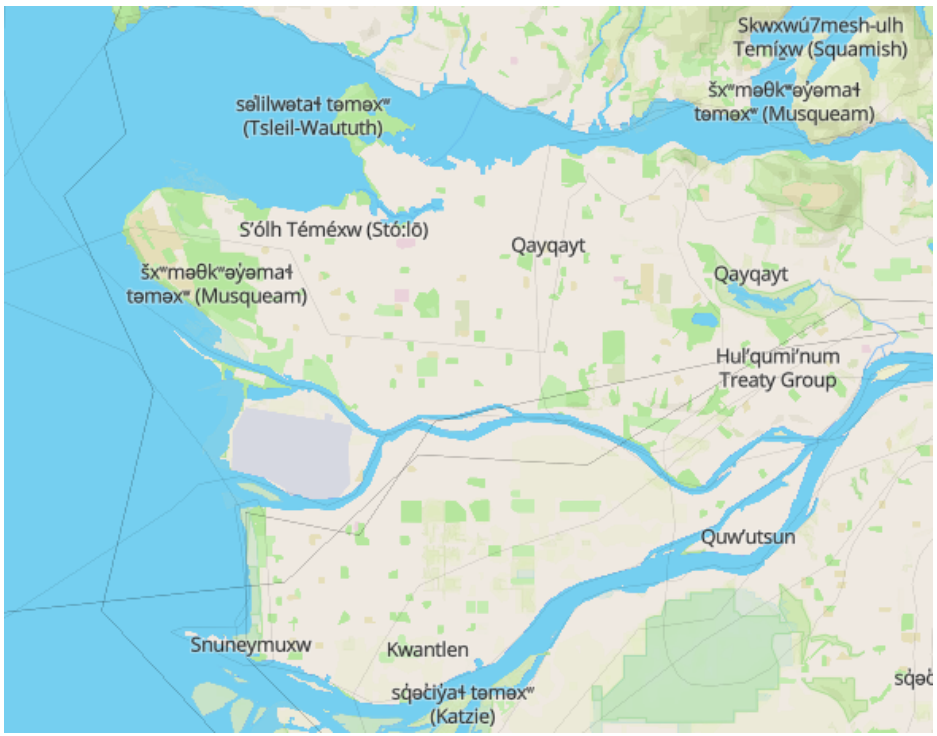
CPEN 400Q Lecture 01
Overview and intro to gate model quantum computing

Monday 9 January 2022

Land acknowledgement

The land on which we gather today is the traditional, ancestral, and unceded territory of the Musqueam People.



Thunder Bay is located on the traditional lands of the Fort William Fist Nation, Signatory to the Robinson-Superior Treaty of 1850. Waterloo is in the traditional territory of the Neutral, Anishinaabeg, and Haudenosaunee peoples.

Explore more:

- https://www.musqueam.bc.ca/
- https://native-land.ca/

Intros

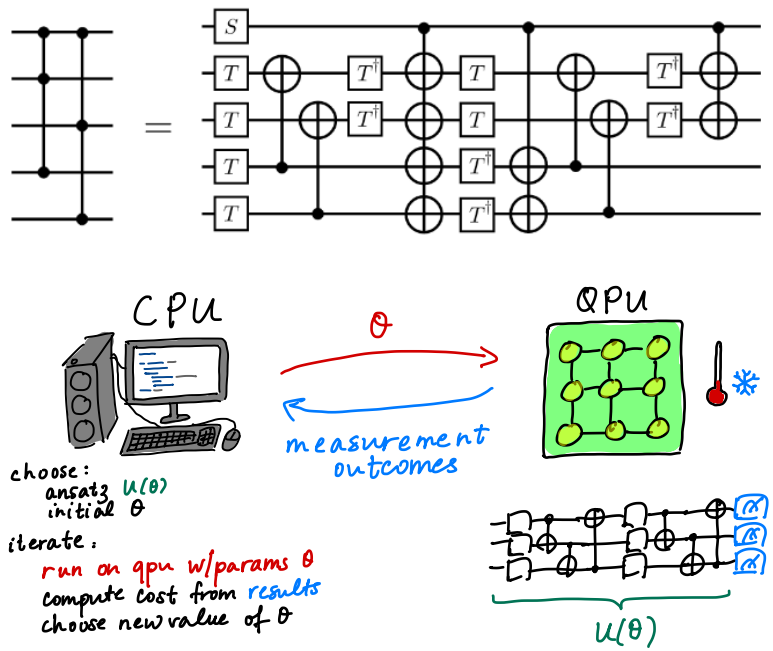
Olivia Di Matteo – KAIS 3043 – olivia@ece.ubc.ca

I have a physics background and I work on the **software and algorithms** side of **quantum computing**.

```python
import pennylane as qml

dev = qml.device("default.qubit", wires=3)

@qml.qnode(dev, diff_method="parameter-shift")
@qml.compile()
def circuit(data, params):
    qml.AngleEmbedding(data, wires=range(3))
    qml.RY(params[0], wires=0)
    qml.RX(params[1], wires=1)
    qml.RZ(params[2], wires=2)
    qml.CNOT(wires=[0, 1])
    qml.CNOT(wires=[1, 2])
    qml.CNOT(wires=[2, 0])
    return qml.expval(qml.PauliZ(0))
```



(Find me on GitHub as glassnotes.)

Why quantum computing?

How do we make computers more powerful?

Why quantum computing?

How do we make computers more powerful?

- make smaller transistors
- put more transistors onto a single chip
- use multiple processors in parallel

