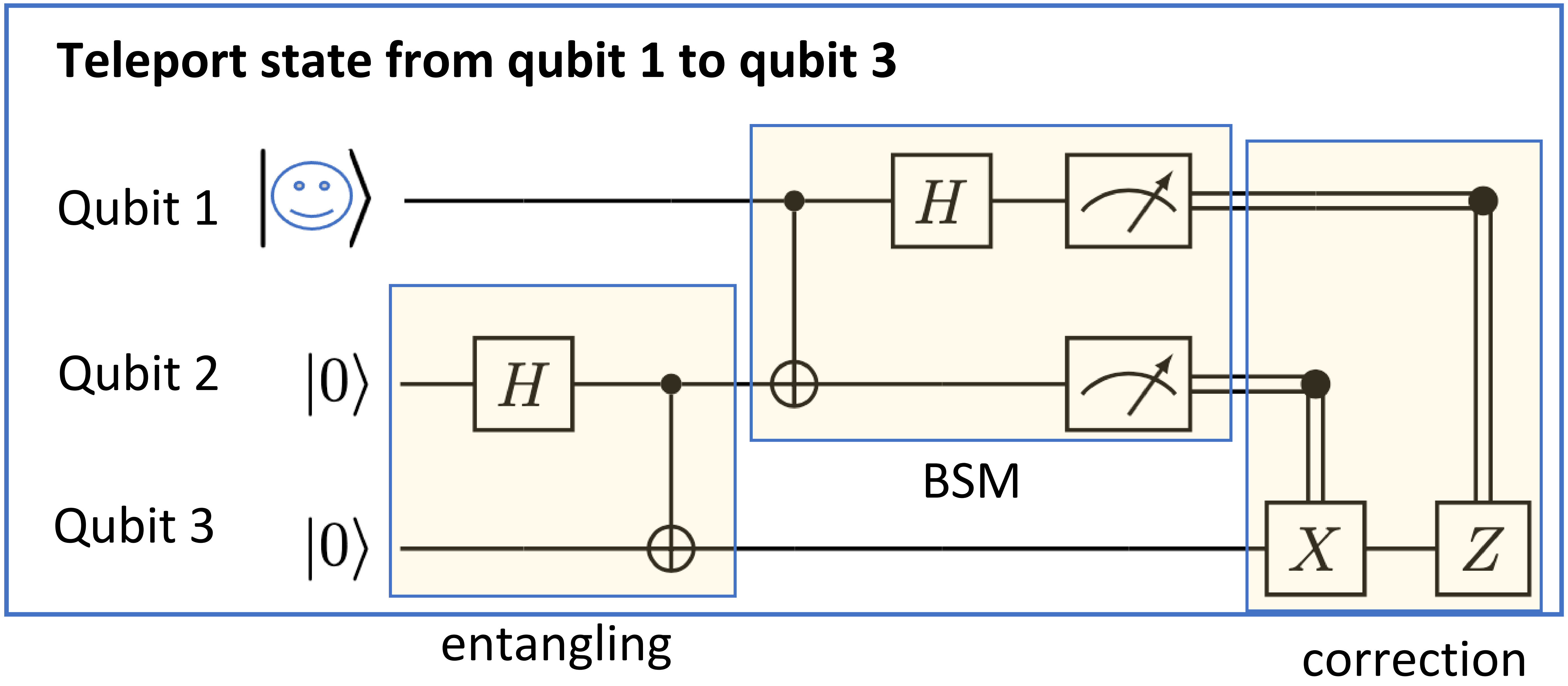
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

Step 3: Agent B uses the classical bits to correct state of local qubit.

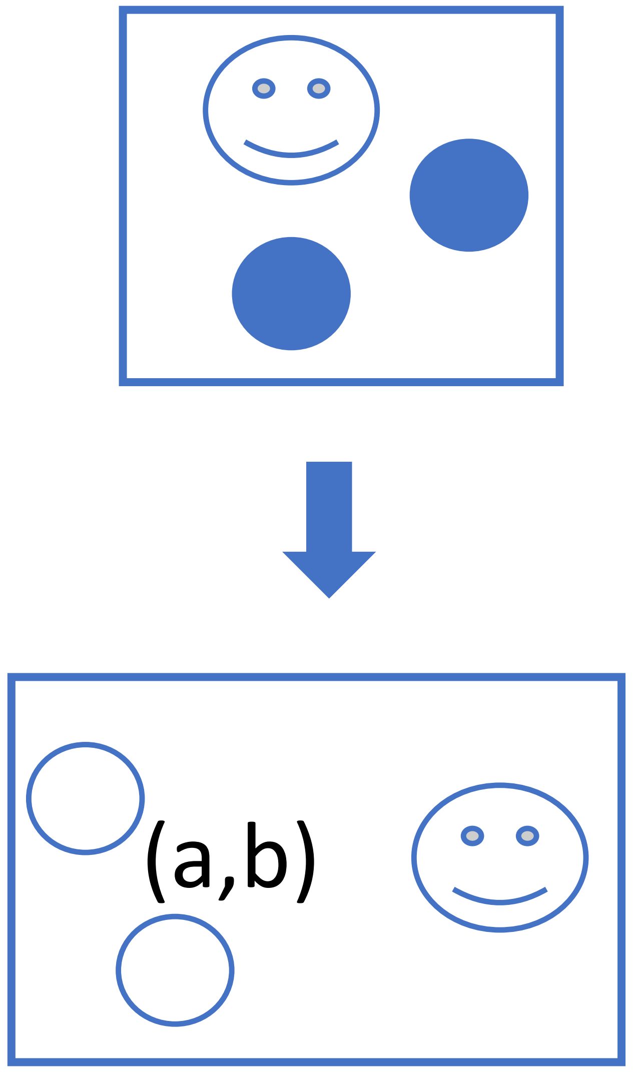


Or: Goal: Teleport state of one qubit to another qubit

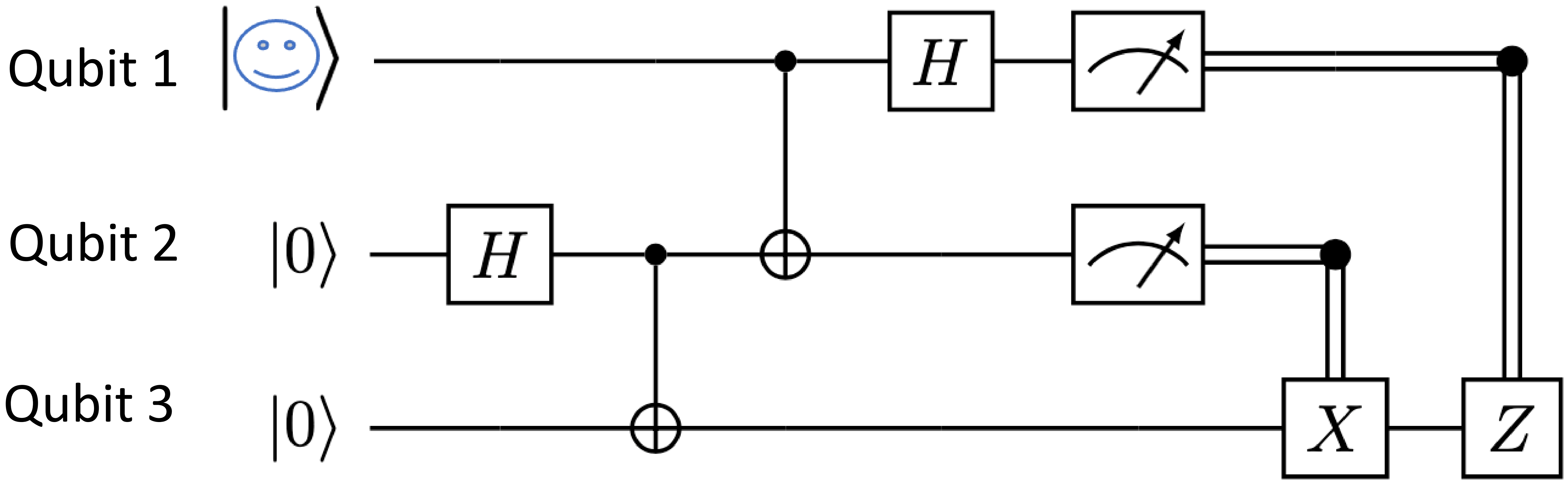




Teleport state from qubit 1 to qubit 3



Teleport state from qubit 1 to qubit 3



What do we need?

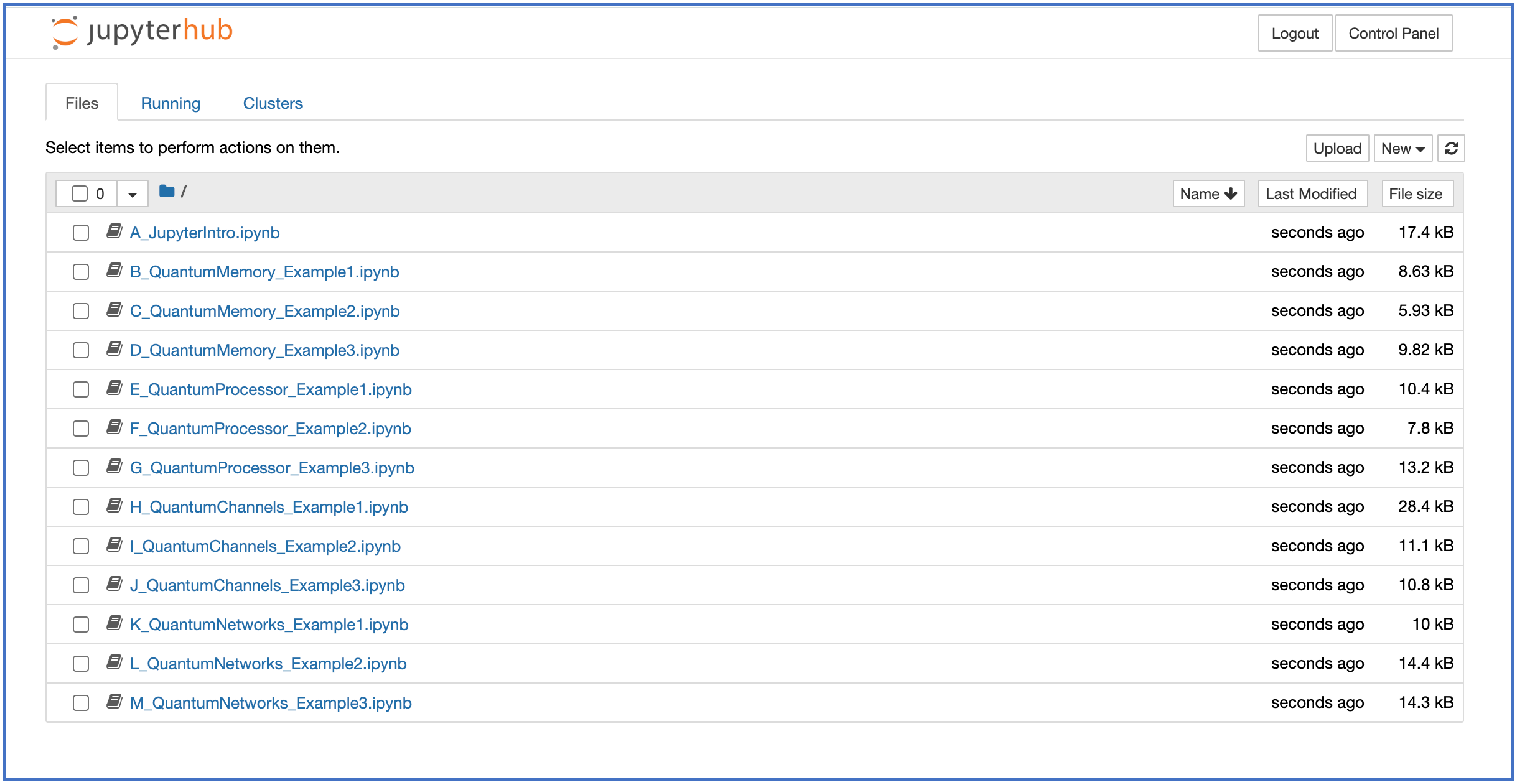
- Quantum memory (way to store qubits)
- Manipulate state of qubits (apply gates)
- Measure qubits

Center for Quantum Networks NSF-ERC

Simulation Deep Dive: Learning by Doing

Hands-on Explorations

Self-contained notebooks

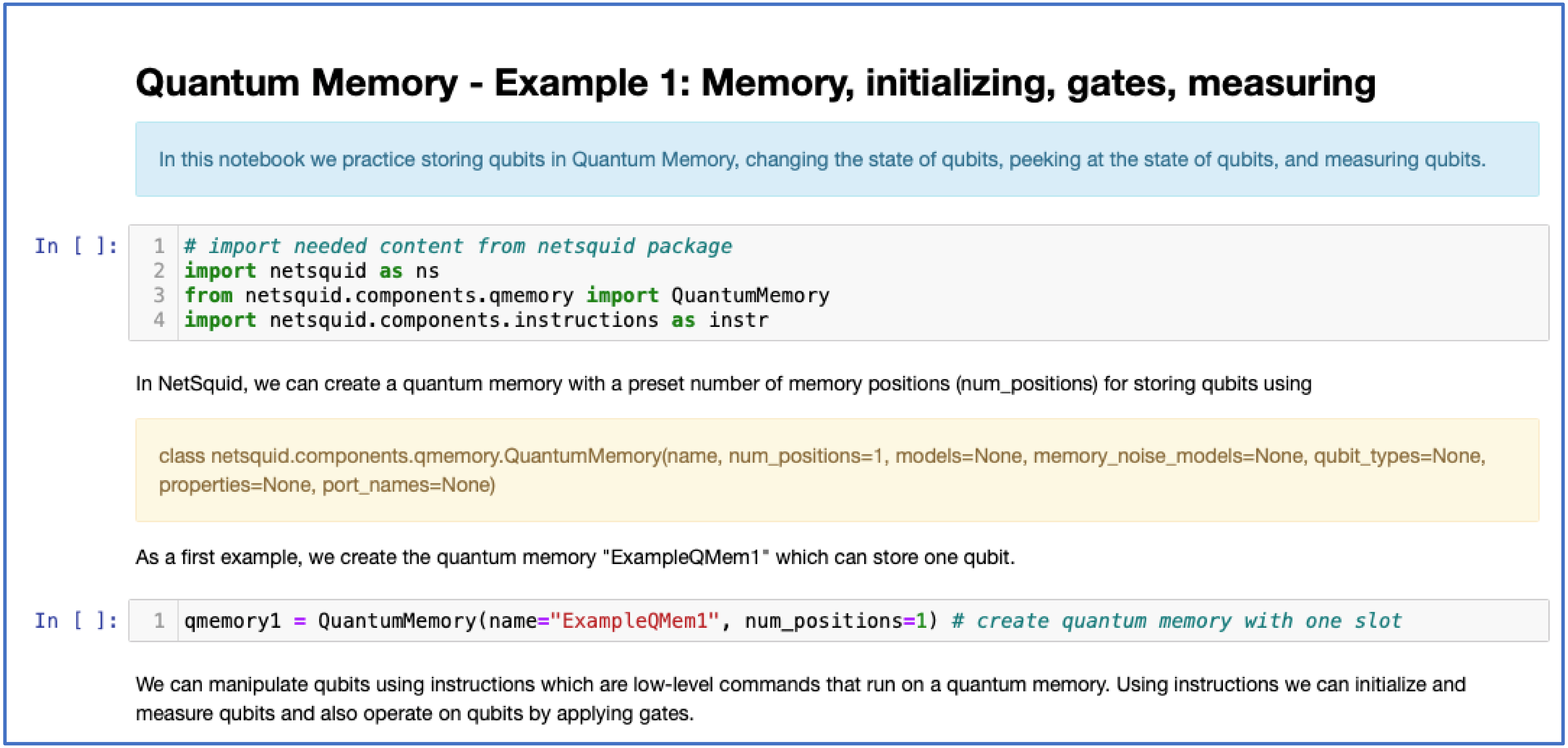


Select items to perform actions on them.

Name	Last Modified	File size
□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□ D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□ E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□ F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
□ G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
□ H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□ I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Center for Quantum Networks
NSF-ERC

What you will find in each notebook:



Quantum Memory - Example 1: Memory, initializing, gates, measuring

In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.

In [ ]:

```
# import needed content from netsquid package
import netsquid as ns
from netsquid.components.qmemory import QuantumMemory
import netsquid.components.instructions as instr
```

In NetSquid, we can create a quantum memory with a preset number of memory positions (num_positions) for storing qubits using

class netsquid.components.qmemory.QuantumMemory(name, num_positions=1, models=None, memory_noise_models=None, qubit_types=None, properties=None, port_names=None)

As a first example, we create the quantum memory "ExampleQMem1" which can store one qubit.

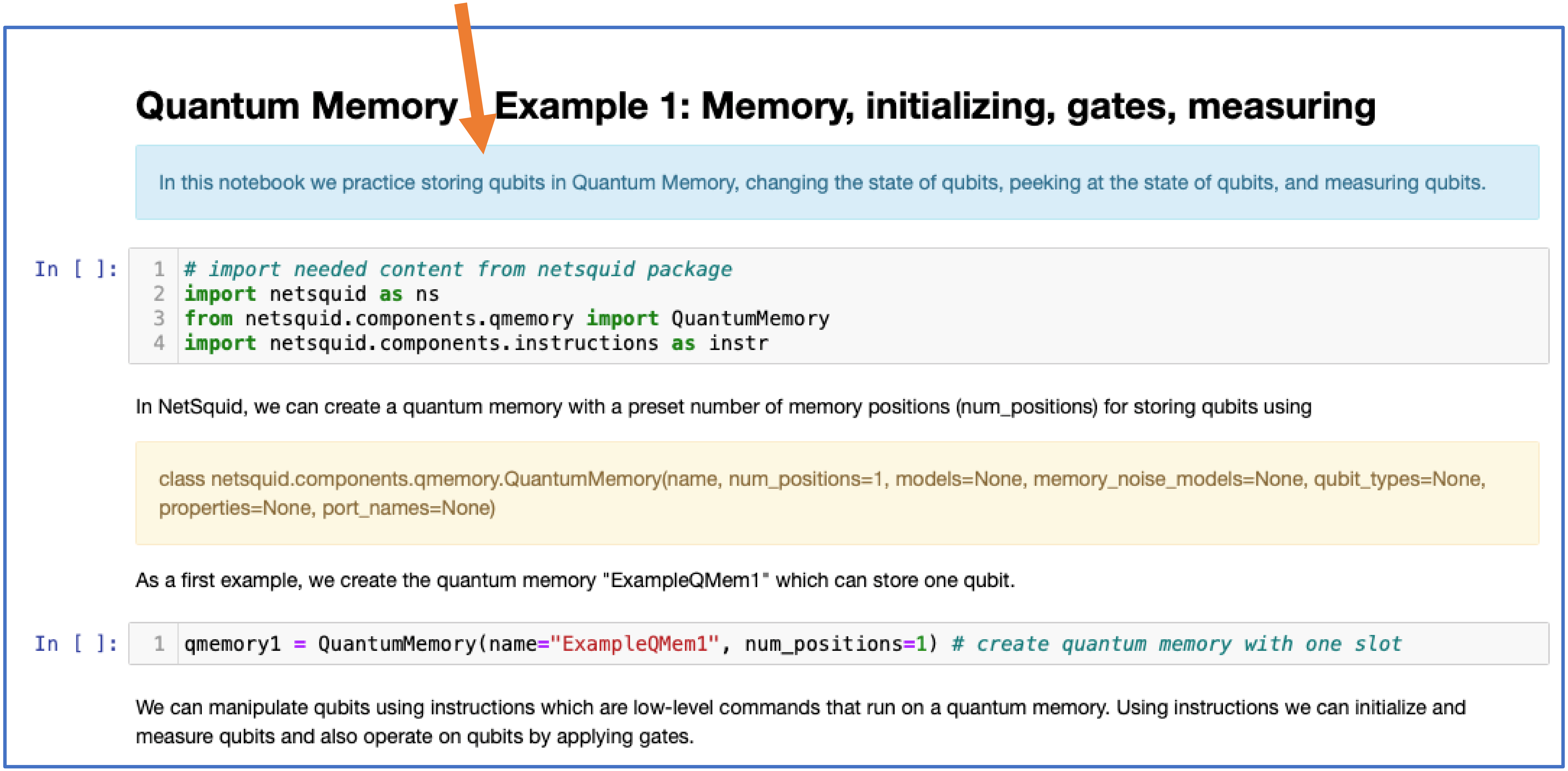
In [ ]:

```
qmemory1 = QuantumMemory(name="ExampleQMem1", num_positions=1) # create quantum memory with one slot
```

We can manipulate qubits using instructions which are low-level commands that run on a quantum memory. Using instructions we can initialize and measure qubits and also operate on qubits by applying gates.

What you will find in each notebook:

Blue box: quick summary of notebook



Quantum Memory Example 1: Memory, initializing, gates, measuring

In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.