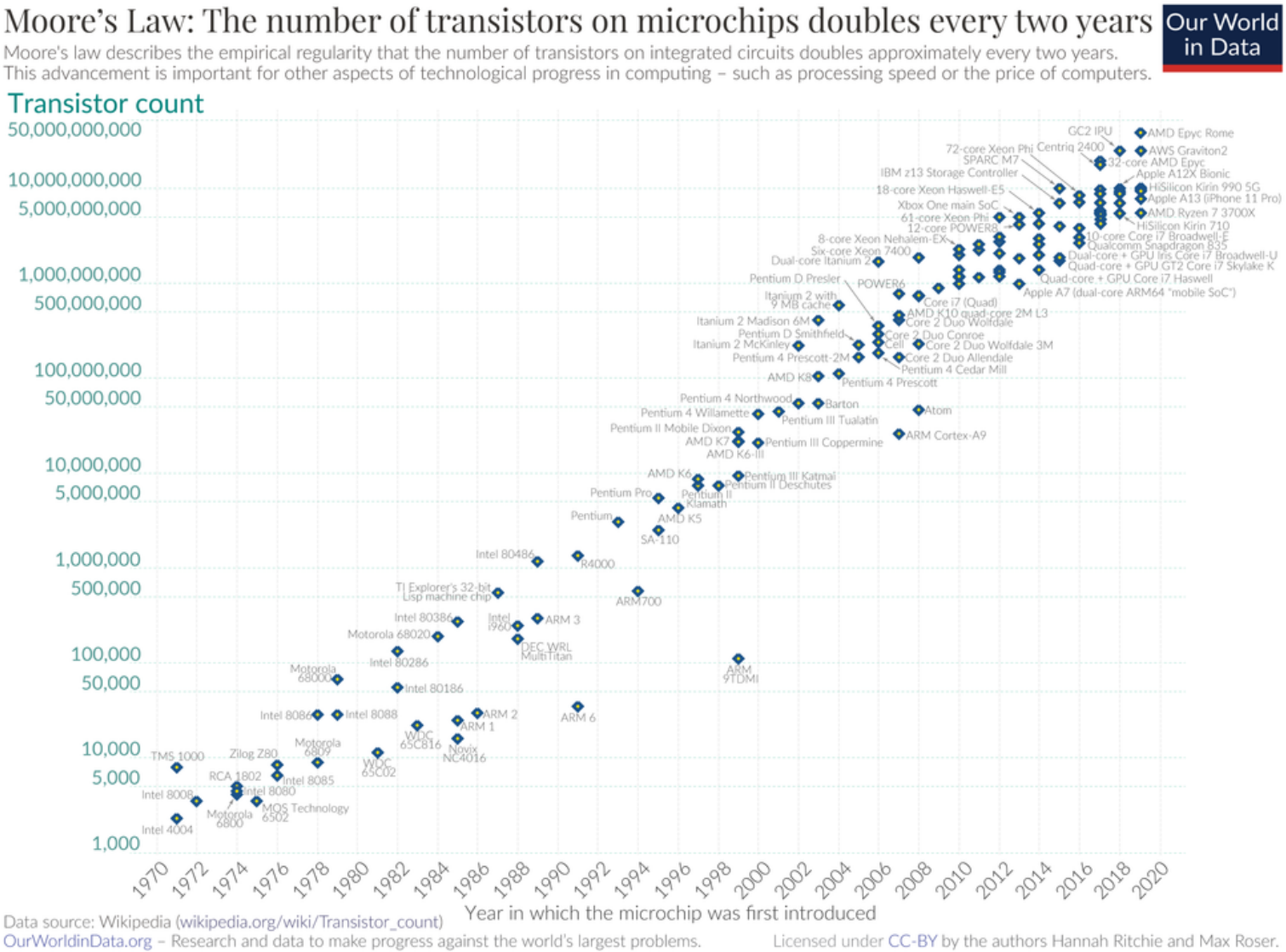
Frontier (ORNL): >8.7 million cores, >1.1 exaFLOPS

Image credit:

https://www.olcf.ornl.gov/wp-content/themes/olcf-edition-child/frontier-assets/systems/frontier.jpg

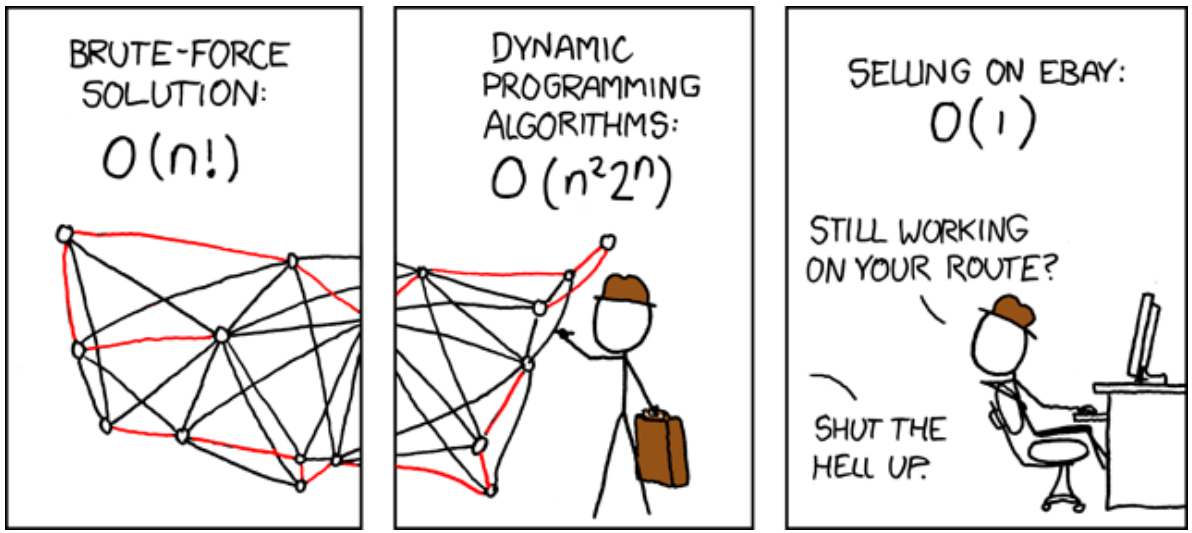
Why quantum computing?

This will only get us so far.



Why quantum computing?

Some problems will still remain intractable.



Sometimes that's a good thing:

- our cryptographic infrastructure relies on some mathematical problems being *computationally hard*

Image credit: https://imgs.xkcd.com/comics/travelling_salesman_problem.png

Why quantum computing?

But usually it just prevents us from doing interesting things:

- solving complex optimization problems
- simulation of molecules and quantum systems
- searching large spaces
- machine learning with large amounts of data