In [ ]:
```
# import needed content from netsquid package
import netsquid as ns
from netsquid.components.qmemory import QuantumMemory
import netsquid.components.instructions as instr
```

In NetSquid, we can create a quantum memory with a preset number of memory positions (num_positions) for storing qubits using

class netsquid.components.qmemory.QuantumMemory(name, num_positions=1, models=None, memory_noise_models=None, qubit_types=None, properties=None, port_names=None)

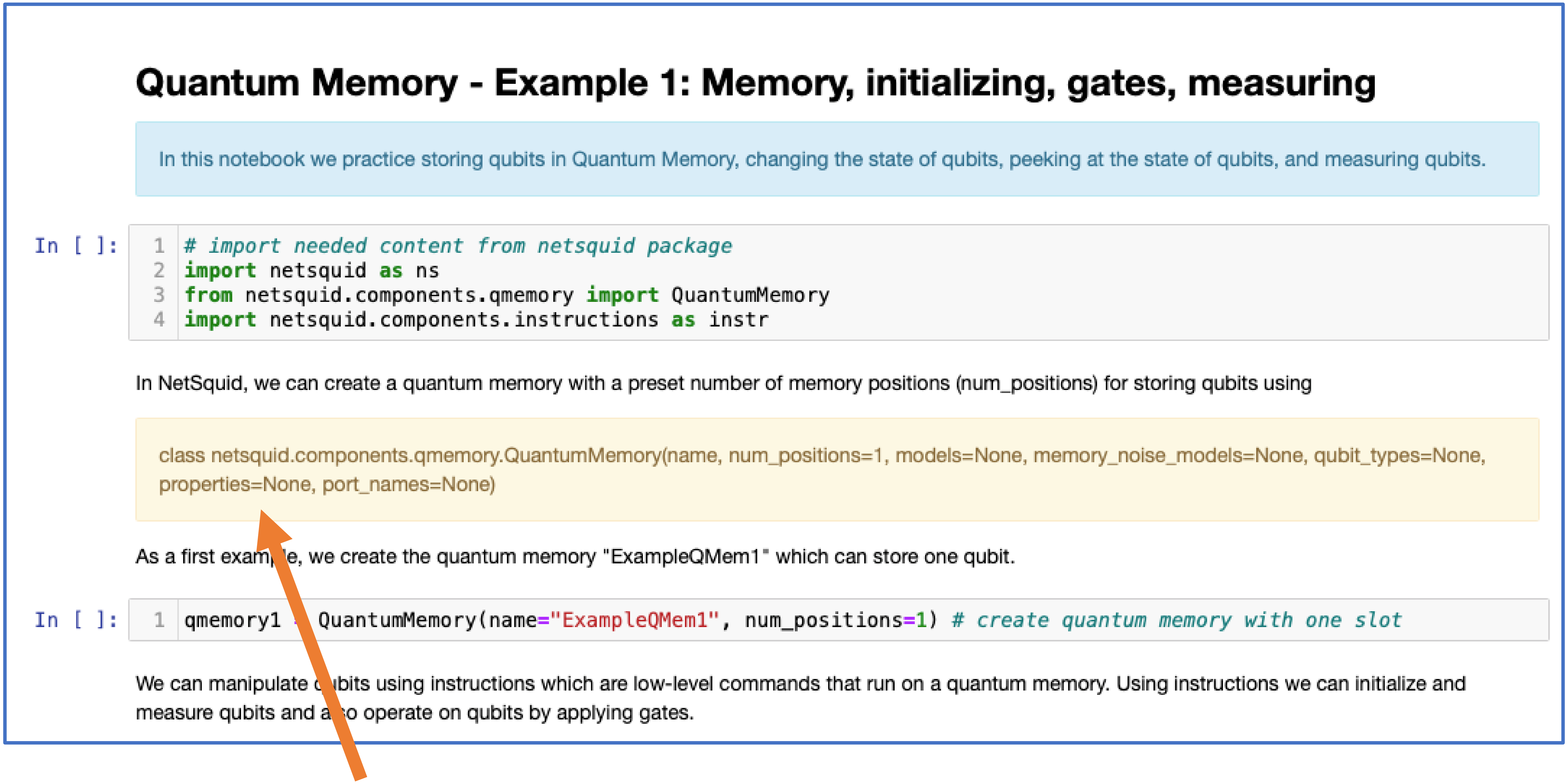
As a first example, we create the quantum memory "ExampleQMem1" which can store one qubit.

In [ ]:
```
qmemory1 = QuantumMemory(name="ExampleQMem1", num_positions=1) # create quantum memory with one slot
```

We can manipulate qubits using instructions which are low-level commands that run on a quantum memory. Using instructions we can initialize and measure qubits and also operate on qubits by applying gates.

Center for
Quantum
Networks
NSF-ERC

What you will find in each notebook:



Quantum Memory - Example 1: Memory, initializing, gates, measuring

In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.

In [ ]:

```
# import needed content from netsquid package
import netsquid as ns
from netsquid.components.qmemory import QuantumMemory
import netsquid.components.instructions as instr
```

In NetSquid, we can create a quantum memory with a preset number of memory positions (num_positions) for storing qubits using

class netsquid.components.qmemory.QuantumMemory(name, num_positions=1, models=None, memory_noise_models=None, qubit_types=None, properties=None, port_names=None)

As a first example, we create the quantum memory "ExampleQMem1" which can store one qubit.

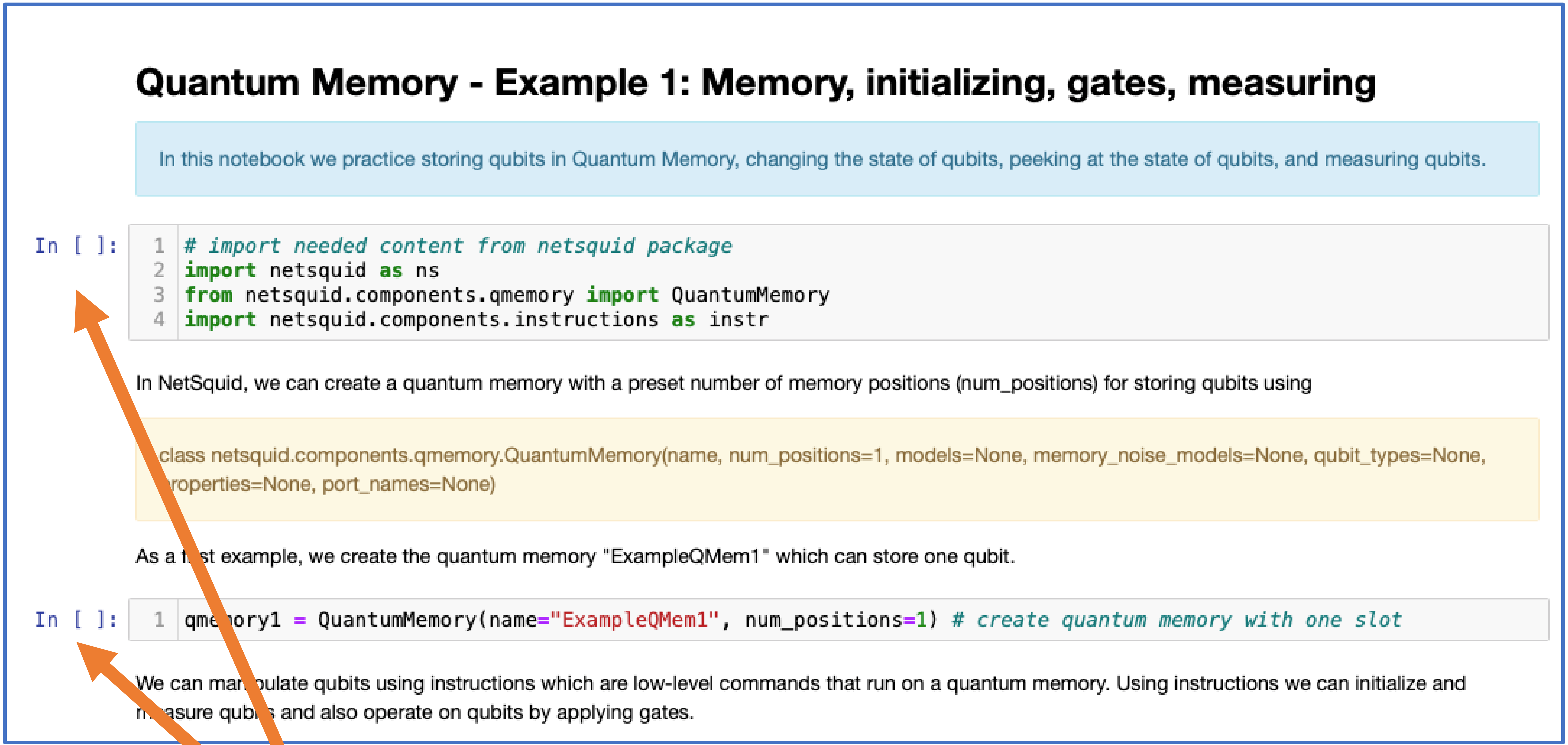
In [ ]:

```
qmemory1 = QuantumMemory(name="ExampleQMem1", num_positions=1) # create quantum memory with one slot
```

We can manipulate qubits using instructions which are low-level commands that run on a quantum memory. Using instructions we can initialize and measure qubits and also operate on qubits by applying gates.

Yellow boxes: new definitions with default settings (for future reference)

What you will find in each notebook:



Quantum Memory - Example 1: Memory, initializing, gates, measuring

In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.

In [ ]:

```
# import needed content from netsquid package
import netsquid as ns
from netsquid.components.qmemory import QuantumMemory
import netsquid.components.instructions as instr
```

In NetSquid, we can create a quantum memory with a preset number of memory positions (num_positions) for storing qubits using

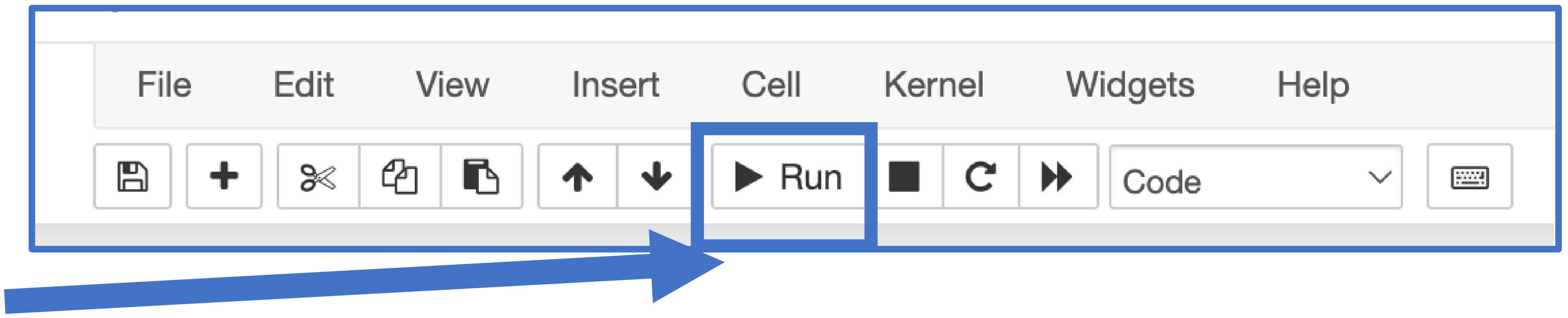
class netsquid.components.qmemory.QuantumMemory(name, num_positions=1, models=None, memory_noise_models=None, qubit_types=None, properties=None, port_names=None)

As a first example, we create the quantum memory "ExampleQMem1" which can store one qubit.

In [ ]:

```
qmemory1 = QuantumMemory(name="ExampleQMem1", num_positions=1) # create quantum memory with one slot
```

We can manipulate qubits using instructions which are low-level commands that run on a quantum memory. Using instructions we can initialize and measure qubits and also operate on qubits by applying gates.

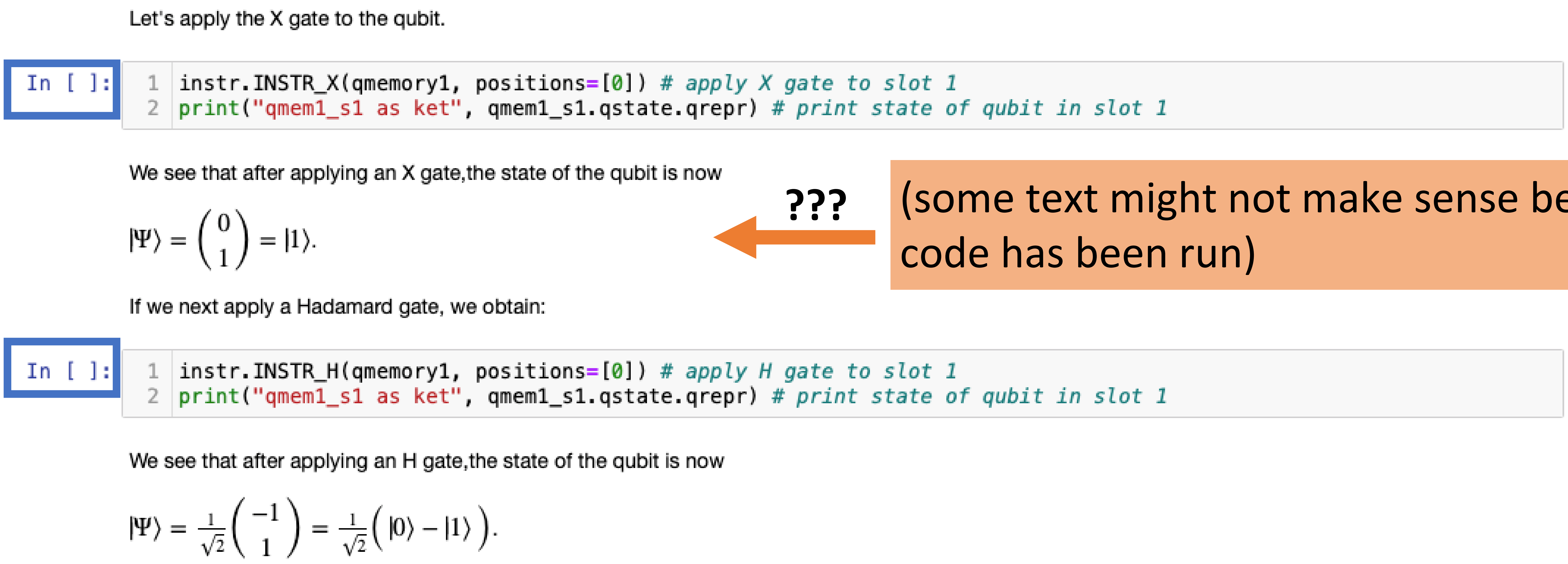


Code cells: code to run

(Click in cell to select)

What you will find in each notebook:

Before code has been run:



Let's apply the X gate to the qubit.

In [ ]:
```
instr.INSTR_X(qmemory1, positions=[0]) # apply X gate to slot 1
print("qmem1_s1 as ket", qmem1_s1.qstate.qrepr) # print state of qubit in slot 1
```

We see that after applying an X gate,the state of the qubit is now

$$|\Psi\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle.$$

If we next apply a Hadamard gate, we obtain:

In [ ]:
```
instr.INSTR_H(qmemory1, positions=[0]) # apply H gate to slot 1
print("qmem1_s1 as ket", qmem1_s1.qstate.qrepr) # print state of qubit in slot 1
```

We see that after applying an H gate,the state of the qubit is now

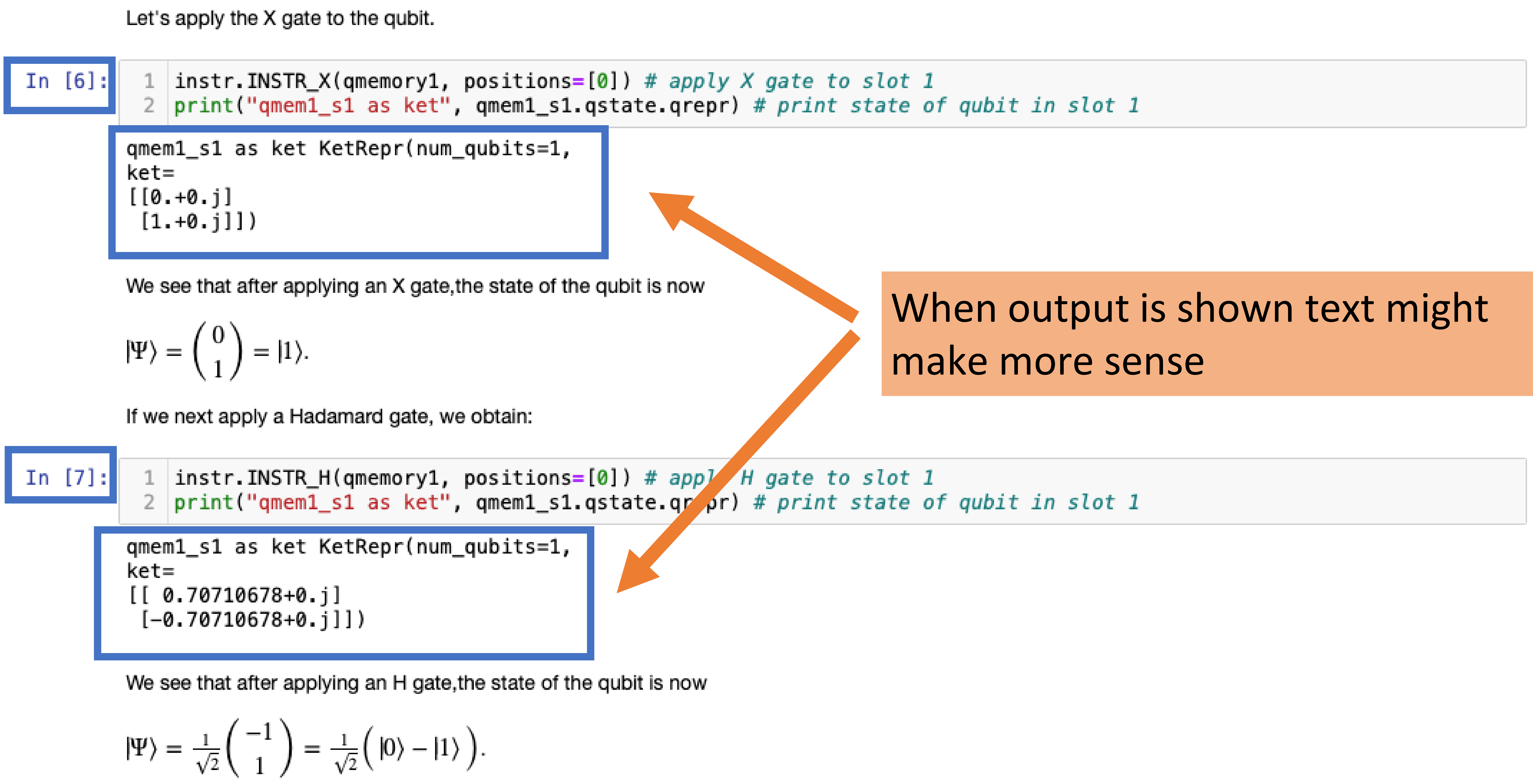
$$|\Psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \Big(|0\rangle - |1\rangle \Big).$$

???

(some text might not make sense before code has been run)

What you will find in each notebook:

After code has been run:



Let's apply the X gate to the qubit.

In [6]:
```
instr.INSTR_X(qmemory1, positions=[0]) # apply X gate to slot 1
print("qmem1_s1 as ket", qmem1_s1.qstate.qrepr) # print state of qubit in slot 1
```

```
qmem1_s1 as ket KetRepr(num_qubits=1,
ket=
[[0.+0.j]
 [1.+0.j]])
```

We see that after applying an X gate,the state of the qubit is now

$$|\Psi\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle.$$

If we next apply a Hadamard gate, we obtain:

In [7]:
```
instr.INSTR_H(qmemory1, positions=[0]) # apply H gate to slot 1
print("qmem1_s1 as ket", qmem1_s1.qstate.qrepr) # print state of qubit in slot 1
```

```
qmem1_s1 as ket KetRepr(num_qubits=1,
ket=
[[ 0.70710678+0.j]
 [-0.70710678+0.j]])
```

We see that after applying an H gate,the state of the qubit is now

$$|\Psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\Big(|0\rangle - |1\rangle\Big).$$

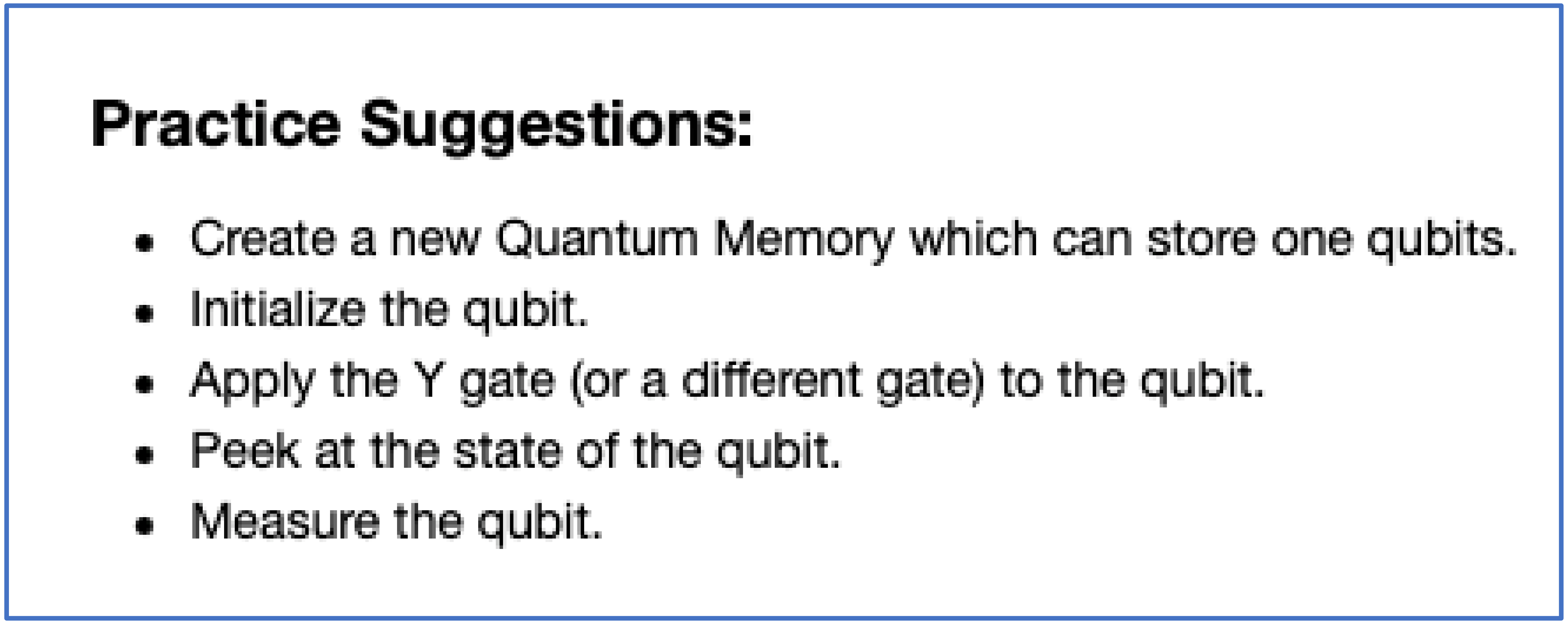
When output is shown text might make more sense

(if you see In [*]: then code is not done running yet)

Center for Quantum Networks
NSF-ERC

What you will find in each notebook:

Practice suggestions:

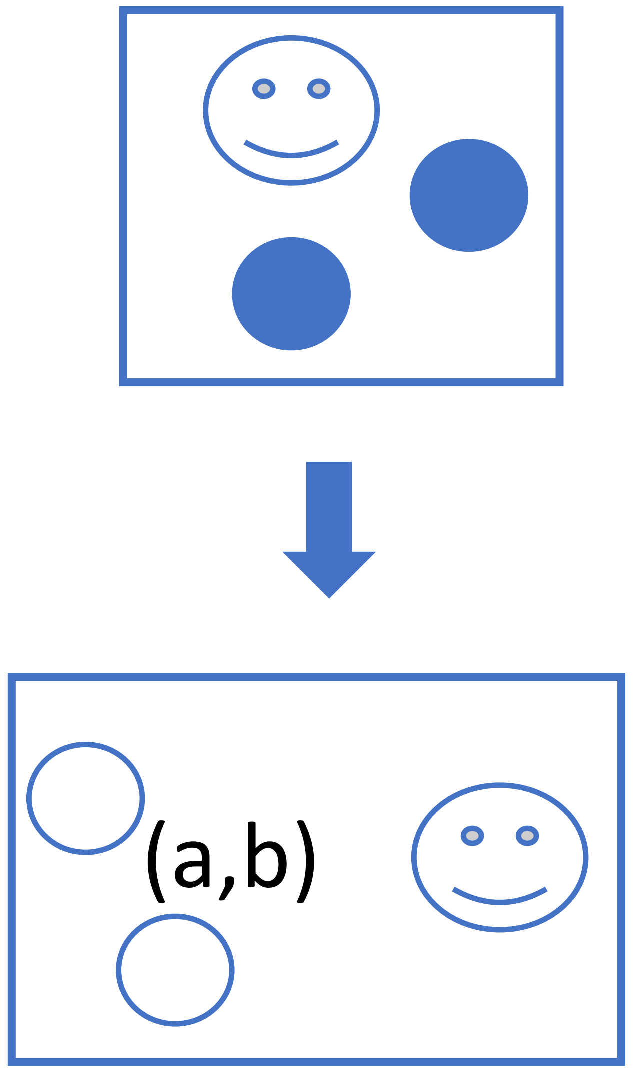


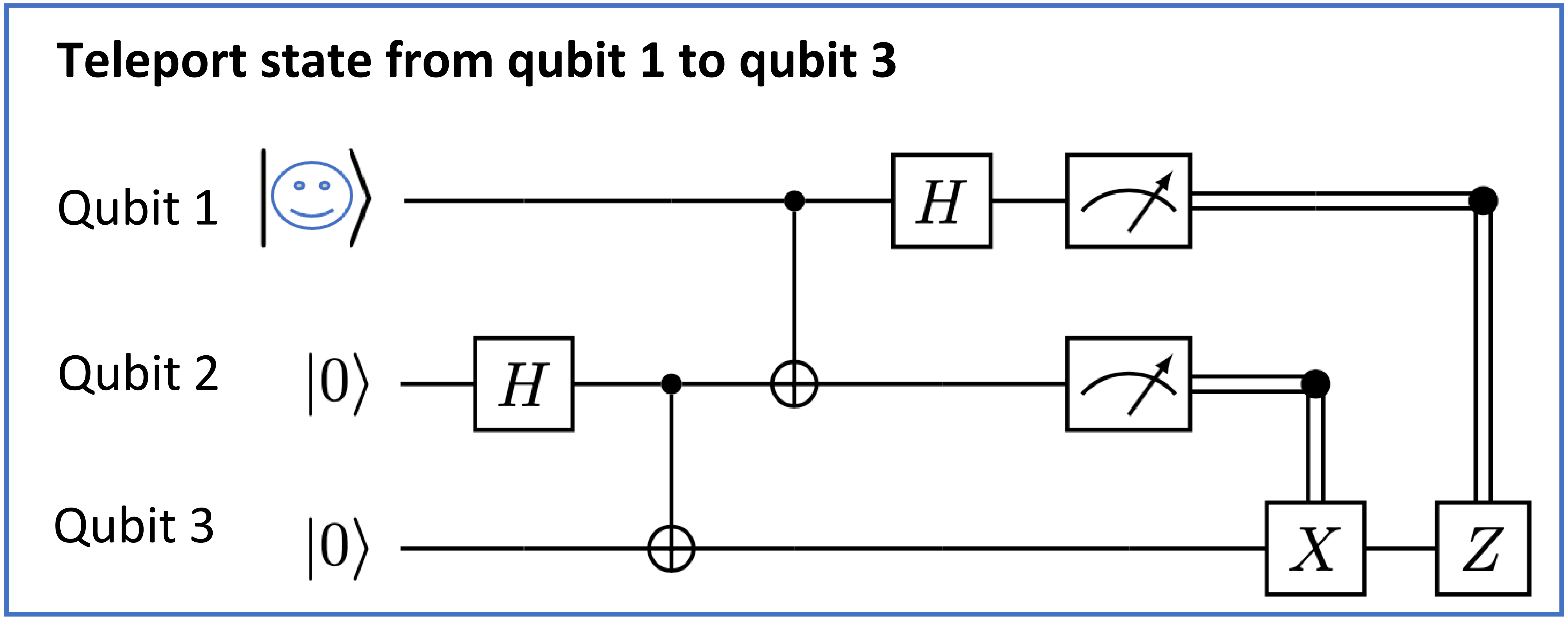
Practice Suggestions:

- Create a new Quantum Memory which can store one qubits.
- Initialize the qubit.
- Apply the Y gate (or a different gate) to the qubit.
- Peek at the state of the qubit.
- Measure the qubit.

How to get the most out of this short course:

- Be engaged!
- Work with the provided notebooks
- Try to apply the material
- Ideally: work with others, discuss your questions etc.



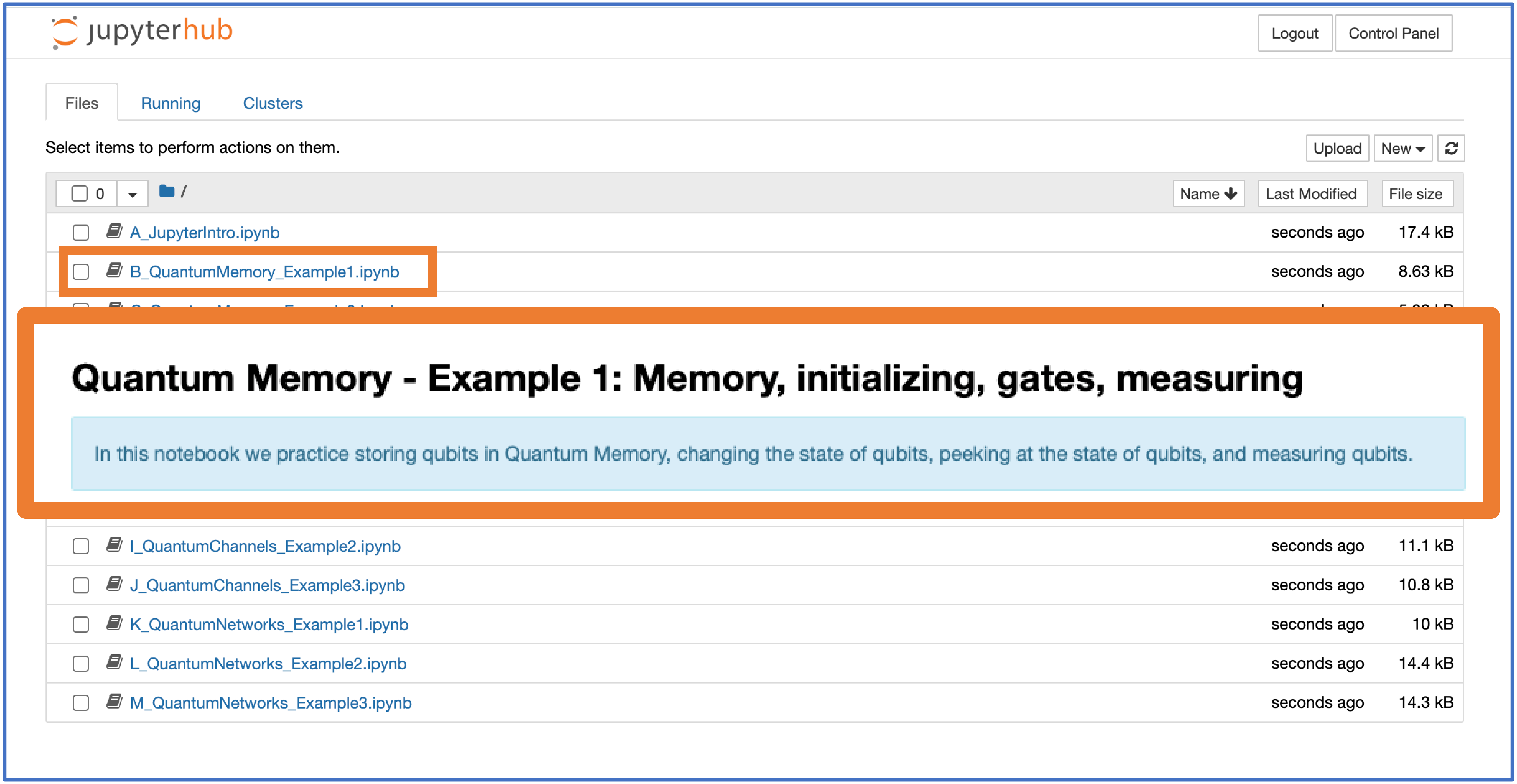


What do we need?

- Quantum memory (way to store qubits)
- Manipulate state of qubits (apply gates)
- Measure qubits

Let’s simulate it!

Notebook: B_QuantumMemory_Example1.ipyn



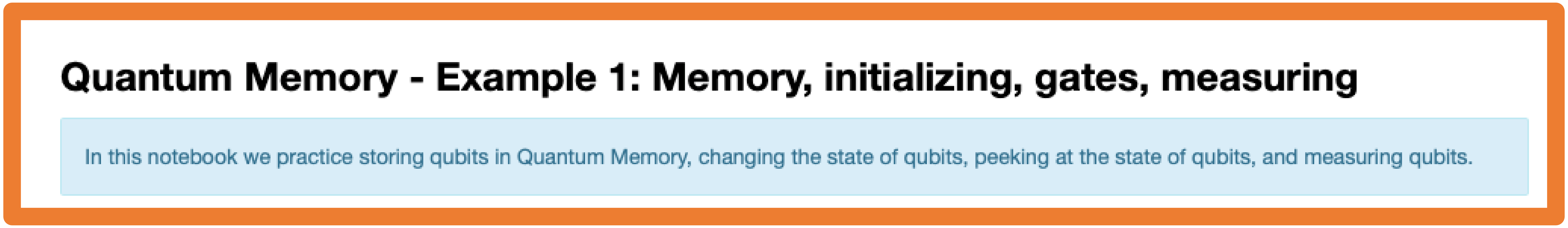
Select items to perform actions on them.

□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Memory - Example 1: Memory, initializing, gates, measuring

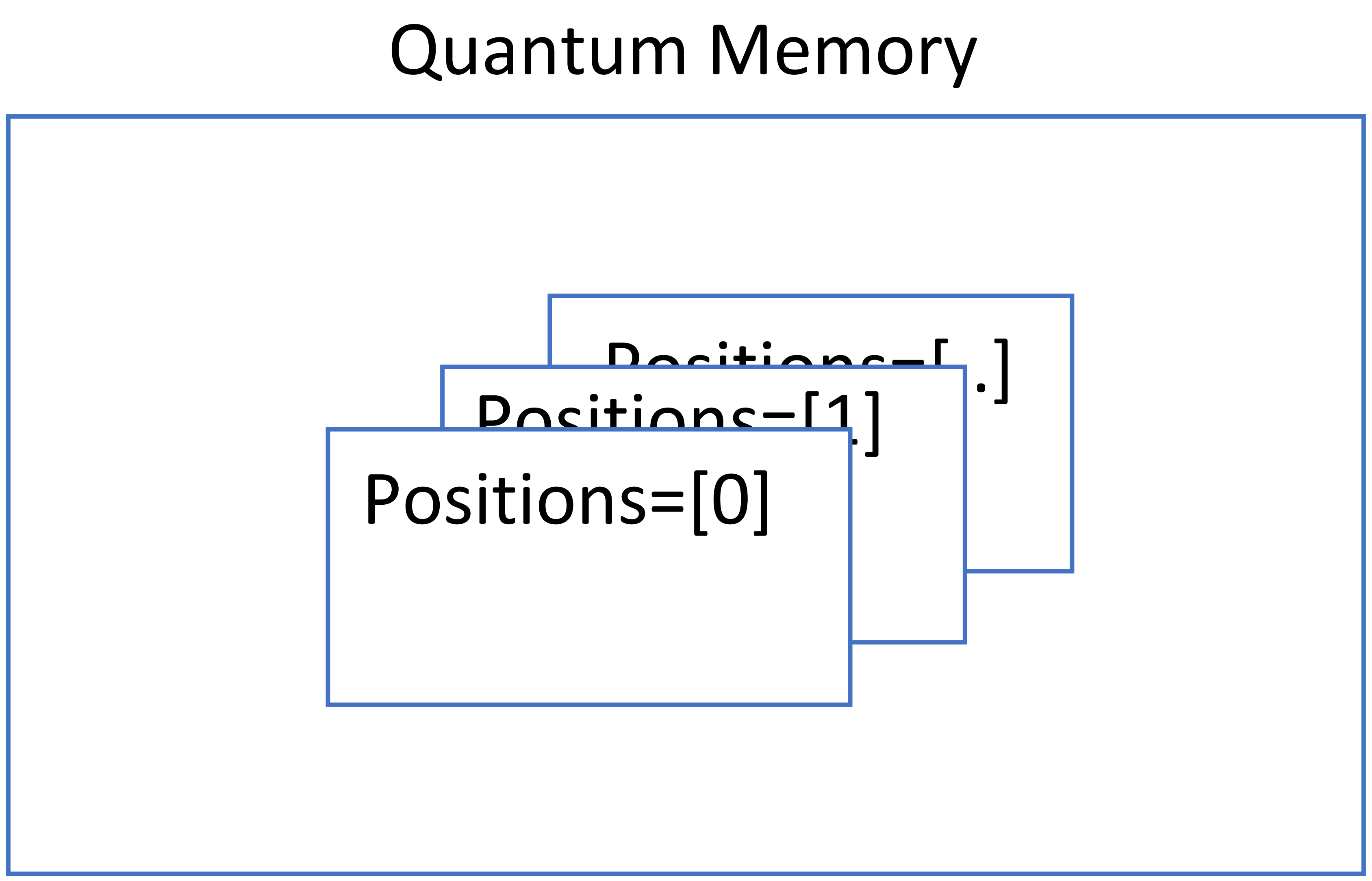
In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.

Notebook: B_QuantumMemory_Example1.ipyn



Quantum Memory - Example 1: Memory, initializing, gates, measuring

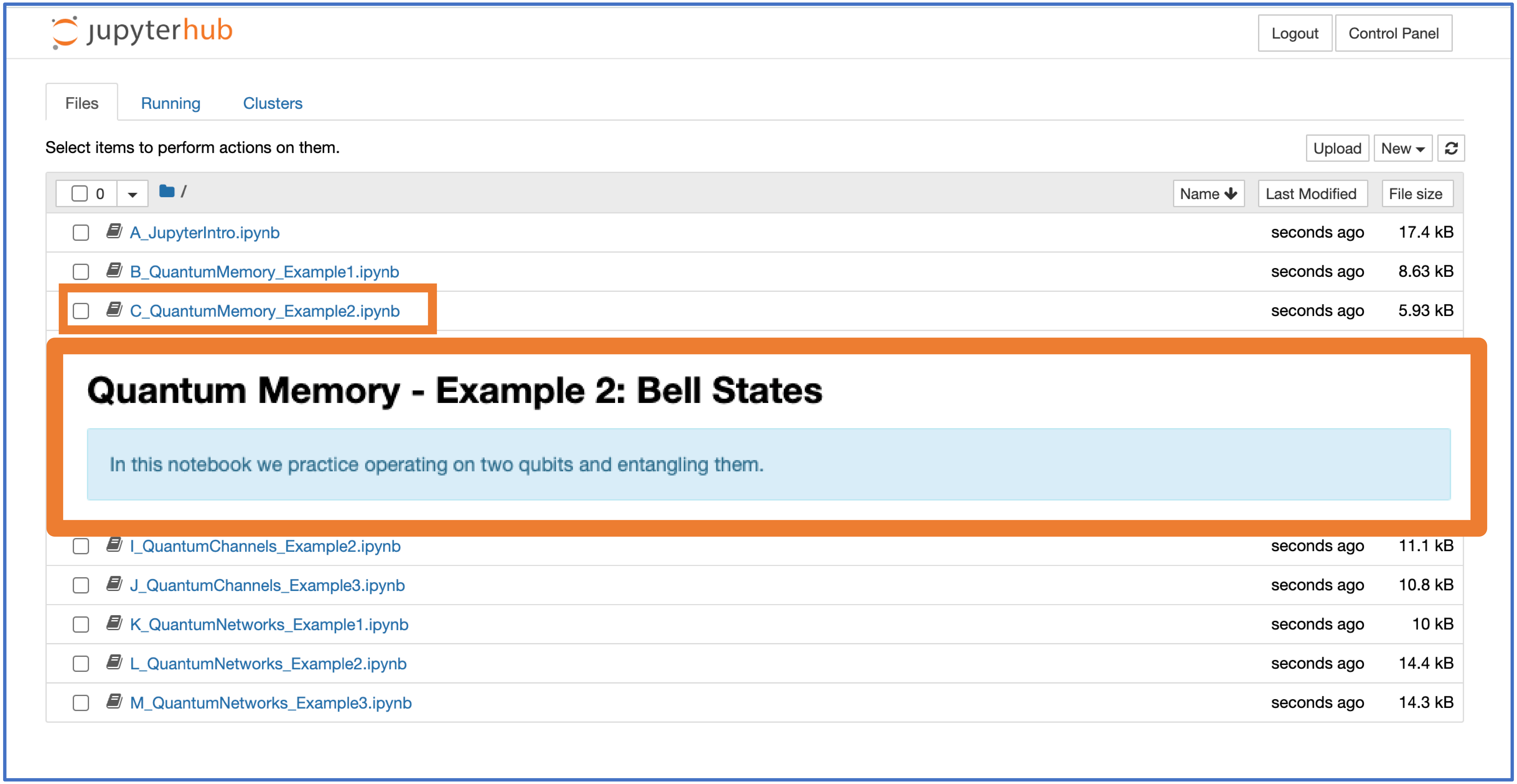
In this notebook we practice storing qubits in Quantum Memory, changing the state of qubits, peeking at the state of qubits, and measuring qubits.



- Place a qubit in a memory
- Apply gates to change state
- Peek at a qubit (not physically possible but so convenient..)
- measure

Center for Quantum Networks
NSF-ERC

Notebook: C_QuantumMemory_Example2.ipyn



Select items to perform actions on them.

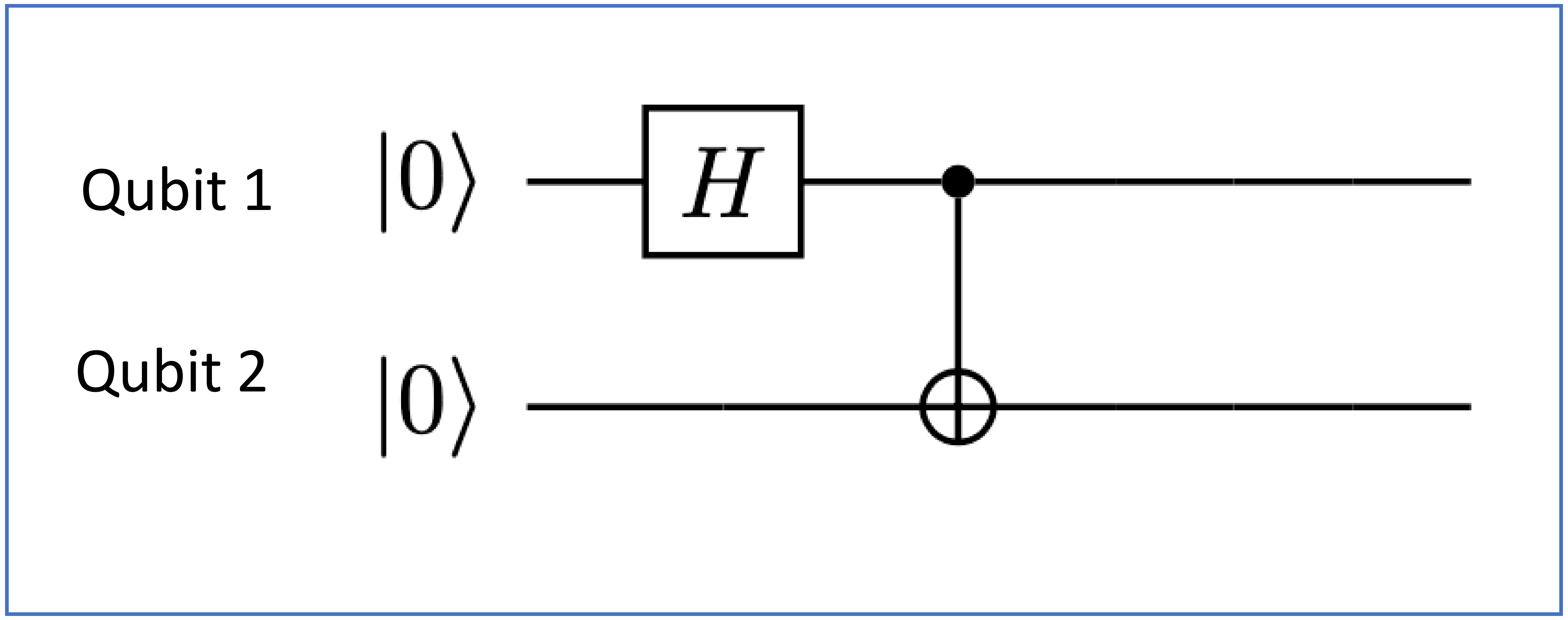
	Name	Last Modified	File size
□	A_JupyterIntro.ipynb	seconds ago	17.4 kB
□	B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□	C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□	I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□	J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□	K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□	L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□	M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Memory - Example 2: Bell States

In this notebook we practice operating on two qubits and entangling them.