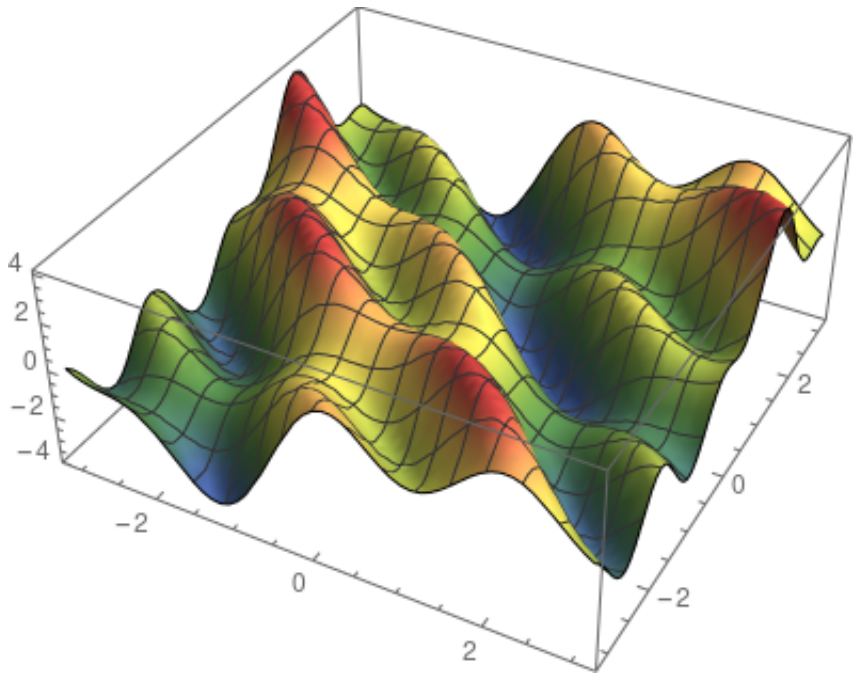


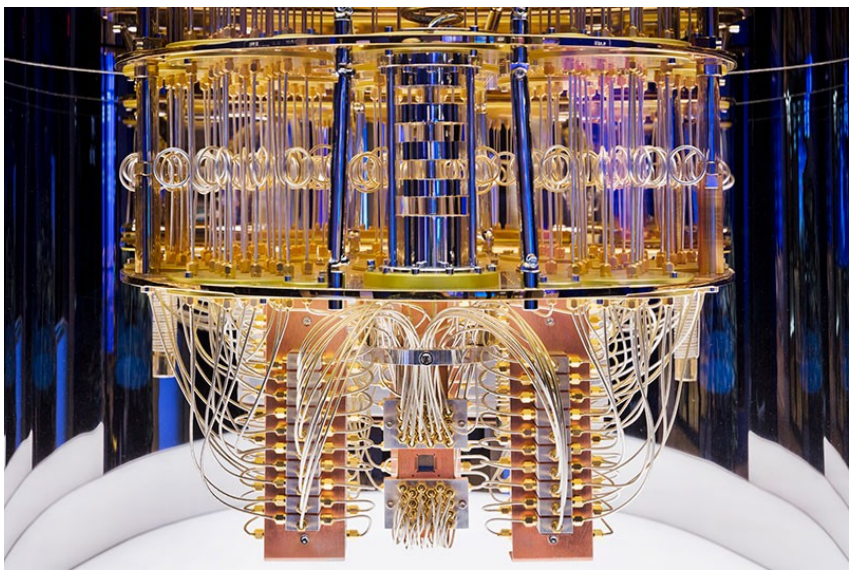
Image credit:

https://towardsdatascience.com/the-mathematics-of-optimization-for-deep-learning-11af2b1fda30

Quantum computing

"Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems."

– Nielsen & Chuang



There exist quantum algorithms that have an *exponential speedup* over regular computers (and some that don't, but are still good).

Image credit: https://www.nature.com/articles/d41586-021-03476-5

Quantum computing

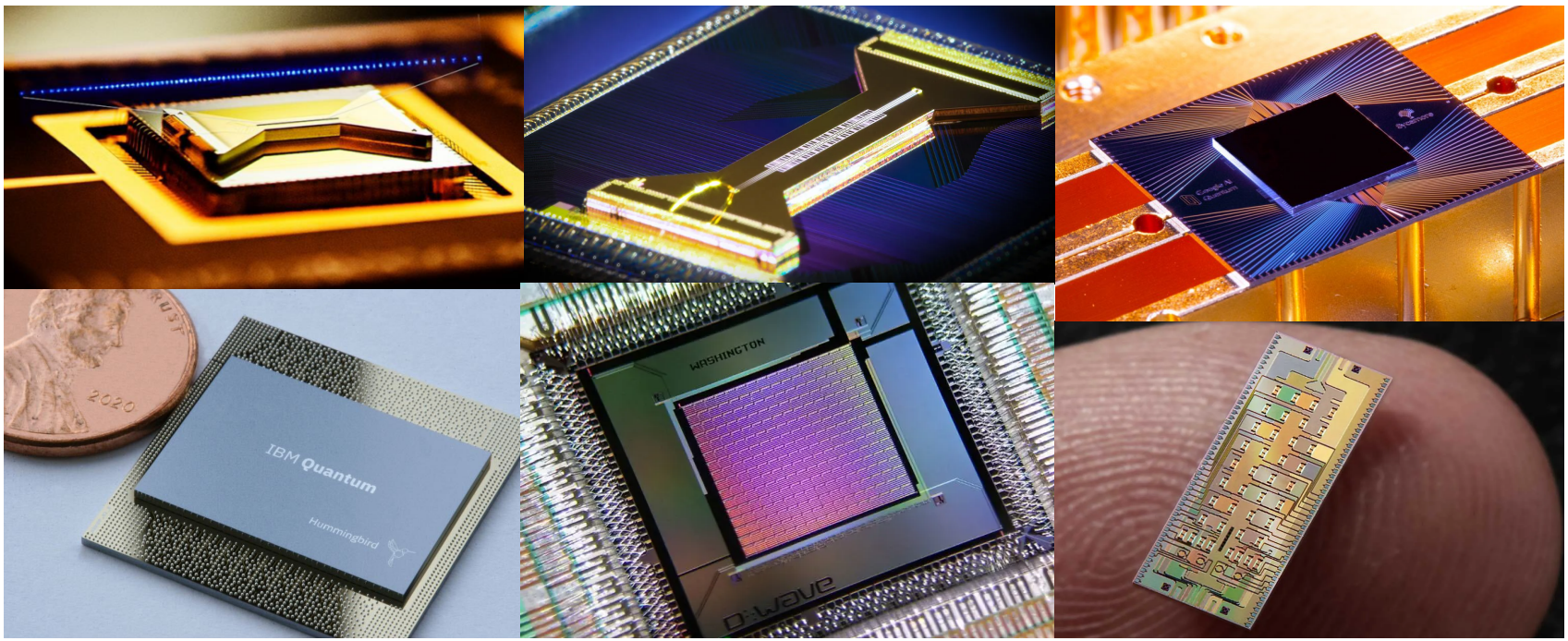
Lots of applications:

- Searching large spaces
- Discrete mathematics, (breaking) cryptography
- Simulating and calculating properties of physical systems
- Optimization
- Machine learning
- Things we haven't thought of yet!

We are already capable of doing small, proof-of-concept implementations of some of these algorithms...

Quantum computers are here

There are lots of different ones.



Commercial quantum processors (images clockwise from to left): IonQ (trapped-ion), Honeywell (trapped-ion), Google (superconducting), Xanadu (photonic), D-Wave (superconducting), IBM (superconducting).

Quantum computers are here

Largest device (IBM's "Osprey") has 433 superconducting qubits.