Notebook: C_QuantumMemory_Example2.ipyn

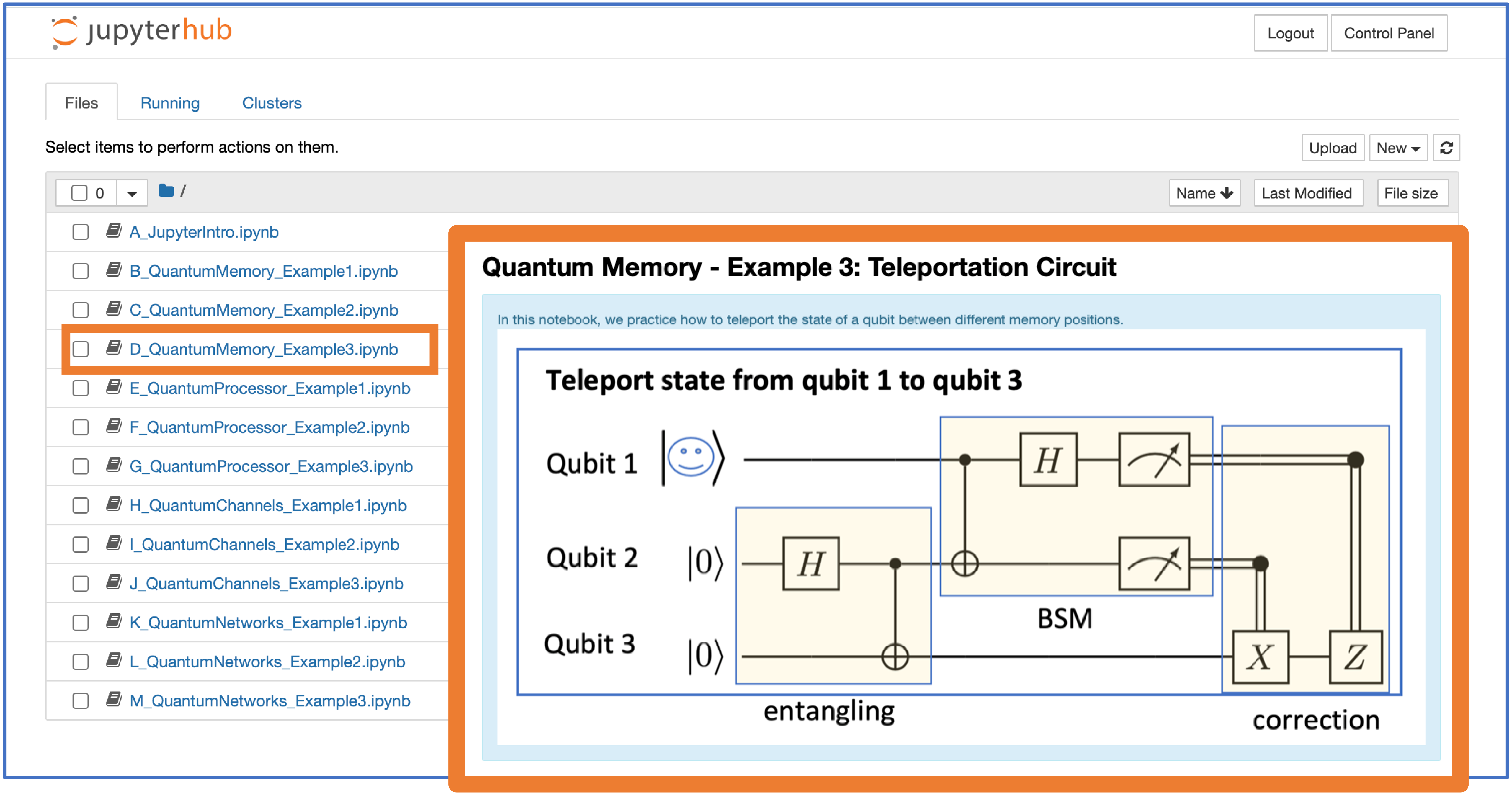
Quantum Memory



- Place two qubits in a quantum memory
- Entangle the two qubits
- Measure the two qubits

Center for Quantum Networks *NSF-ERC*

Notebook: D_QuantumMemory_Example3.ipyn



Select items to perform actions on them.

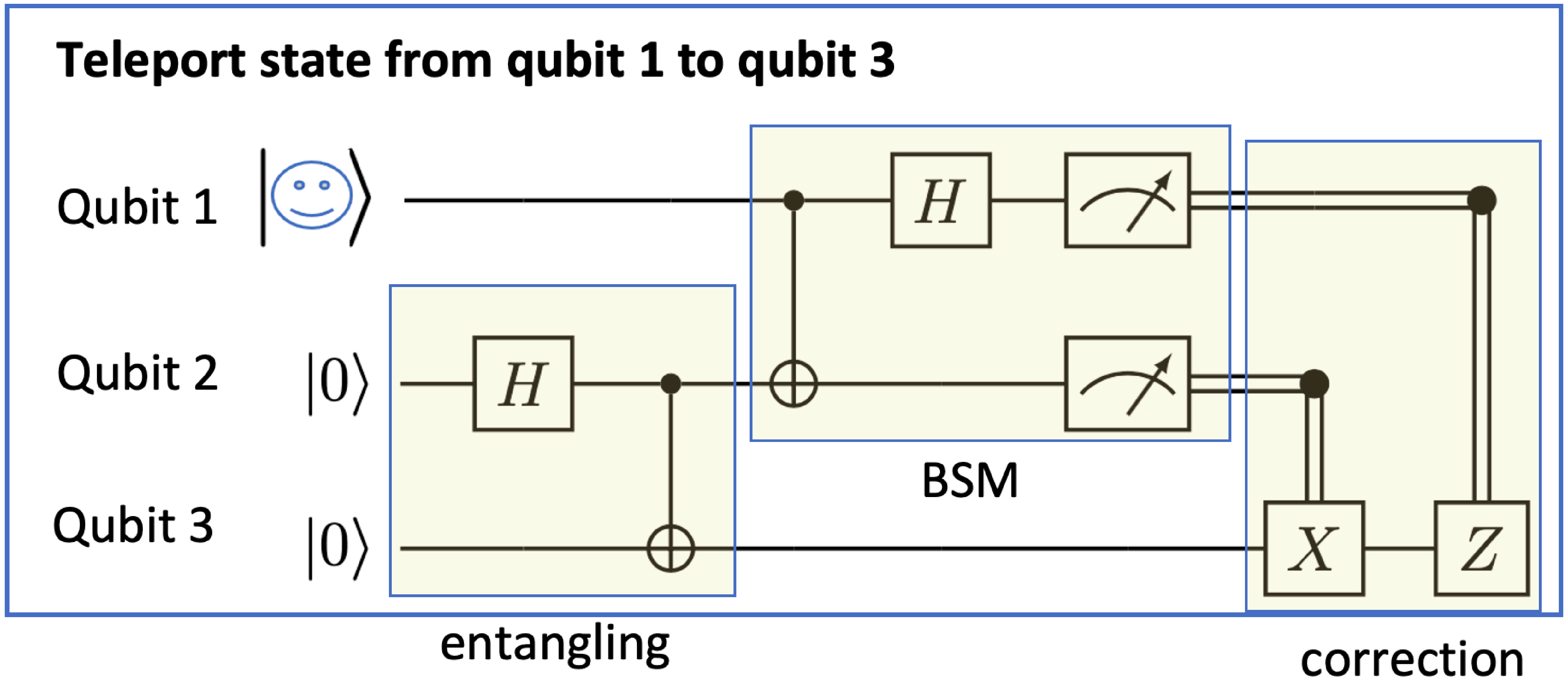
- □ A_JupyterIntro.ipynb
- □ B_QuantumMemory_Example1.ipynb
- □ C_QuantumMemory_Example2.ipynb
- □ D_QuantumMemory_Example3.ipynb
- □ E_QuantumProcessor_Example1.ipynb
- □ F_QuantumProcessor_Example2.ipynb
- □ G_QuantumProcessor_Example3.ipynb
- □ H_QuantumChannels_Example1.ipynb
- □ I_QuantumChannels_Example2.ipynb
- □ J_QuantumChannels_Example3.ipynb
- □ K_QuantumNetworks_Example1.ipynb
- □ L_QuantumNetworks_Example2.ipynb
- □ M_QuantumNetworks_Example3.ipynb

Quantum Memory - Example 3: Teleportation Circuit

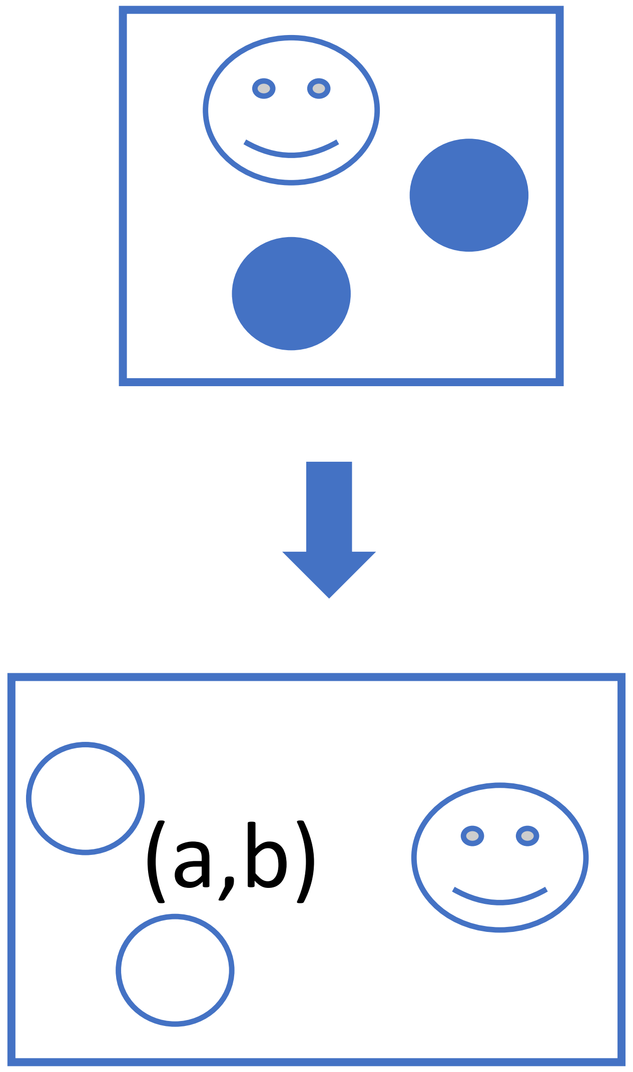
In this notebook, we practice how to teleport the state of a qubit between different memory positions.

Notebook: D_QuantumMemory_Example3.ipyn

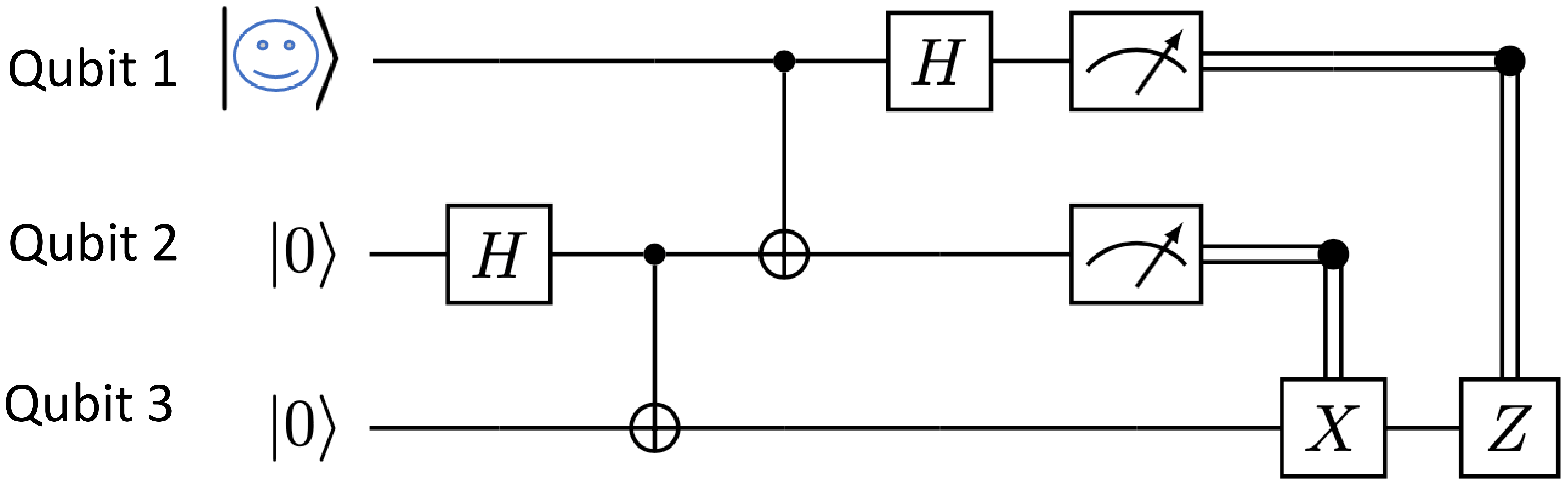
Quantum Memory



- Place three qubits in a quantum memory
- Entangle two qubits
- Perform a BSM on two qubits
- Perform corrections
- Calculate fidelity



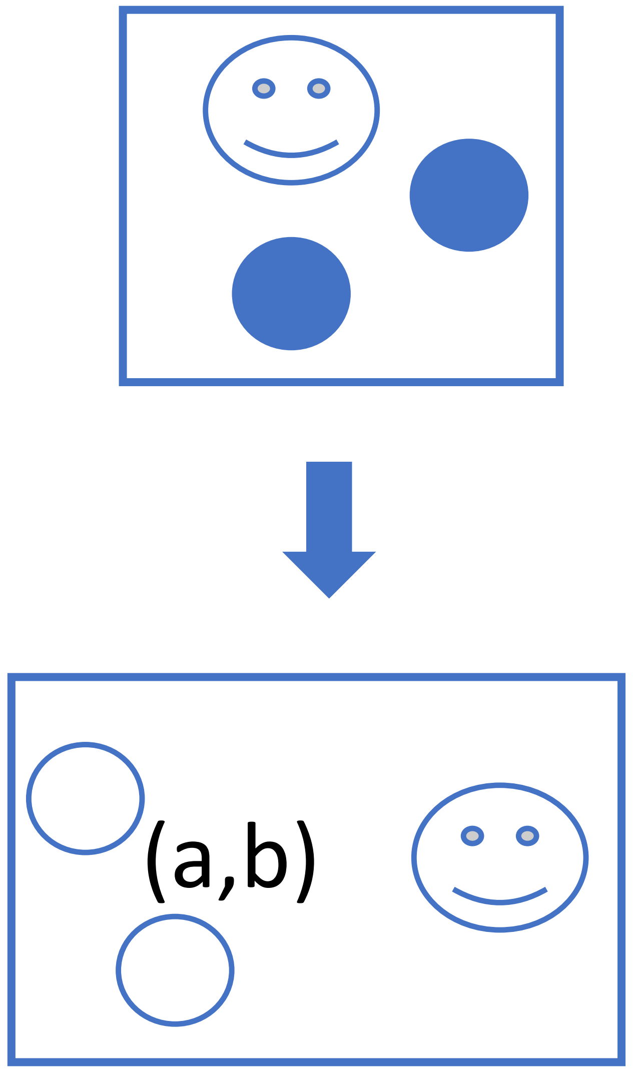
Teleport state from qubit 1 to qubit 3



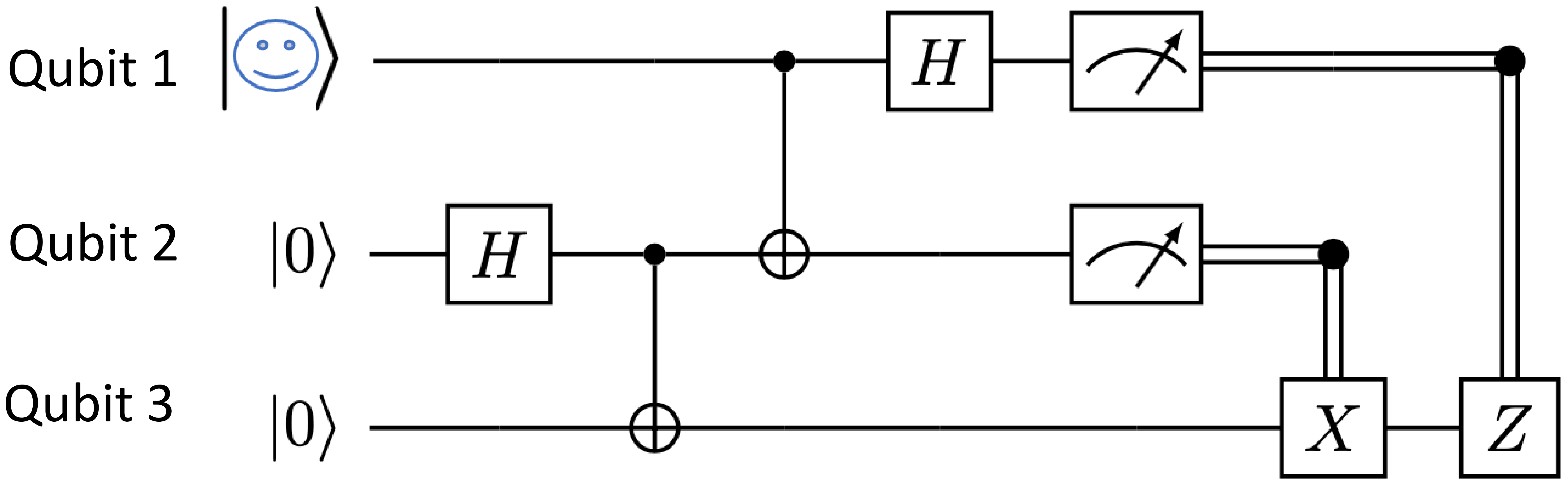
What do we need?

- Quantum memory (way to store qubits)
- Manipulate state of qubits (apply gates)
- Measure qubits

Let’s simulate it!



Teleport state from qubit 1 to qubit 3



What do we need?

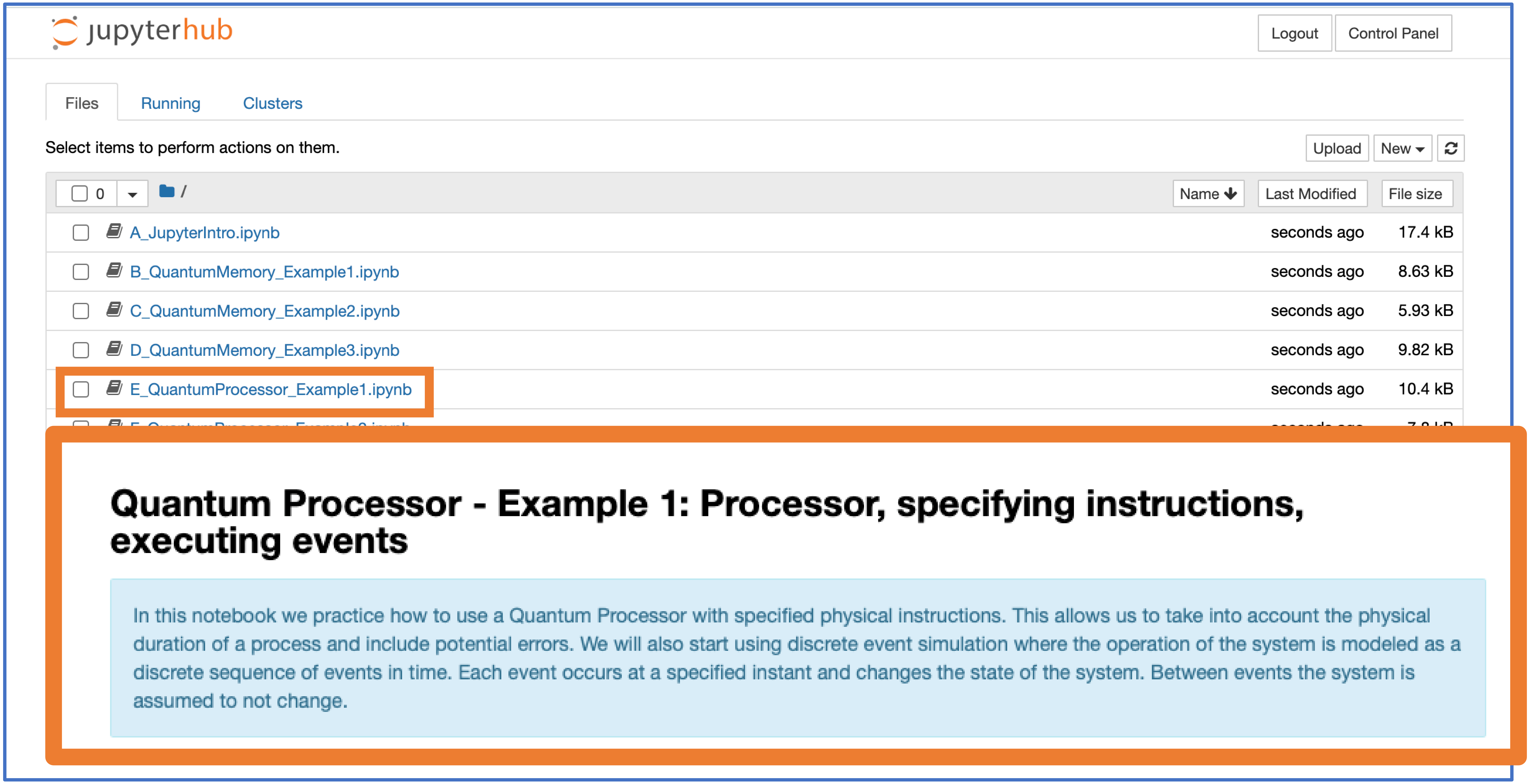
Let’s simulate it!

- Quantum memory (way to store qubits)
-
-

Problem: No physical instructions
(no execution times, no errors, just abstract circuit)

Center for
Quantum
Networks
NSF-ERC

Notebook: E_QuantumProcessor_Example1.ipyn



Select items to perform actions on them.

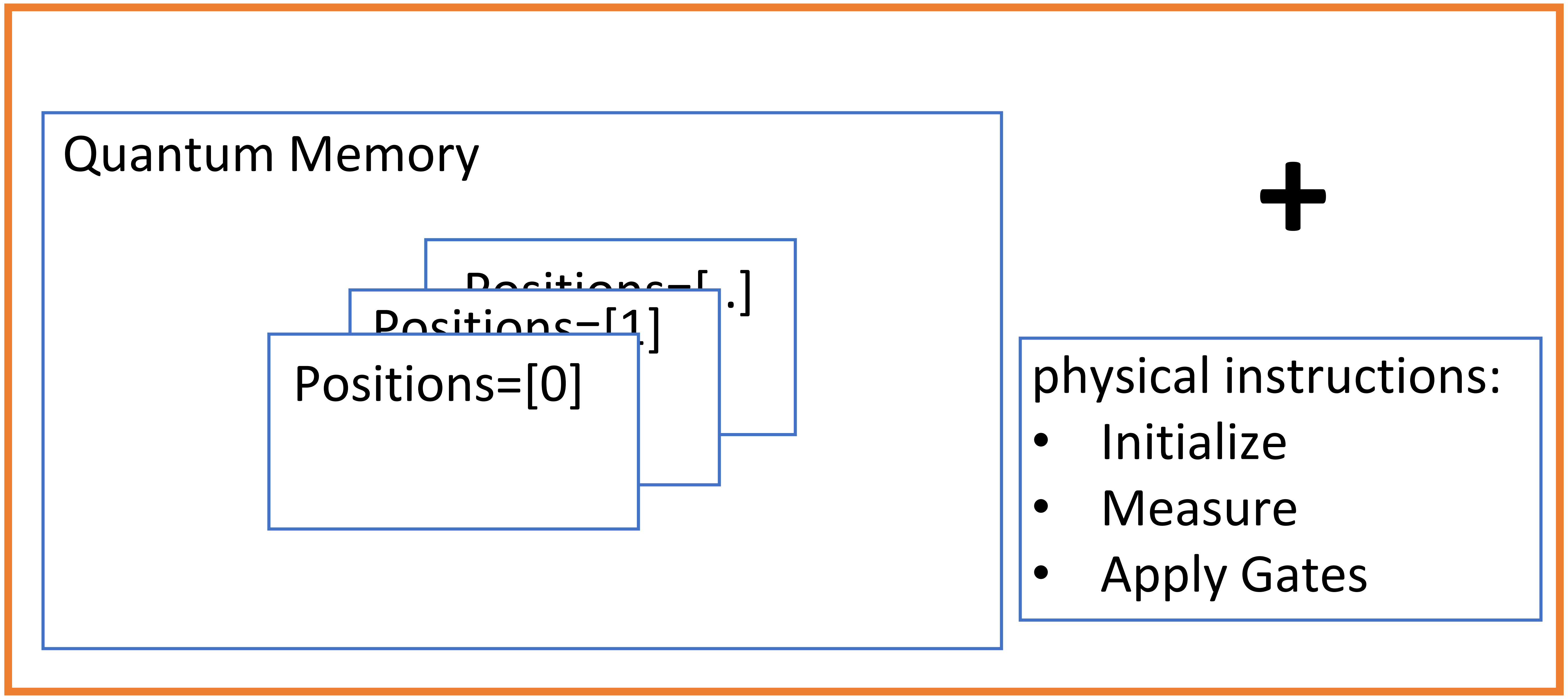
□ 0 /		
□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□ D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□ E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□ F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB

Quantum Processor - Example 1: Processor, specifying instructions, executing events

In this notebook we practice how to use a Quantum Processor with specified physical instructions. This allows us to take into account the physical duration of a process and include potential errors. We will also start using discrete event simulation where the operation of the system is modeled as a discrete sequence of events in time. Each event occurs at a specified instant and changes the state of the system. Between events the system is assumed to not change.

Notebook: E_QuantumProcessor_Example1.ipyn

Quantum Processor:



Quantum Memory

Positions=[..]

Positions=[1]

Positions=[0]

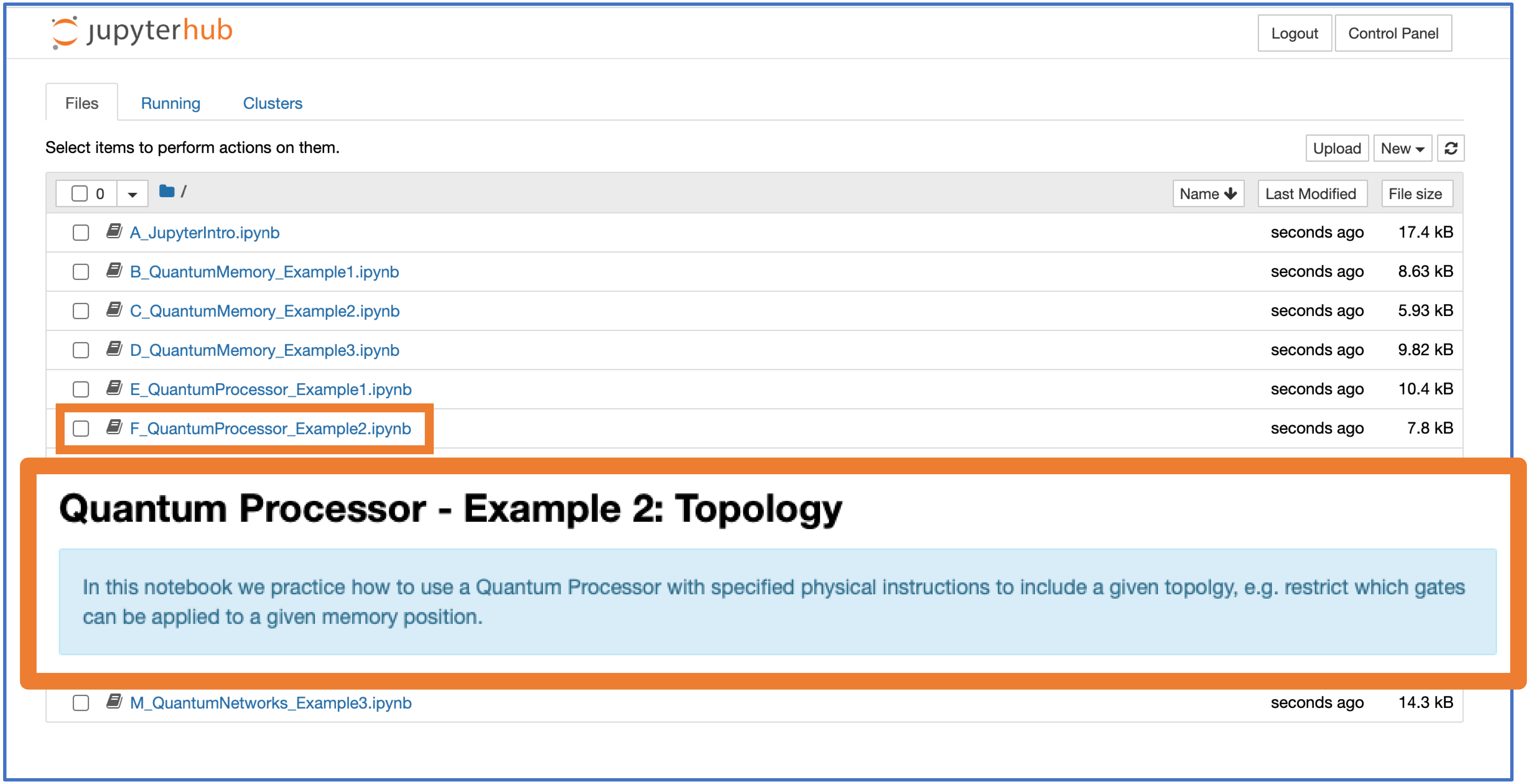
+

physical instructions:

- Initialize
- Measure
- Apply Gates

- Quantum processor
- Physical instructions (time duration)
- Discrete event simulation
- Operate on qubits in a quantum processor

Notebook: F_QuantumProcessor_Example2.ipyn



Select items to perform actions on them.

□ 0 /

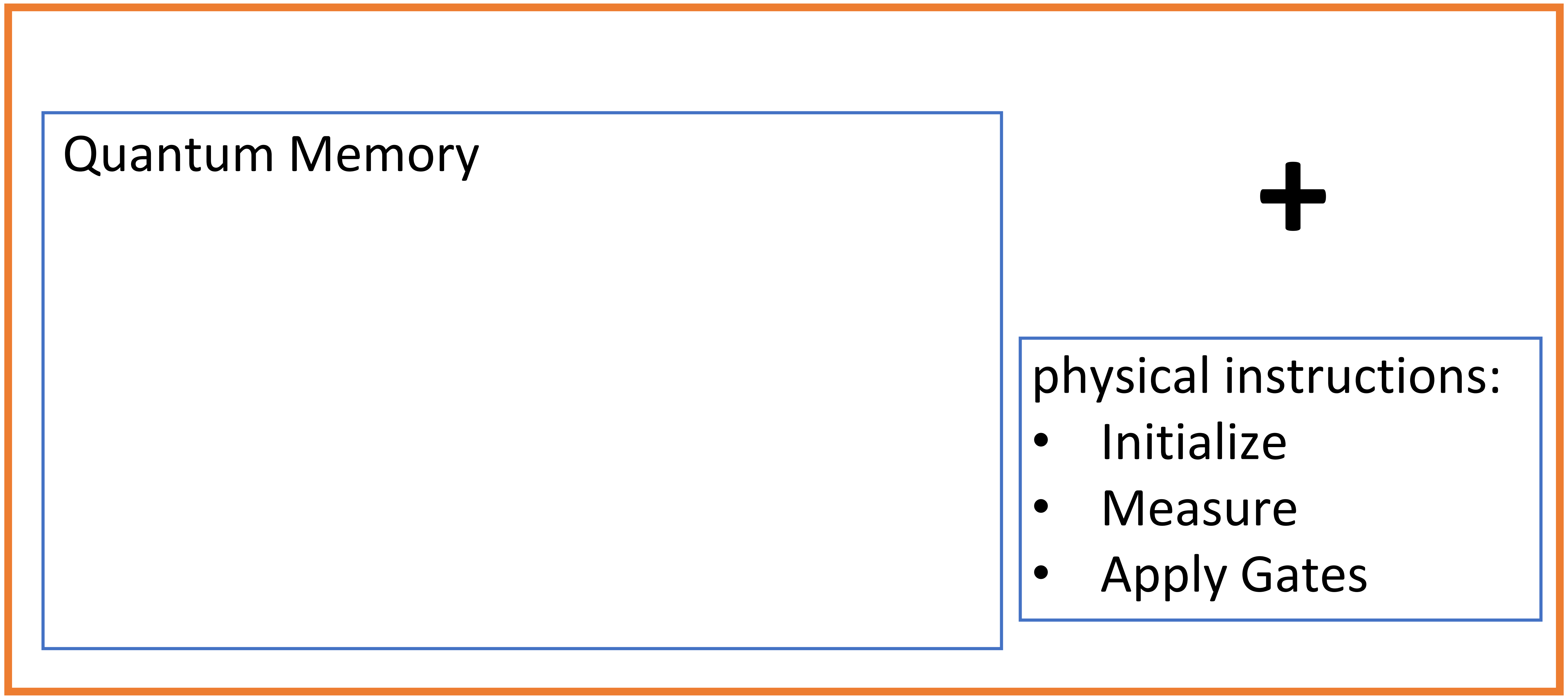
	Name	Last Modified	File size
□	A_JupyterIntro.ipynb	seconds ago	17.4 kB
□	B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□	C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□	D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□	E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□	F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
□	M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Processor - Example 2: Topology

In this notebook we practice how to use a Quantum Processor with specified physical instructions to include a given topolgy, e.g. restrict which gates can be applied to a given memory position.

Notebook: F_QuantumProcessor_Example2.ipyn

Quantum Processor:



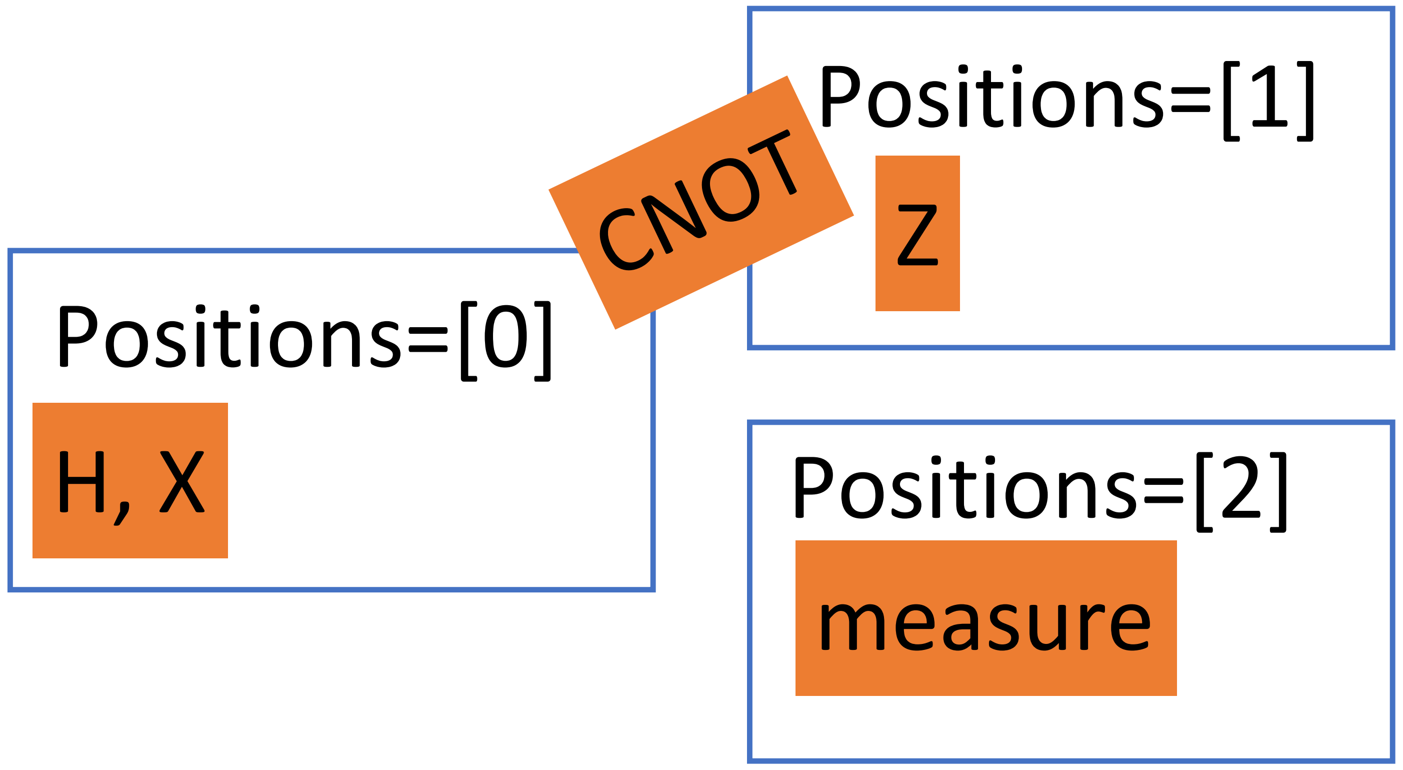
Quantum Memory

+

physical instructions:

- Initialize
- Measure
- Apply Gates

- Quantum processor
- Physical instructions **(topology)**
- Discrete event simulation
- Operate on qubits in a quantum processor



Positions=[1]

Z

CNOT

Positions=[0]

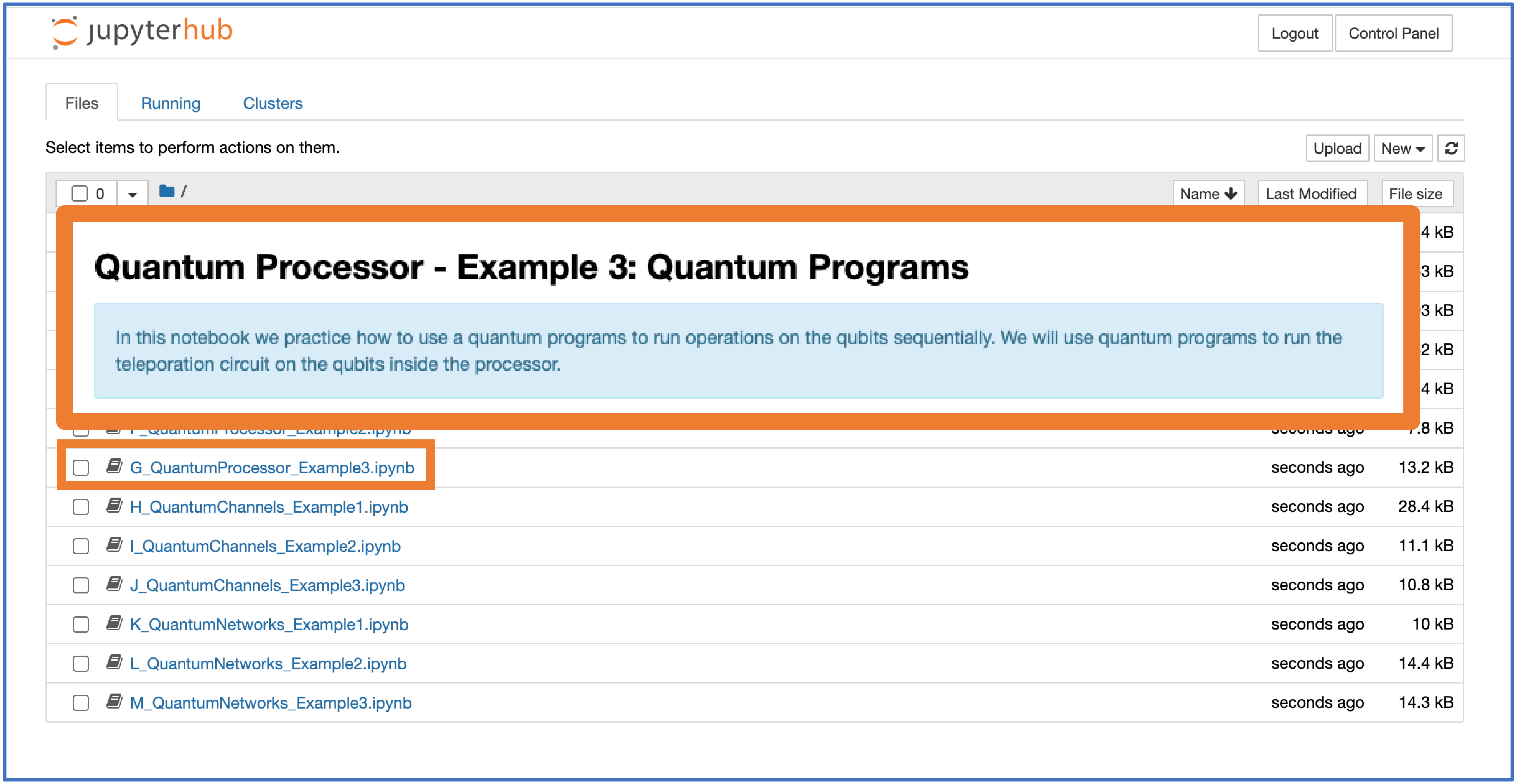
H, X

Positions=[2]

measure

Center for Quantum Networks *NSF-ERC*

Notebook: G_QuantumProcessor_Example3.ipyn



Select items to perform actions on them.