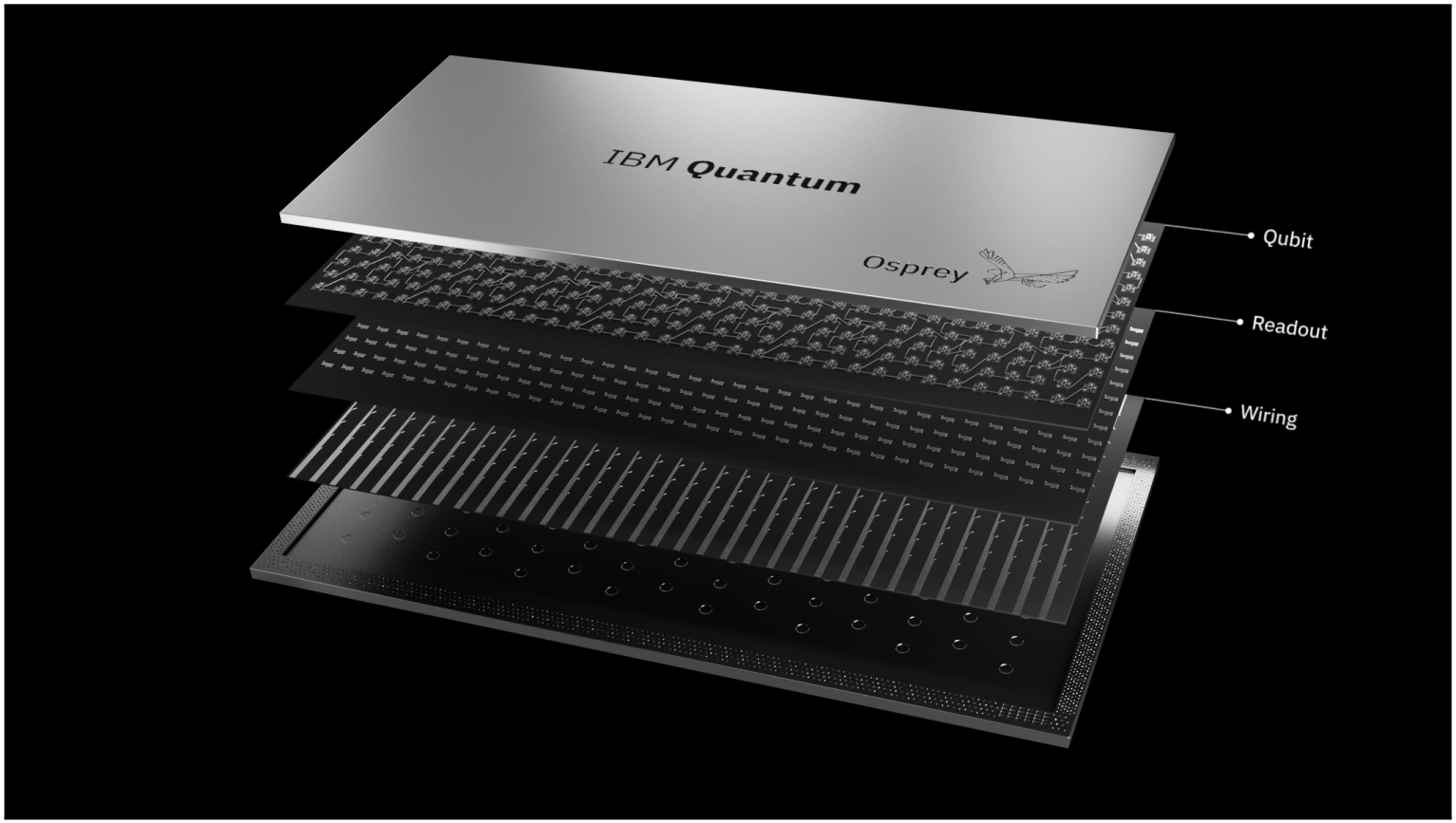


Image source: `https://newsroom.ibm.com/`

`2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two`

Quantum computers are here

Today's QCs are termed **noisy, intermediate scale quantum** (NISQ) devices.

Quantum Computing in the NISQ era and beyond

John Preskill

Institute for Quantum Information and Matter and Walter Burke Institute for Theoretical Physics, California Institute of Technology, Pasadena CA 91125, USA

Published: 2018-08-06, volume 2, page 79

Eprint: arXiv:1801.00862v3

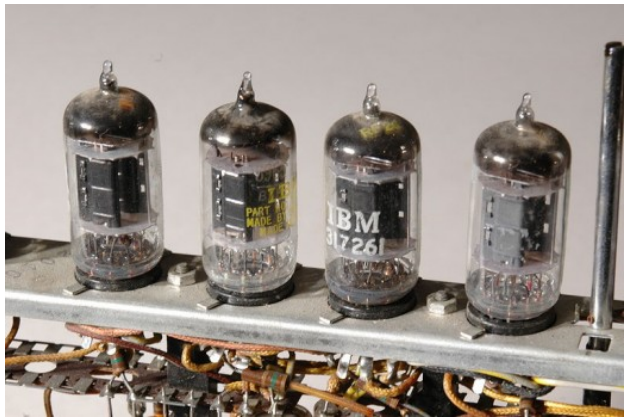
Doi: https://doi.org/10.22331/q-2018-08-06-79

Citation: Quantum 2, 79 (2018).

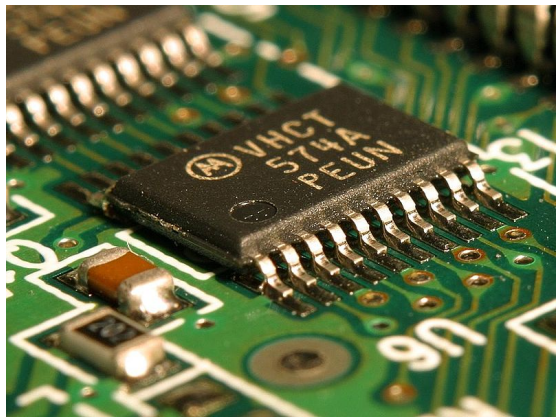
- Short coherence times
- High gate error rates (single-qubit, two-qubit operations)
- Limited qubit connectivity
- Challenging to scale up the hardware

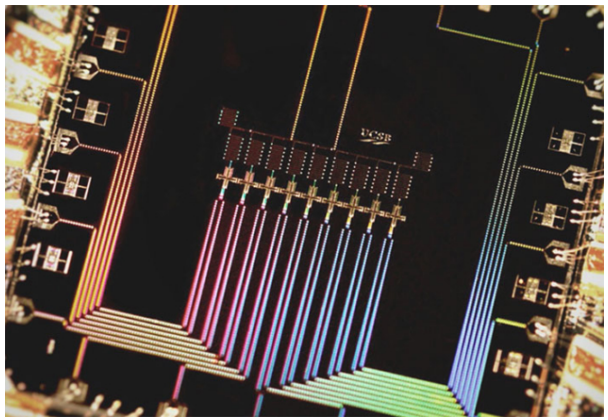
Quantum computers are here

Many competing technologies; still too early to know which (combination?) of them, if any, will win out long term.



vs.





vs.