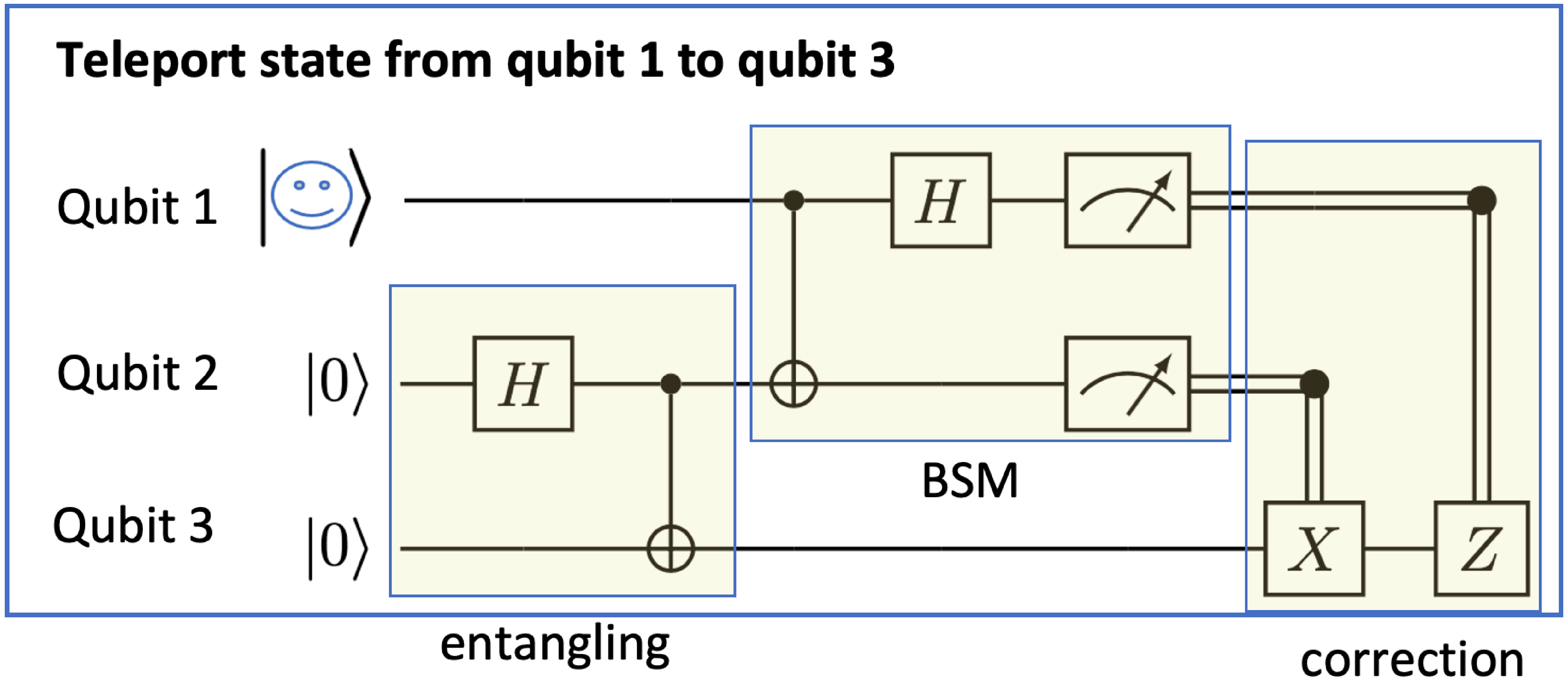
Quantum Processor - Example 3: Quantum Programs

In this notebook we practice how to use a quantum programs to run operations on the qubits sequentially. We will use quantum programs to run the teleporation circuit on the qubits inside the processor.

Name	Last Modified	File size
		4 kB
		3 kB
		3 kB
		2 kB
		4 kB
F_QuantumProcessor_Example2.ipynb	seconds ago	7.8 kB
□ G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
□ H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□ I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Notebook: G_QuantumProcessor_Example3.ipyn

Quantum Memory



- Quantum program for teleportation circuit
- Physical instructions (time duration, topology)

Quantum State Teleportation - Refresher

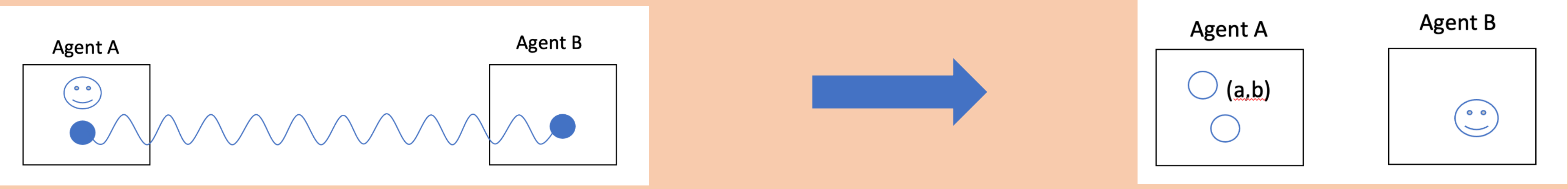
Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.

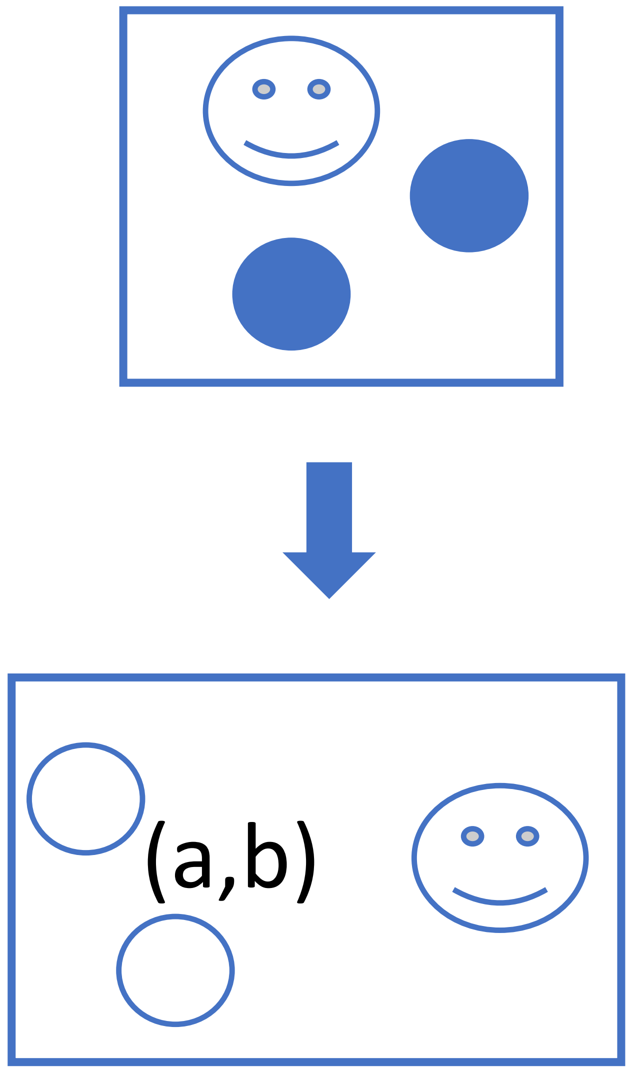
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.

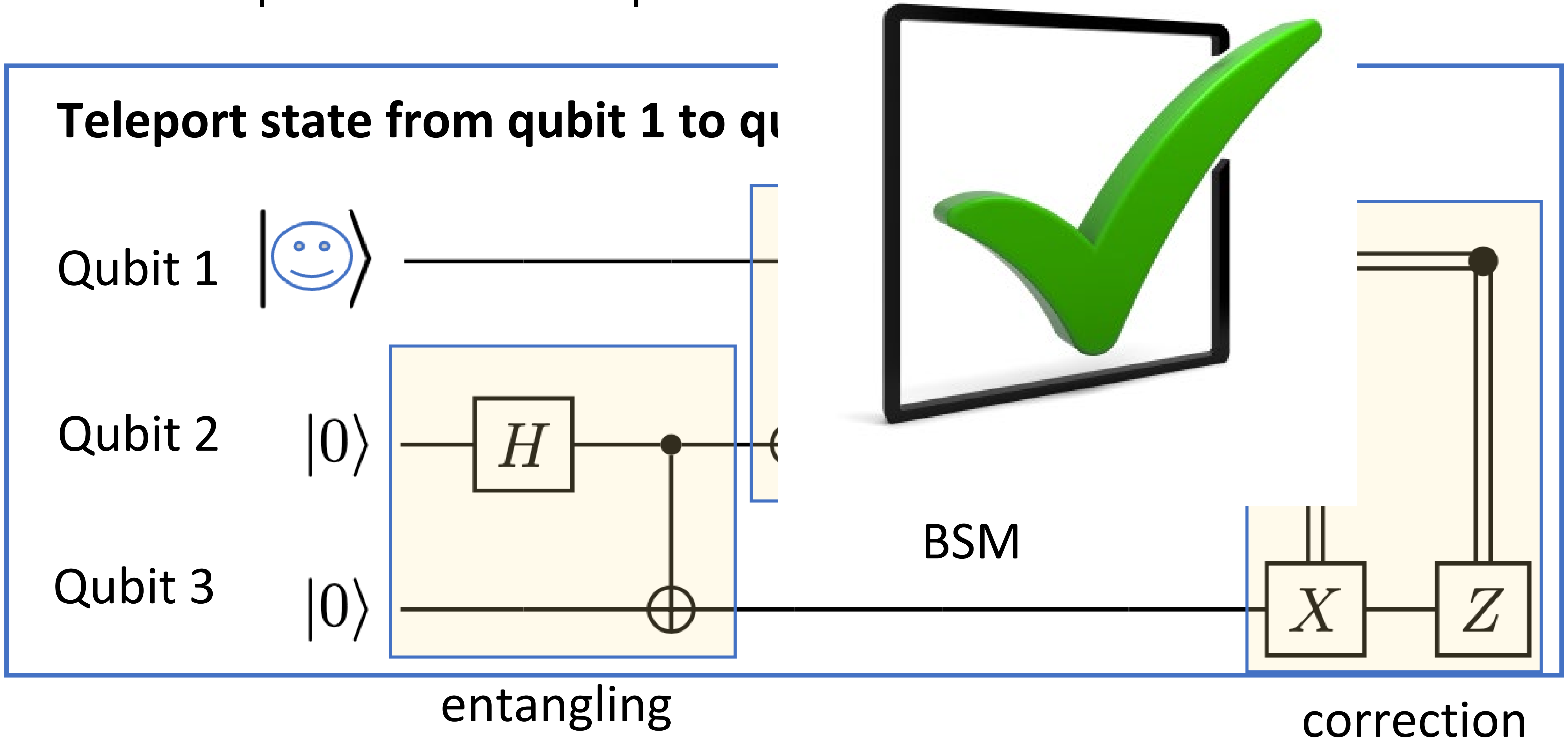
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).

Step 3: Agent B uses the classical bits to correct state of local qubit.



Or: Goal: Teleport state of one qubit to another qubit

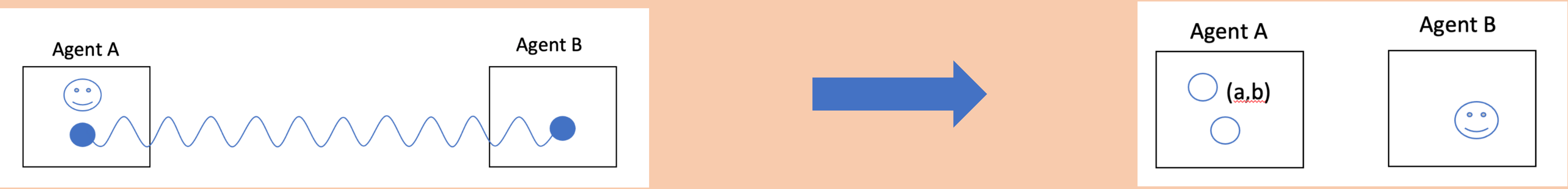


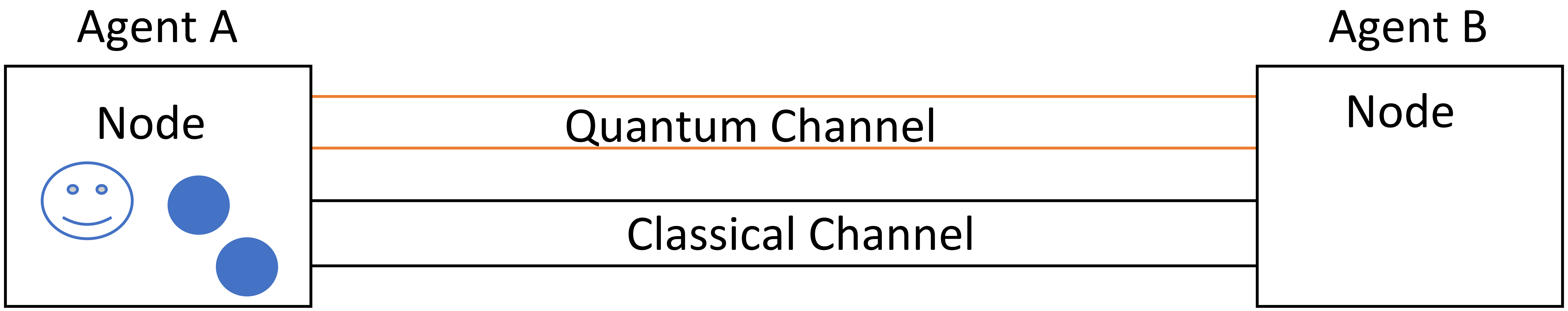


Quantum State Teleportation - Refresher

Goal: Teleport state of qubit from Agent A to Agent B

Step 0: Agent A and Agent B share an entangled qubit pair.
Step 1: Agent A performs a joint (or Bell State) measurement on the local qubits.
Step 2: Agent A send the measurement outcomes to Agent B (classical bits).
Step 3: Agent B uses the classical bits to correct state of local qubit.





Agent A

Node

Quantum Channel

Classical Channel

Agent B

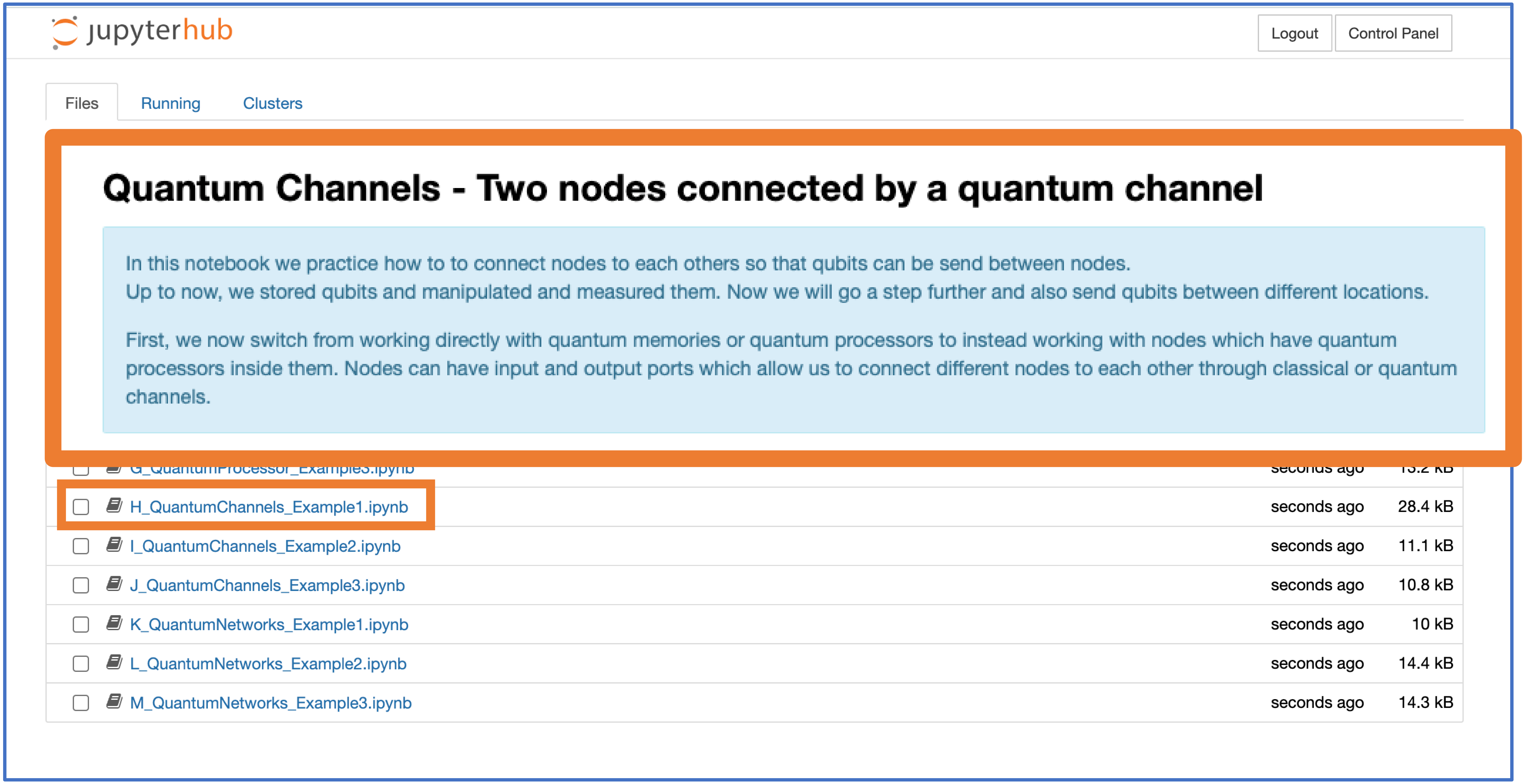
Node

Let’s simulate it!

What do we need?

- Everything we covered so far
- Channels
- Connections
- Node Protocols

Notebook: H_QuantumChannels_Example1.ipyn



Quantum Channels - Two nodes connected by a quantum channel

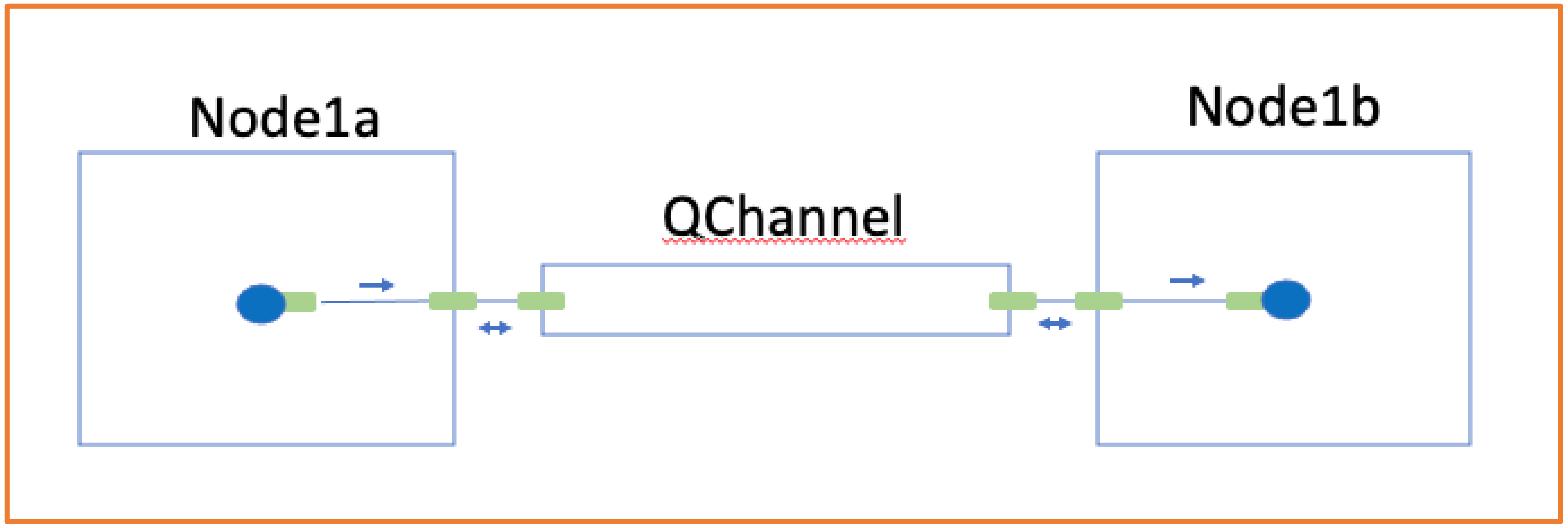
In this notebook we practice how to to connect nodes to each others so that qubits can be send between nodes.
Up to now, we stored qubits and manipulated and measured them. Now we will go a step further and also send qubits between different locations.

First, we now switch from working directly with quantum memories or quantum processors to instead working with nodes which have quantum processors inside them. Nodes can have input and output ports which allow us to connect different nodes to each other through classical or quantum channels.