???

Image credits:
https://hubpages.com/business/What-Is-a-Transistor-and-Why-is-it-Important
https://en.wikipedia.org/wiki/Solid-state_electronics
https://physicsworld.com/a/google-gains-new-ground-on-universal-quantum-computer/

Quantum computers are here

We can still use NISQ devices to do interesting things; but to solve problems "at scale" we need **fault-tolerant quantum computing**. We are starting to see error-corrected qubits in the wild...

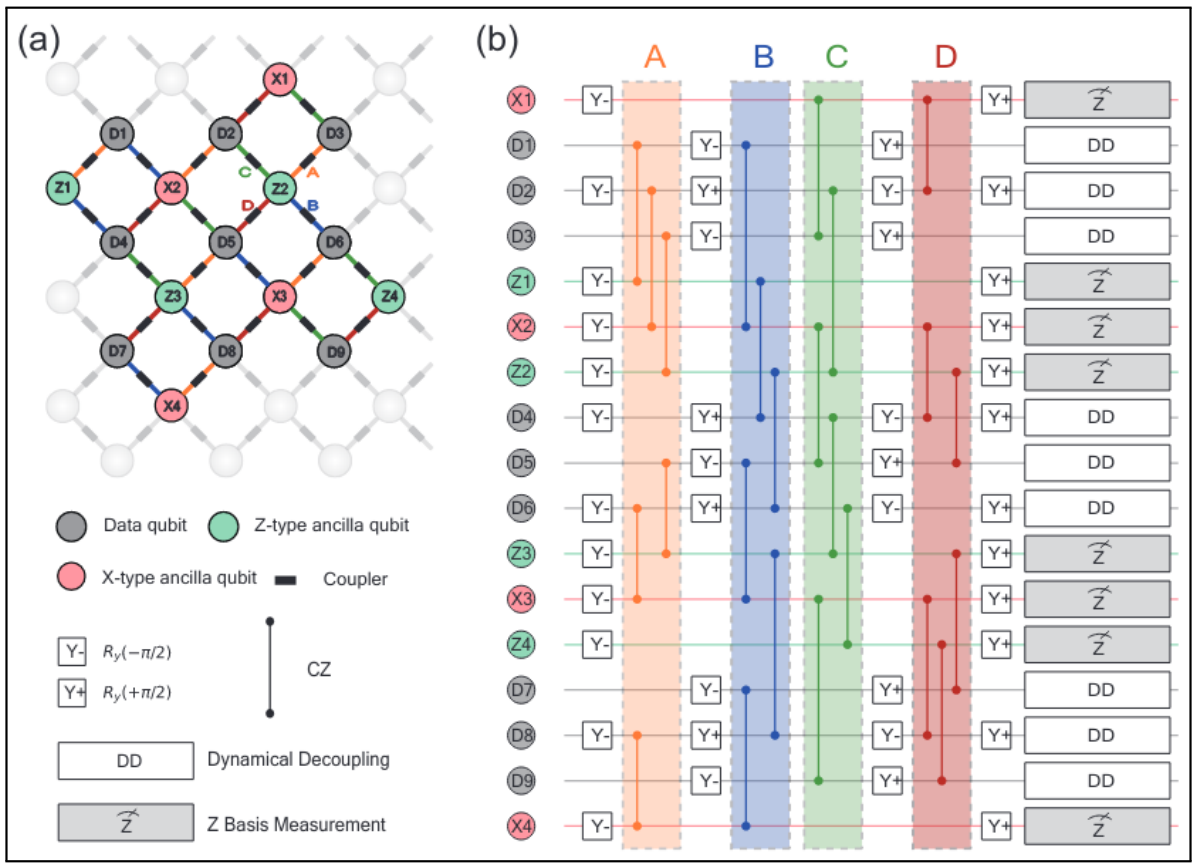


Image: 17-qubit surface code, Zhao et al. https://arxiv.org/abs/2112.13505v1

Quantum computers are here

Some applications will require *millions* of qubits to run an algorithm on noisy qubits with full error correction.

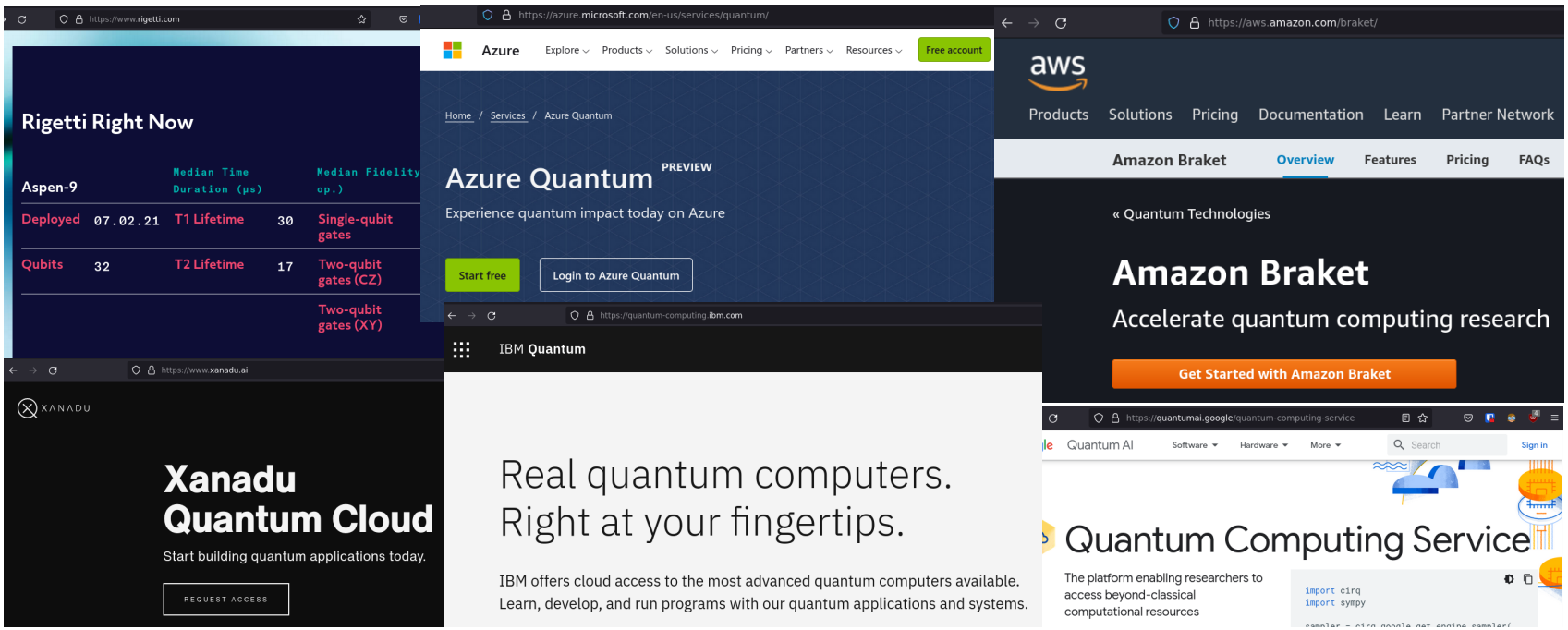
RSA-2048	**Old estimates**				**Current estimates**			
p_g	n_ℓ	n_p	quantum resources	time	n_ℓ	n_p	quantum resources	time
10^{-3}	6190	19.20	1.17	1.46	8194	22.27	0.27	0.34
10^{-5}	6190	9.66	0.34	0.84	8194	8.70	0.06	0.15

Table 2. RSA-2048 security estimates. Here n_ℓ denotes the number of logical qubits, n_p denotes the number of physical qubits (in millions), time denotes the expected time (in hours) to break the scheme, and *quantum resources* (*quantum resources*) are expressed in units of megaqubitdays. The corresponding classical security parameter is 112 bits.

Image: V. Gheorghiu and M. Mosca, *A resource estimation framework for quantum attacks against cryptographic functions - recent developments.* Feb. 15 2021.

Quantum computers are here

You can go use them *right now*!



But *how*?

Course learning outcomes

Core goal: **learn how to program quantum computers** in a hands-on, software-focused setting.

- Describe the societal importance and implications of quantum computing
- Explain the theory and principles behind gate-model quantum computing
- Describe the operation of key quantum algorithms
- Implement basic and research-level quantum algorithms using Python and PennyLane

In this course you will implement everything you learn!

Covered in this course

- qubit states, operations, and measurements
- quantum circuits
- basic quantum algorithms
- quantum compilation
- Grover's algorithm
- Shor's algorithm
- variational quantum algorithms
- Hamiltonian simulation
- considerations for running on near-term hardware

Not covered in this course

- How the hardware actually works
- Non-gate-model quantum computing
  - adiabatic quantum computing
  - quantum annealing
  - measurement-based quantum computing
  - continuous-variable quantum computing
- Advanced theoretical topics (quantum information, complexity theory, quantum error correction)

Logistics

	Olivia Di Matteo	Jose Pinilla (TA)
Office:	KAIS 3043	KAIS 4025
Office hrs:	M 13:00-14:00 or by appointment. Open-door policy after 12:00 weekdays.	Th 11:00-12:00
Email:	`olivia@ece.ubc.ca`	`jpinilla@ece.ubc.ca`
GitHub:	`glassnotes`	`joseppinilla`

All lecture slides/demos will be posted on the course GitHub:

`https://github.com/glassnotes/CPEN-400Q`

(Find last year's materials under "Releases" section)

Textbook

Primary resource will be the **free**, **online** Xanadu Quantum Codebook. Use to learn about the content, PennyLane, and do practice programming exercises to supplement class work.

XANADU QUANTUM CODEBOOK

codebook.xanadu.ai

Secondary (optional) resource: *Introduction to Quantum Computation and Quantum Information*, Nielsen and Chuang.

Assignments and grading

Three components:

- 30% technical assignments
- 10% "quantum literacy" assignments
- 10% in-class quizzes
- 50% final group project

No exams!

Assignments and grading, cntd.

30% technical assignments (5-6):

- Done on PrairieLearn
- Implement algorithms and solve quantum computing programming problems in PennyLane
- May discuss with classmates but what you submit must be your own work (and cite your collaborators!)

You may submit one assignment $<$24hrs late, with advance notice.

Assignment 0 will be available today. It is a review of linear algebra, Python, NumPy, and will introduce you to PrairieLearn.

Assignments and grading, cntd.

10% quantum literacy assignments (5-6):

- Distributed on PrairieLearn
- Done individually, no collaboration unless otherwise stated
- Activities may include:
  - critical analysis of media articles or videos
  - small creative writing projects
  - group discussions or debates

Purpose is to explore the ethical and societal implications around construction and use of quantum computers, and help dispel hype.

Assignments and grading, cntd.

10% quizzes (10):

- Starting next Monday; done at start of class in PrairieLearn
- Done individually, but open book/notes/docs
- Includes math and/or programming problems and conceptual questions; based on previous week's lecture material

Purpose is for reviewing content.

Assignments and grading, cntd.

50% final project:

- Implement the methods / algorithms in a research paper and reproduce the results.
- Work in groups of 4
- Three components:
  - $\sim$10 minute in-class lecture and demonstration
  - Fully-documented software implementation
  - Companion report
- Can use a different quantum programming framework if you like (Qiskit, Q#, Quantum++, Cirq, Yao, etc.)

More details (rubric, suggested papers, etc.) to follow.

Today

Learning outcomes:

1. Define quantum computing, and explain its societal importance
2. Define a *qubit* both conceptually and mathematically
3. Simulate quantum computing on a single-qubit

Bits

All computation in our computers today is done with bits.

The state of a bit is *either* 0 or 1.