□	G_QuantumProcessor_Example3.ipynb	seconds ago	13.2 kB
□	H_QuantumChannels_Example1.ipynb	seconds ago	28.4 kB
□	I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□	J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□	K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□	L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□	M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

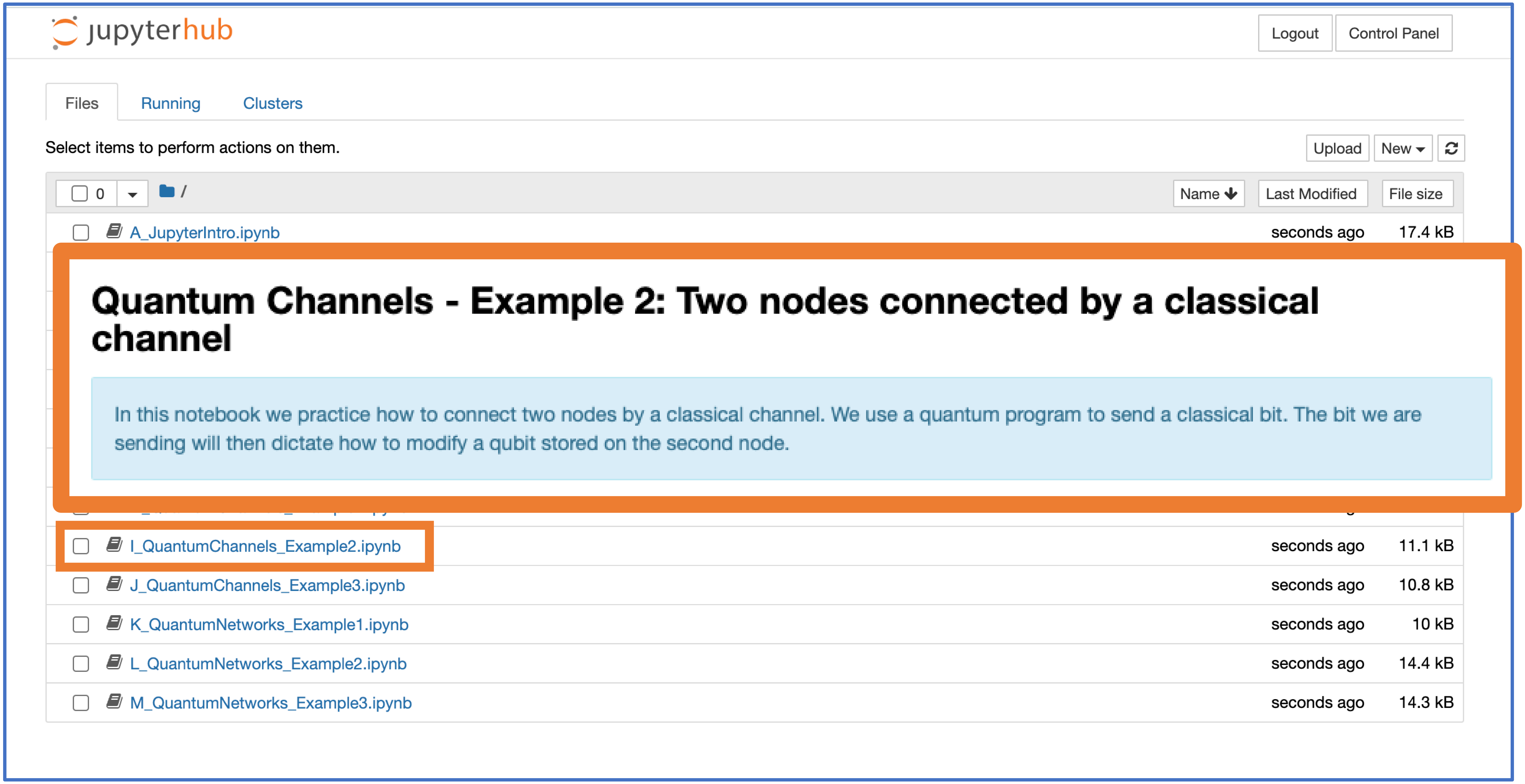
Center for Quantum Networks *NSF-ERC*

Notebook: H_QuantumChannels_Example1.ipyn



- Nodes (with processor and memory)
- Input/Output ports
- Quantum Channel

Notebook: I_QuantumChannels_Example2.ipyn



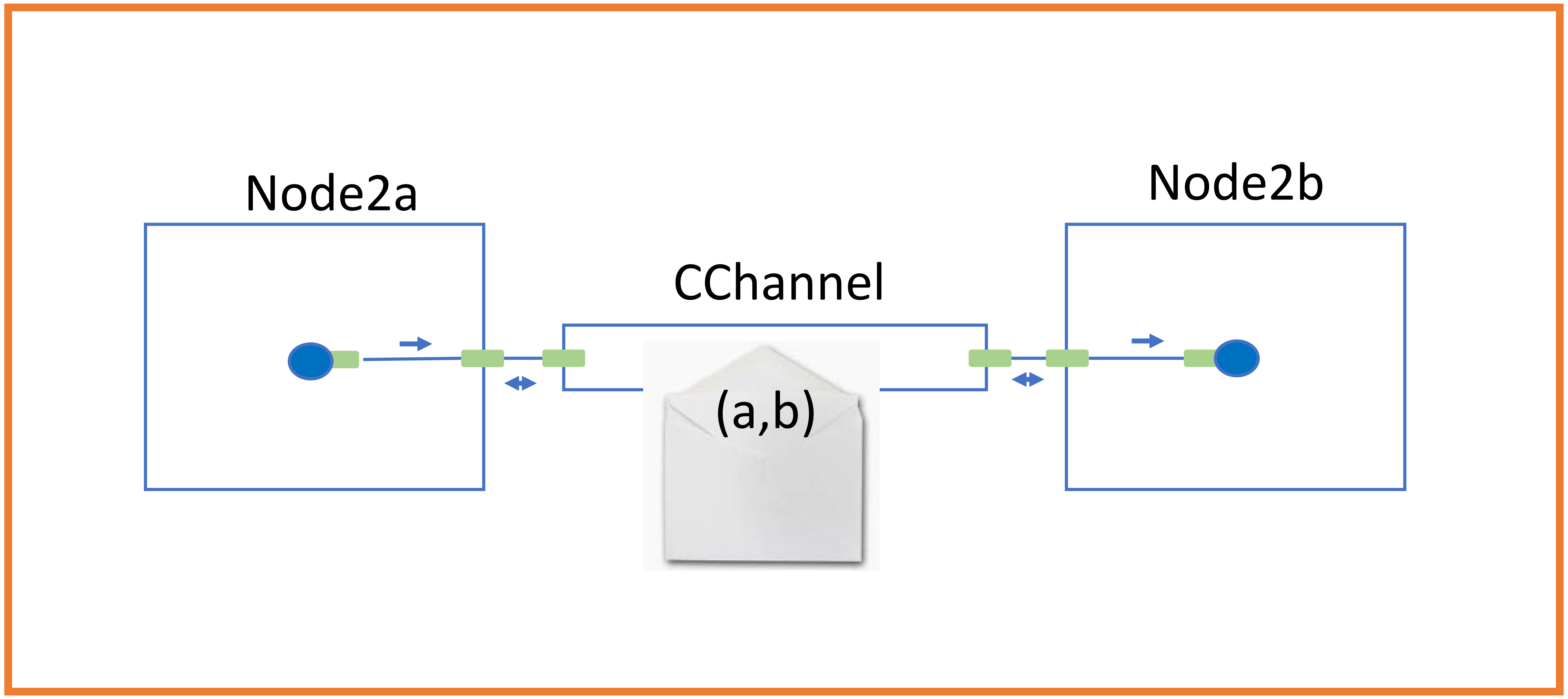
Select items to perform actions on them.

A_JupyterIntro.ipynb	seconds ago	17.4 kB
I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Channels - Example 2: Two nodes connected by a classical channel

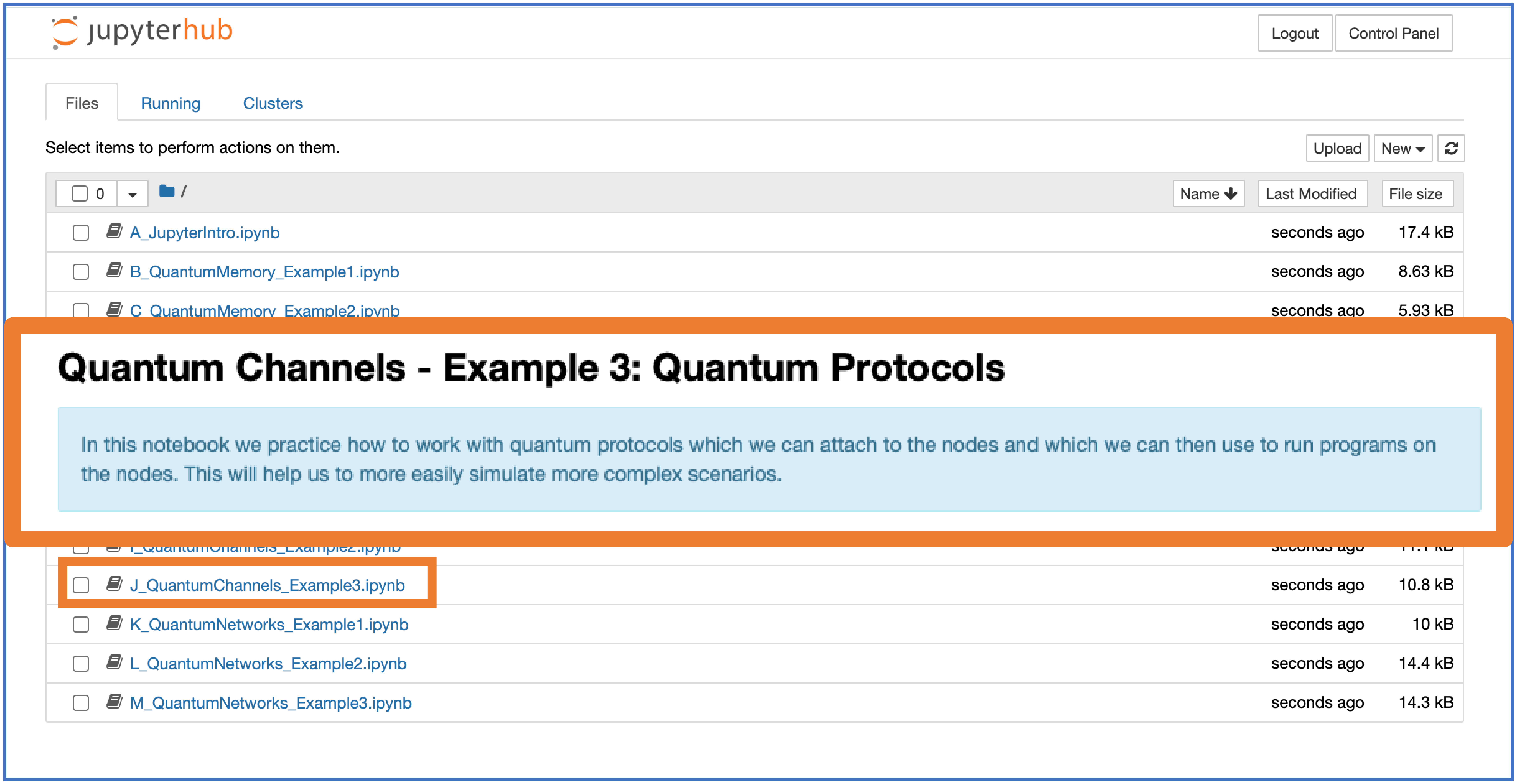
In this notebook we practice how to connect two nodes by a classical channel. We use a quantum program to send a classical bit. The bit we are sending will then dictate how to modify a qubit stored on the second node.

Notebook: I_QuantumChannels_Example2.ipyn



- Nodes (with processor and memory)
- Input/Output ports
- Classical Channel
- Quantum Programs

Notebook: J_QuantumChannels_Example3.ipyn



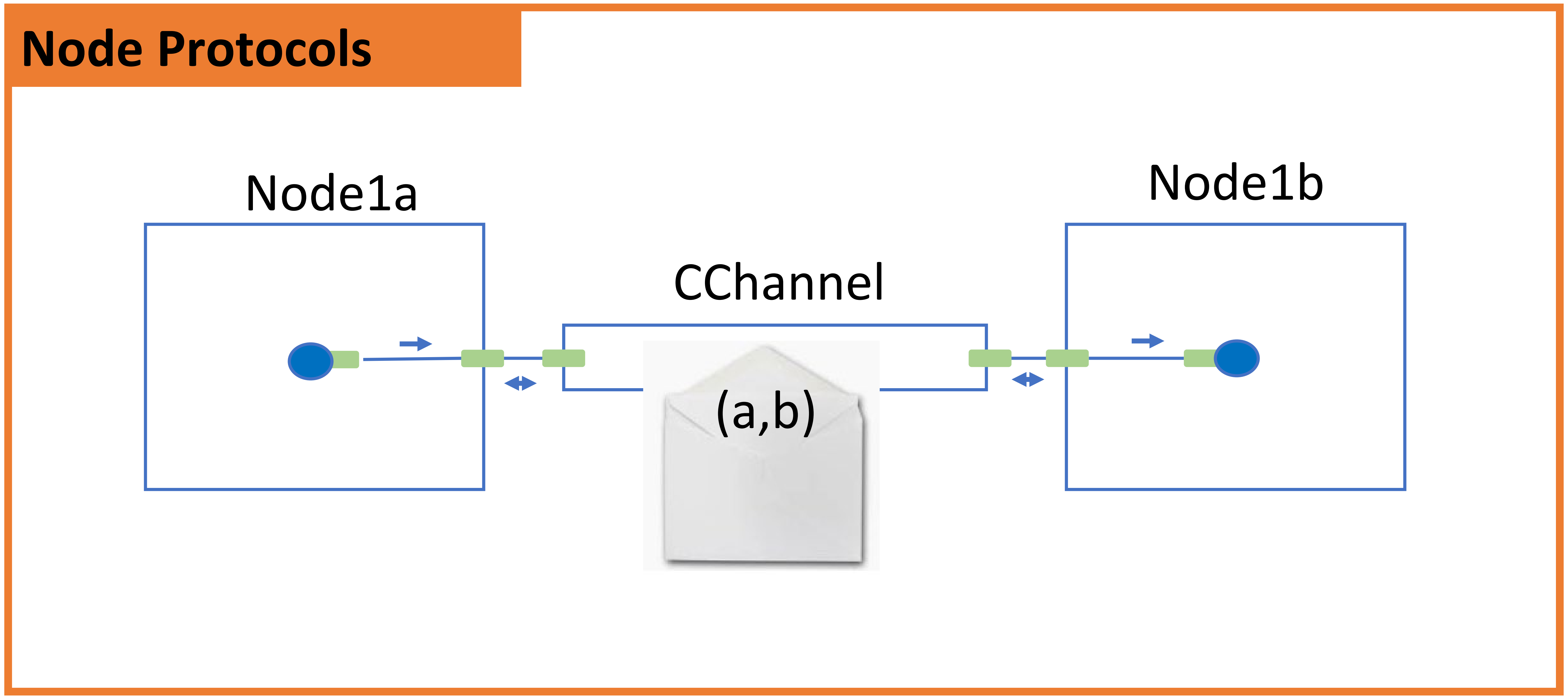
Select items to perform actions on them.

Name	Last Modified	File size
□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□ I_QuantumChannels_Example2.ipynb	seconds ago	11.1 kB
□ J_QuantumChannels_Example3.ipynb	seconds ago	10.8 kB
□ K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Channels - Example 3: Quantum Protocols

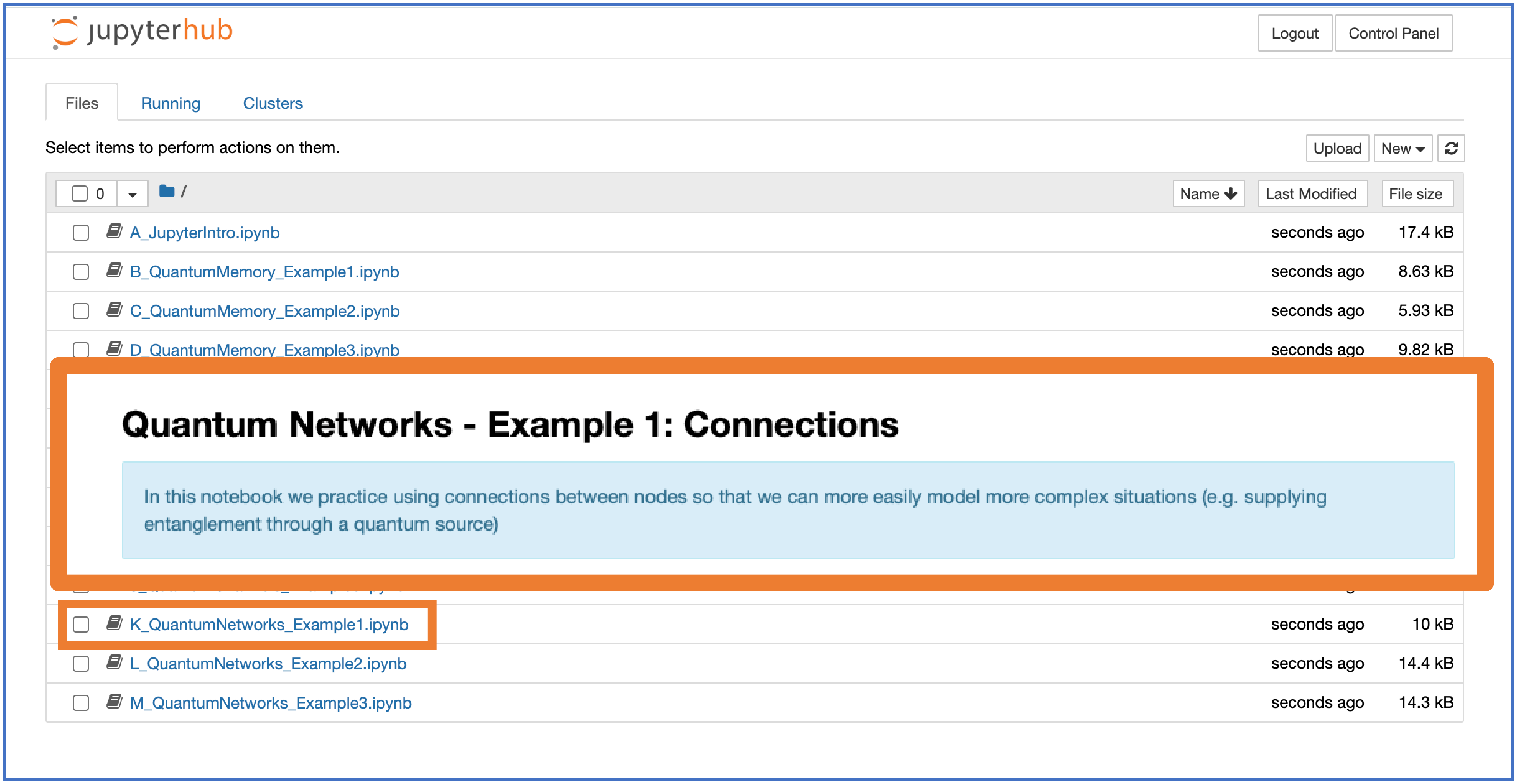
In this notebook we practice how to work with quantum protocols which we can attach to the nodes and which we can then use to run programs on the nodes. This will help us to more easily simulate more complex scenarios.

Notebook: J_QuantumChannels_Example3.ipyn



- Nodes (with processor and memory)
- Input/Output ports
- Classical Channel
- **Node Protocols**

Notebook: K_QuantumNetworks_Example1.ipyn



Select items to perform actions on them.

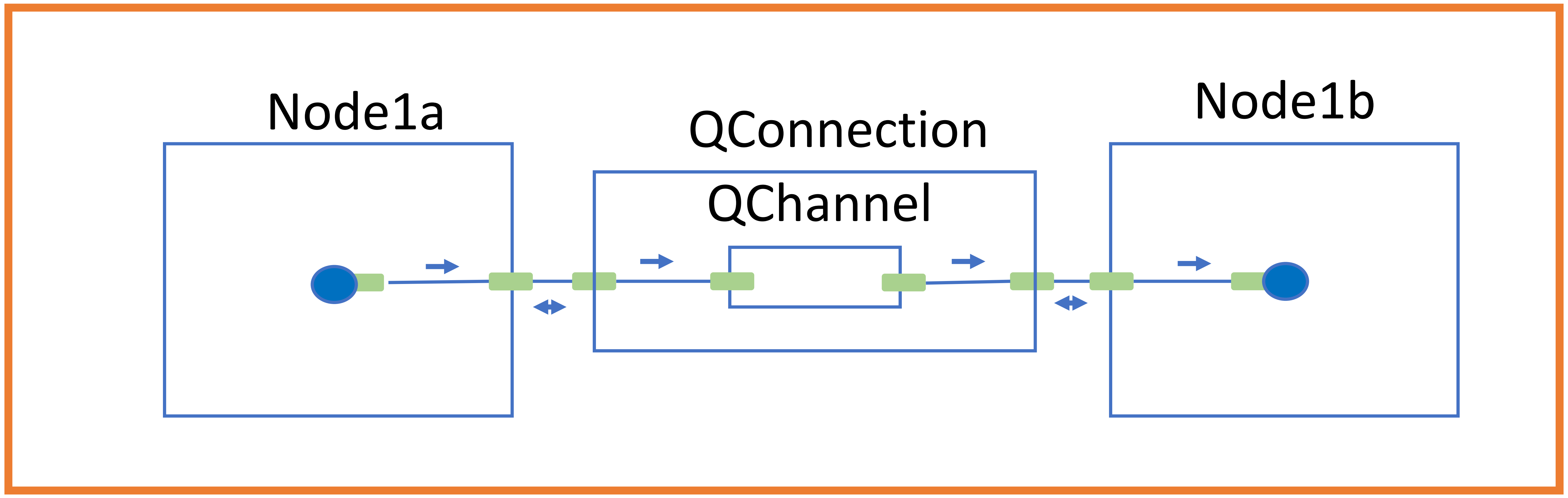
A_JupyterIntro.ipynb	seconds ago	17.4 kB
B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
K_QuantumNetworks_Example1.ipynb	seconds ago	10 kB
L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Networks - Example 1: Connections

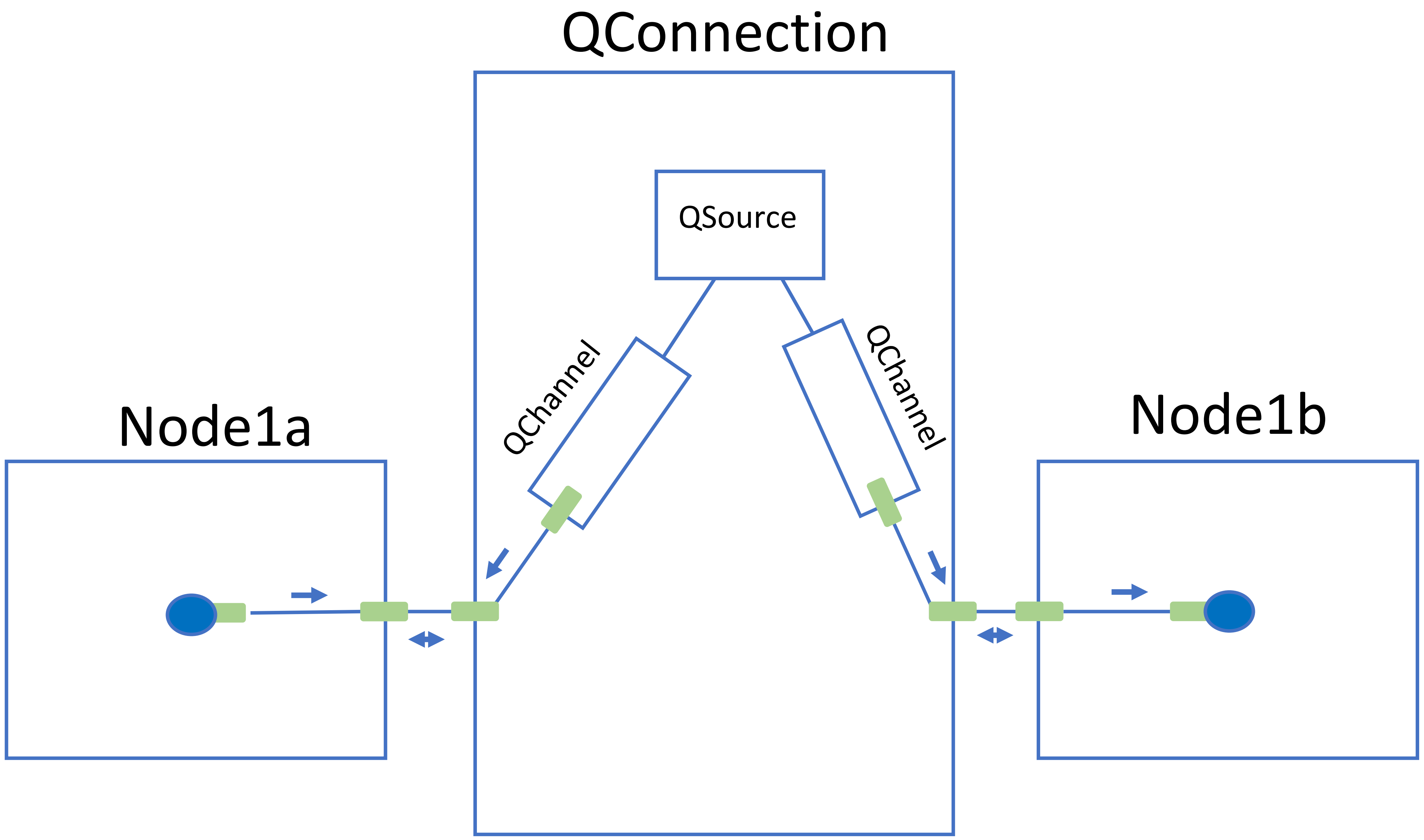
In this notebook we practice using connections between nodes so that we can more easily model more complex situations (e.g. supplying entanglement through a quantum source)