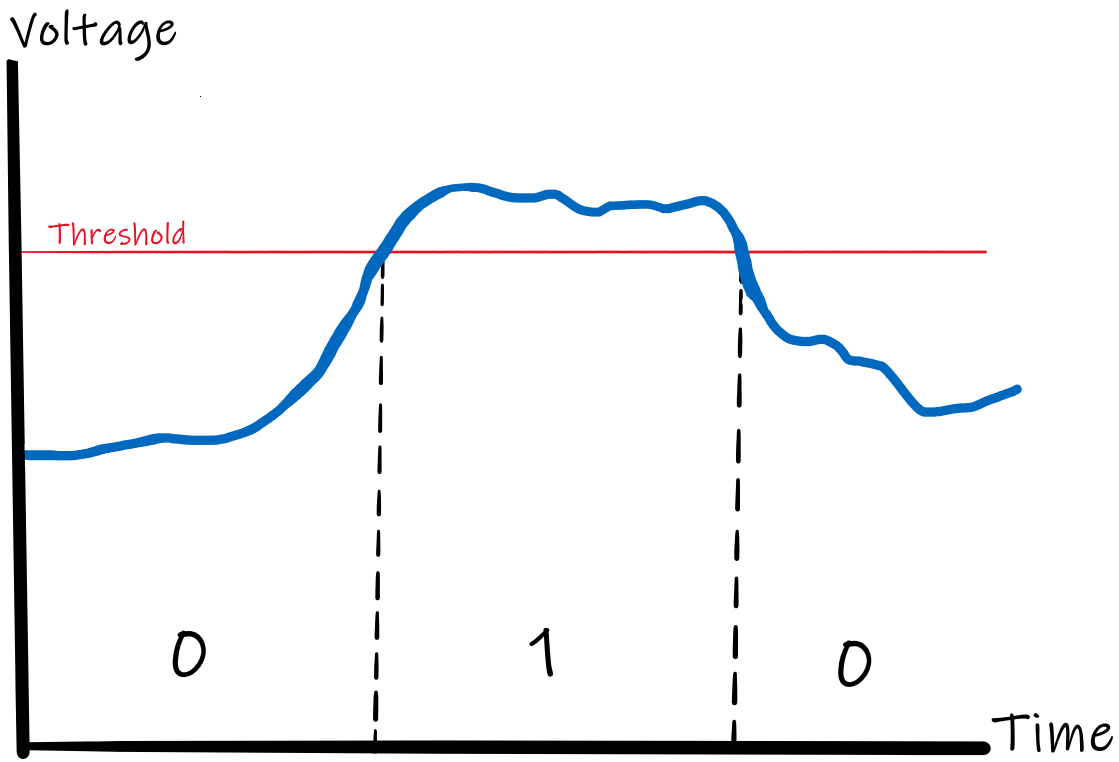
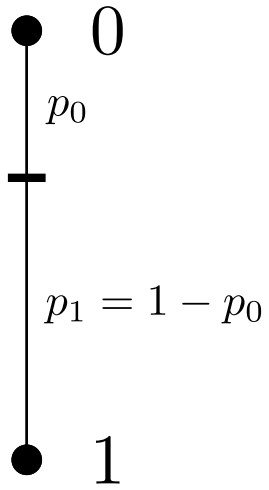


Image credit: Xanadu Quantum Codebook I.1

Probabilistic bits

The value of a bit can be represented by a probability distribution.



Example: A coin flip results in Heads (1) or Tails (0).

Example: Is it raining? Yes (1) or No (0)

Probabilistic bits

These probability distributions can evolve over time.

Exercise:

- If rain today, 70% probability it will rain tomorrow.
- If no rain today, 20% probability it will rain tomorrow.

Suppose it is raining today. What is the % chance it will rain two days from now?

Probabilistic bits

Represent the system as a 2-dimensional real-valued **vector** ($\mathbb{R}^2$)

$$\text{Today's weather} = \begin{pmatrix} \text{Prob. rain} \\ \text{Prob. no rain} \end{pmatrix}$$

Elements necessarily sum to 1. More compactly:

$$\begin{aligned} \vec{w} &= \begin{pmatrix} p_r \\ 1 - p_r \end{pmatrix} \\ &= p_r \begin{pmatrix} 1 \\ 0 \end{pmatrix} + (1 - p_r) \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= p_r \vec{r} + (1 - p_r) \vec{n} \end{aligned}$$

We call $\vec{r}$ and $\vec{n}$ basis vectors.

Probabilistic bits

- Rain today, 70% probability it will rain tomorrow.
- No rain today, 20% probability it will rain tomorrow.

How does this affect our basis vectors?

$$\text{(raining)} \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \to \quad \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} \qquad \vec{r} \to 0.7\vec{r} + 0.3\vec{n}$$

$$\text{(not raining)} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \to \quad \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix} \qquad \vec{n} \to 0.2\vec{r} + 0.8\vec{n}$$

More compactly,

$$\vec{w} \to P\vec{w} = \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \vec{w}$$

Probabilistic bits

To solve our problem, check what happens tomorrow:

$$\vec{t} = P\vec{r} = \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix}$$

Then the day after:

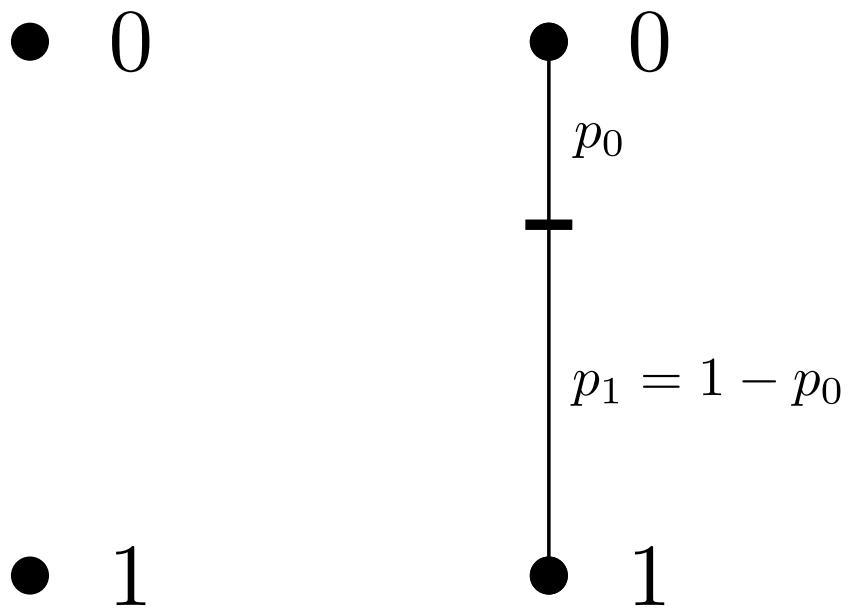
$$\vec{a} = P\vec{t} = \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.55 \\ 0.45 \end{pmatrix}$$