Center for Quantum Networks *NSF-ERC*

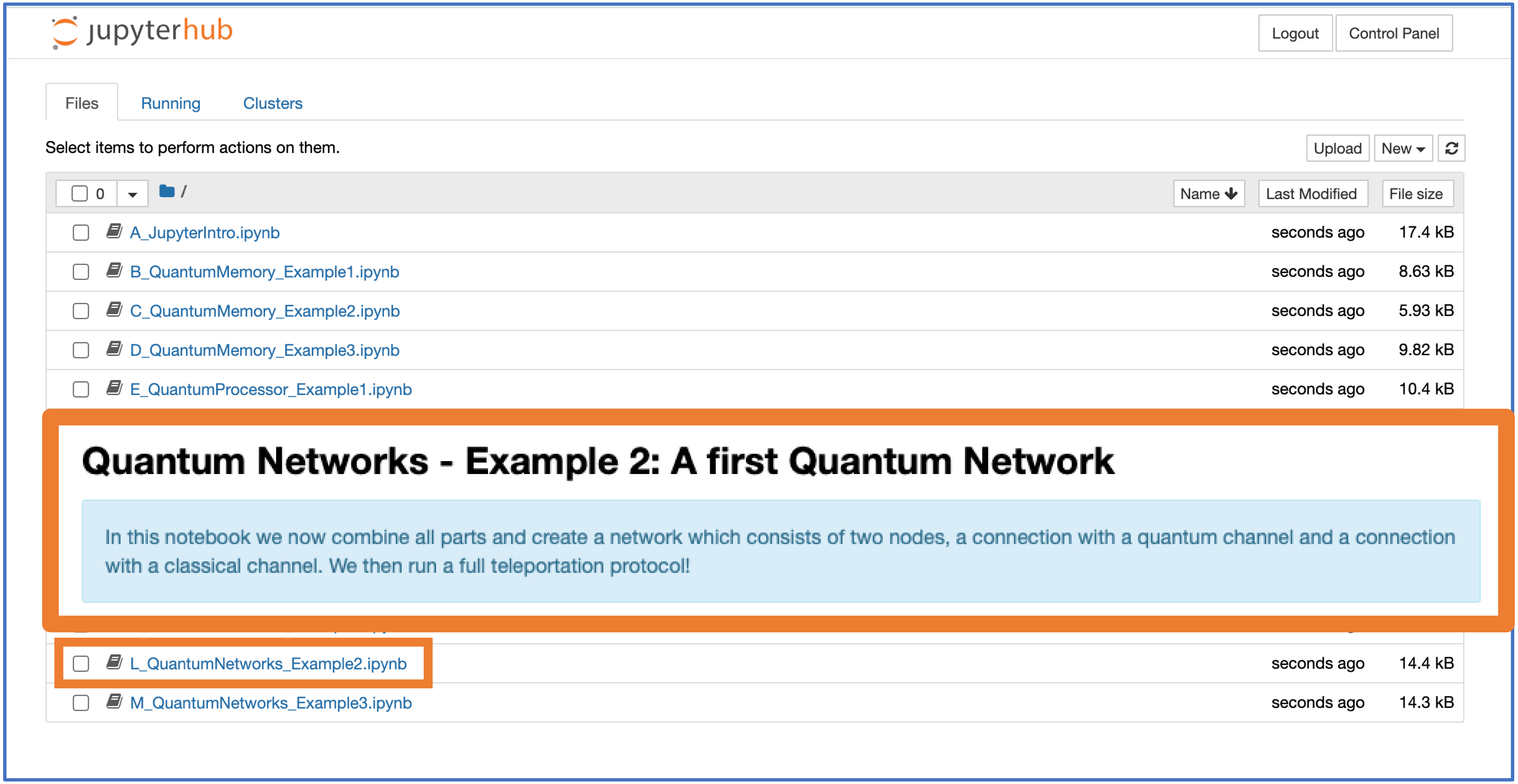
Notebook: K_QuantumNetworks_Example1.ipyn



- Quantum Connections
- Quantum Programs
- Node Protocols



Notebook: L_QuantumNetworks_Example2.ipyn



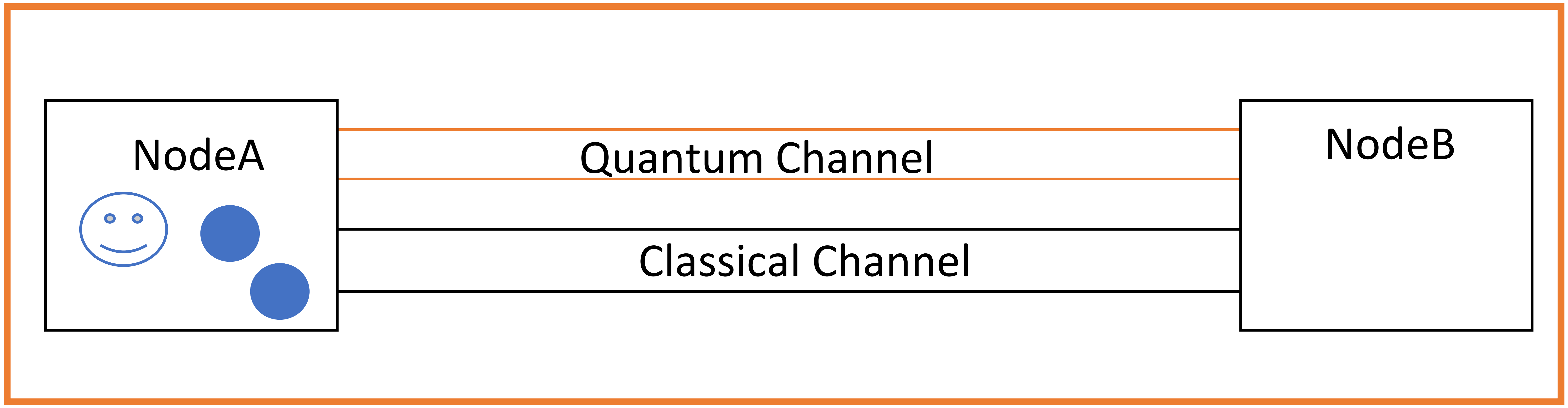
Select items to perform actions on them.

Name	Last Modified	File size
□ A_JupyterIntro.ipynb	seconds ago	17.4 kB
□ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
□ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
□ D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
□ E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
□ L_QuantumNetworks_Example2.ipynb	seconds ago	14.4 kB
□ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Networks - Example 2: A first Quantum Network

In this notebook we now combine all parts and create a network which consists of two nodes, a connection with a quantum channel and a connection with a classical channel. We then run a full teleportation protocol!

Notebook: L_QuantumNetworks_Example2.ipyn



NodeA

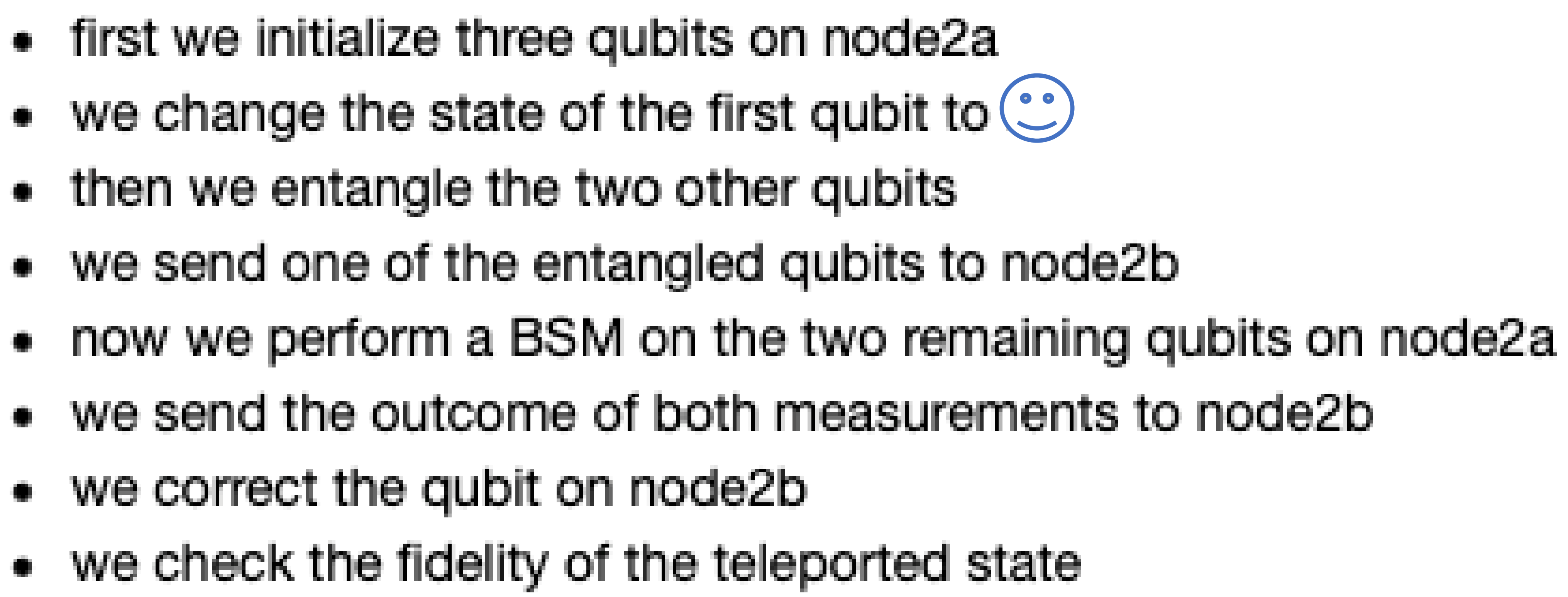
Quantum Channel

Classical Channel

NodeB

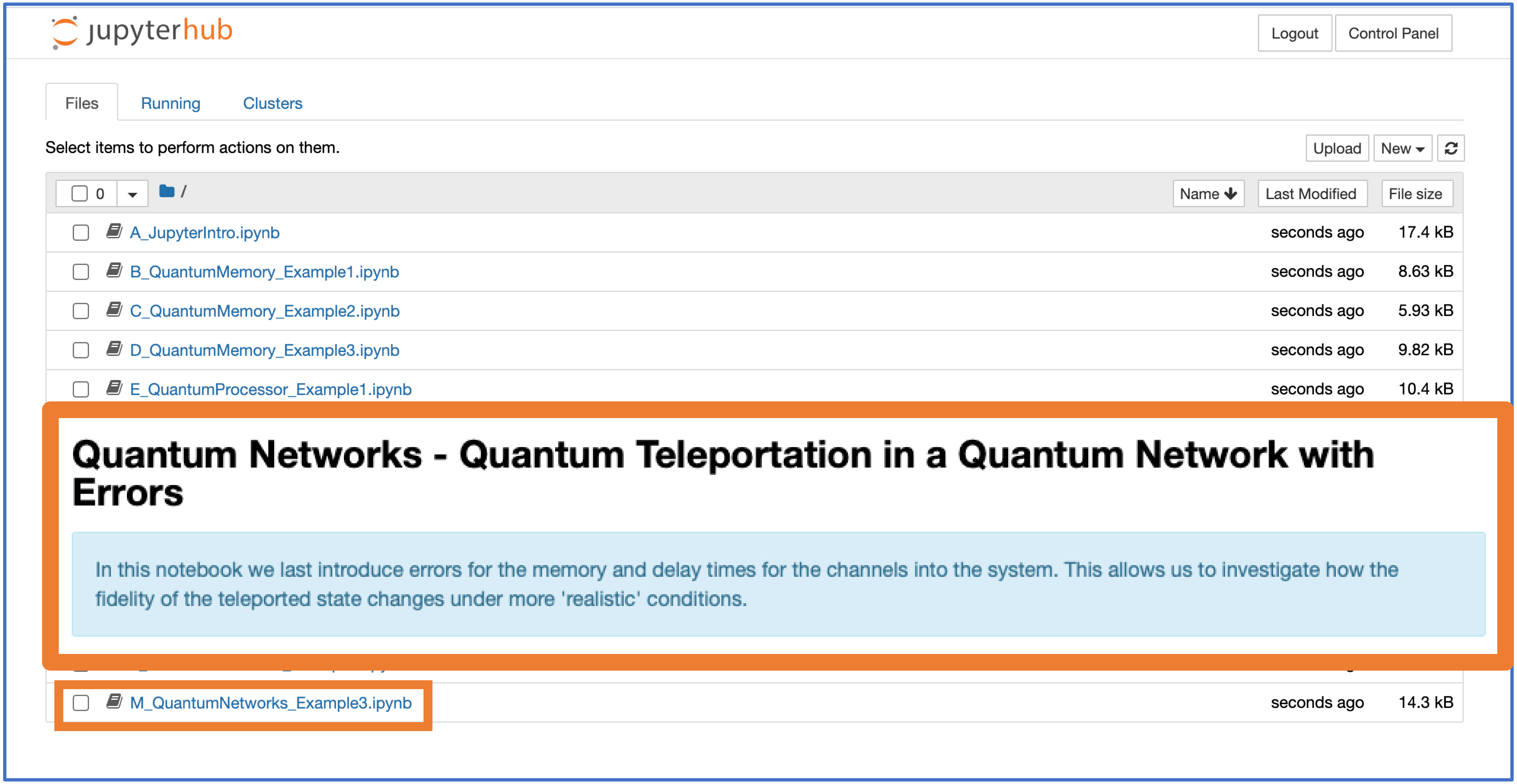
Full network with

- Nodes
- Connections
- Quantum Programs
- Node Protocols



- first we initialize three qubits on node2a
- we change the state of the first qubit to ☺
- then we entangle the two other qubits
- we send one of the entangled qubits to node2b
- now we perform a BSM on the two remaining qubits on node2a
- we send the outcome of both measurements to node2b
- we correct the qubit on node2b
- we check the fidelity of the teleported state

Notebook: M_QuantumNetworks_Example3.ipyn



Select items to perform actions on them.

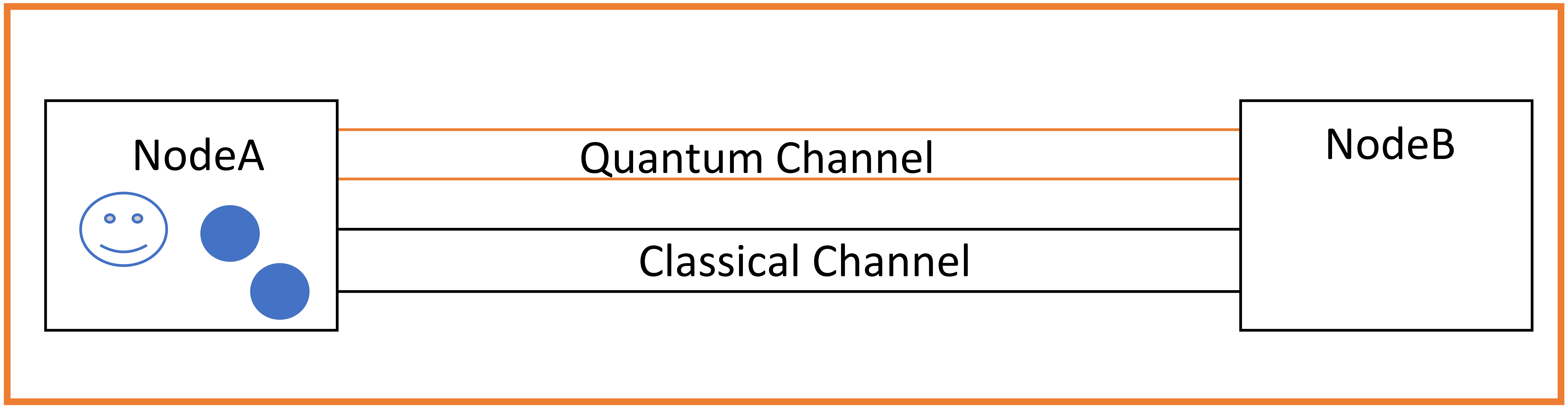
☐ 0 /

☐ A_JupyterIntro.ipynb	seconds ago	17.4 kB
☐ B_QuantumMemory_Example1.ipynb	seconds ago	8.63 kB
☐ C_QuantumMemory_Example2.ipynb	seconds ago	5.93 kB
☐ D_QuantumMemory_Example3.ipynb	seconds ago	9.82 kB
☐ E_QuantumProcessor_Example1.ipynb	seconds ago	10.4 kB
☐ M_QuantumNetworks_Example3.ipynb	seconds ago	14.3 kB

Quantum Networks - Quantum Teleportation in a Quantum Network with Errors

In this notebook we last introduce errors for the memory and delay times for the channels into the system. This allows us to investigate how the fidelity of the teleported state changes under more 'realistic' conditions.

Notebook: M_QuantumNetworks_Example3.ipyn



NodeA

Quantum Channel

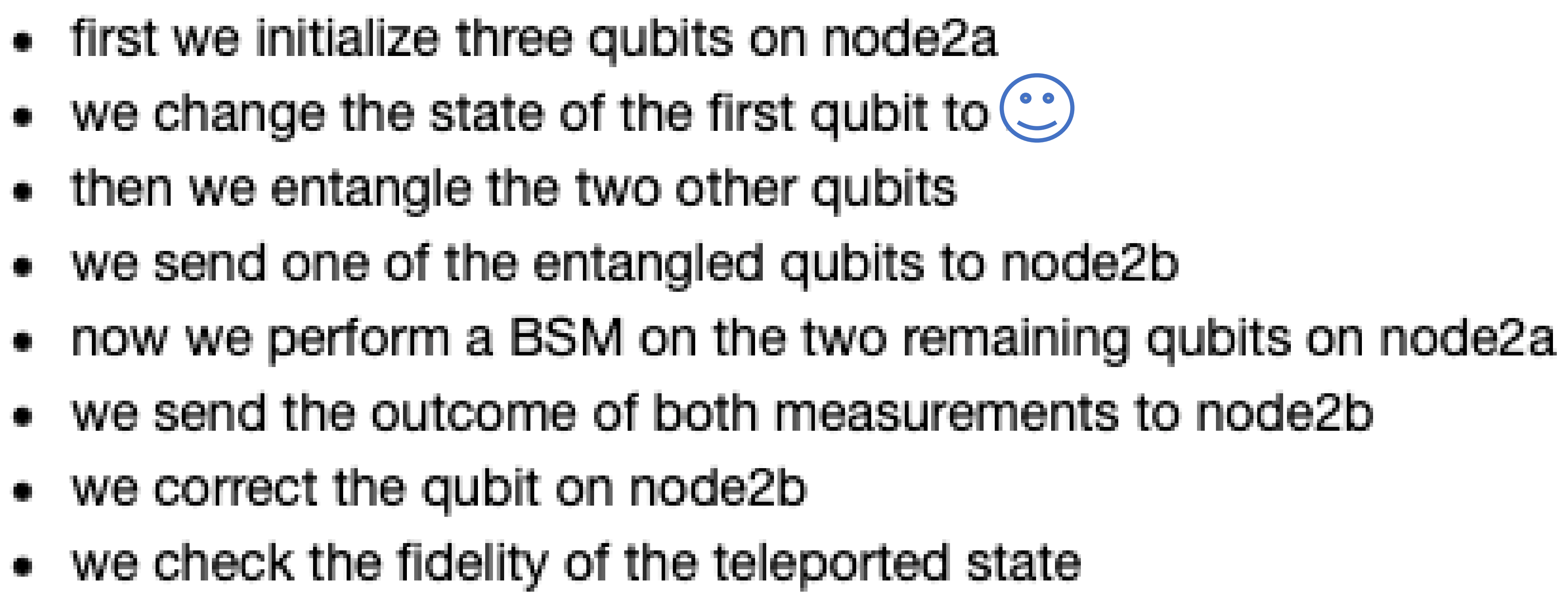
Classical Channel

NodeB

With ERRORS!

Full network with

- Nodes
- Connections
- Quantum Programs
- Node Protocols



- first we initialize three qubits on node2a
- we change the state of the first qubit to ☺
- then we entangle the two other qubits
- we send one of the entangled qubits to node2b
- now we perform a BSM on the two remaining qubits on node2a
- we send the outcome of both measurements to node2b
- we correct the qubit on node2b
- we check the fidelity of the teleported state

Interested in using a quantum network simulator to:

- Explore concepts?
- Integrate it in your research?
- Or just have fun with it?

Hopefully this short course can **help to get you started** in doing **SOMETHING THAT YOU CONSIDER USEFUL** with a quantum network simulator

If want to keep using the notebooks:

- Download your notebooks
- Install NetSquid

Center for
Quantum Networks
NSF Engineering Research Center

Course Evaluation Survey

We value your feedback on all aspects of this short course. Please go to the link provided in the Zoom Chat or in the email you will soon receive to give your opinions of what worked and what could be improved.

CQN Winter School on Quantum Networks

Funded by National Science Foundation Grant #1941583