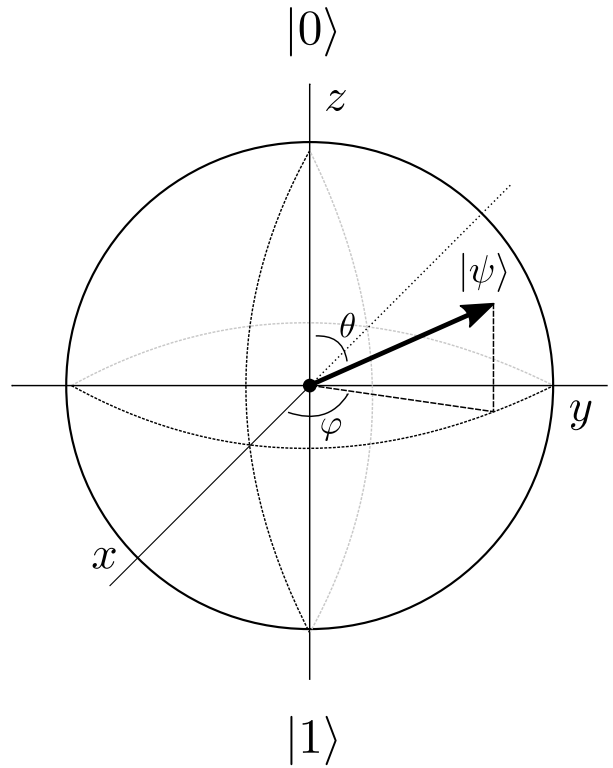
Note that these operations are linear, and we can solve without explicit matrix multiplication:

$$\begin{aligned} \vec{a} &= P\vec{t} = P(0.7\vec{r} + 0.3\vec{n}) \\ &= 0.7P\vec{r} + 0.3P\vec{n} \\ &= 0.7(0.7\vec{r} + 0.3\vec{n}) + 0.3(0.2\vec{r} + 0.8\vec{n}) \\ &= 0.55\vec{r} + 0.45\vec{n} \end{aligned}$$

Qubits

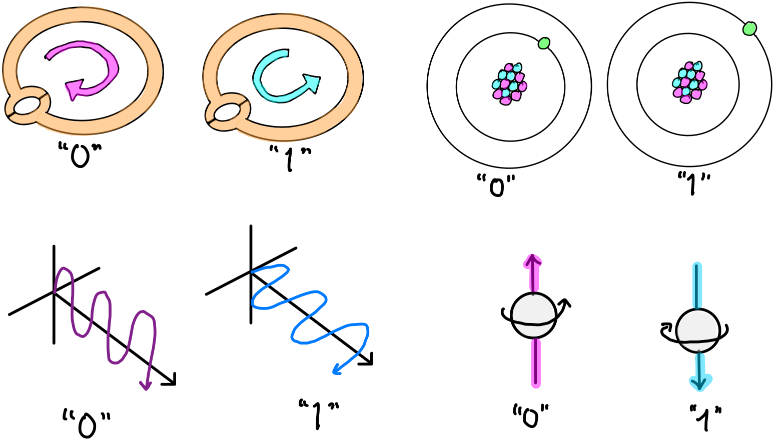
Extension of probabilistic bits to 2-level quantum systems:





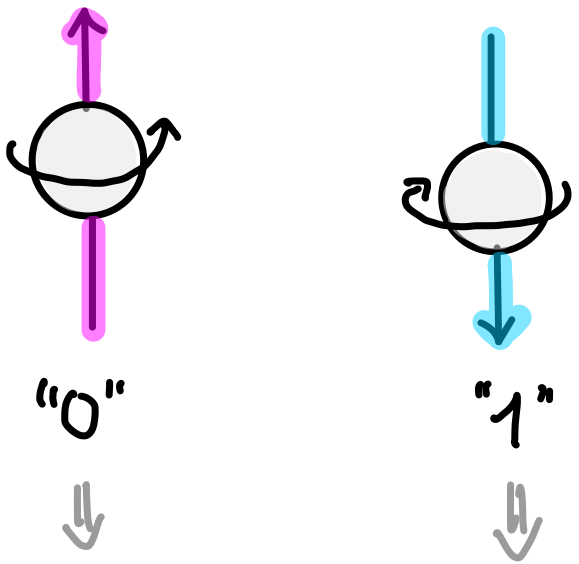
Qubits

Quantum computers work by manipulating **qubits**.



Mathematical representation of qubits

The vector space where qubits live is called **Hilbert space**.

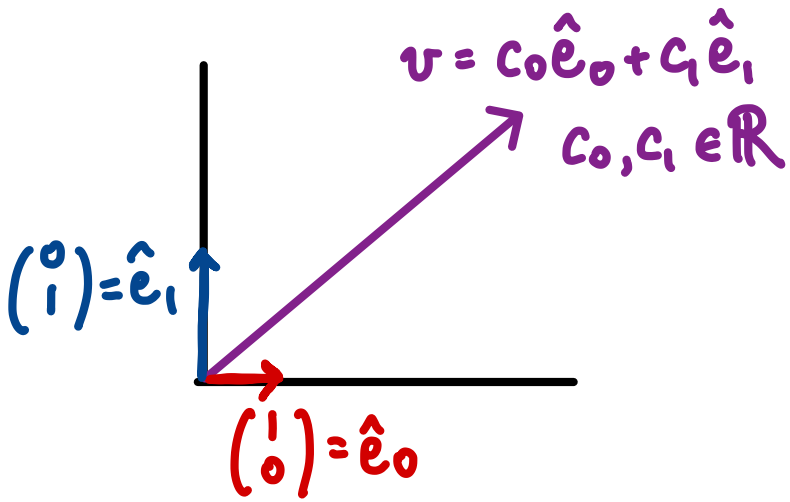


These two vectors together form the **computational basis**.

The notation $|\cdot\rangle$ is called Dirac, or **bra-ket notation**. $|\cdot\rangle$ is a ket.

Mathematical representation of qubits

Qubit states are a linear combination of these basis states; coefficients can be *complex*.



$\Rightarrow$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$
$$\alpha, \beta \in \mathbb{C}$$

The coefficients α and β are called **amplitudes**. "Valid" qubit state vectors have unit length:

$$|\alpha|^2 + |\beta|^2 = \alpha\alpha^* + \beta\beta^* = 1, \quad \alpha, \beta \in \mathbb{C}.$$

Such states are called **normalized**.

Exercise: is $|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}e^{0.2i}|1\rangle$ a valid quantum state?
Solution: check that $|\alpha|^2 + |\beta|^2 = 1$.

$$\alpha = \frac{1}{\sqrt{3}} \rightarrow \qquad |\alpha|^2 = \alpha\alpha^* = \frac{1}{3}$$

$$\beta = \sqrt{\frac{2}{3}}e^{0.2i} \rightarrow \qquad |\beta|^2 = \beta\beta^* = \sqrt{\frac{2}{3}}e^{0.2i}\sqrt{\frac{2}{3}}e^{-0.2i} = \frac{2}{3}$$

Thus, $|\alpha|^2 + |\beta|^2 = \frac{1}{3} + \frac{2}{3} = 1$. The state is properly normalized and is a valid quantum state.

Mathematical representation of qubits

A qubit in a linear combination

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

is in a **superposition** of the two basis states.

Superposition is one of the unique features that makes quantum computing very powerful.

Note: a qubit in superposition is not *"in both states at the same time"*!

Using qubits for computation

1. **Prepare** qubits in a **superposition**
2. Apply **operations** that **entangle** the qubits and manipulate the amplitudes
3. **Measure** qubits to extract an answer
4. Profit

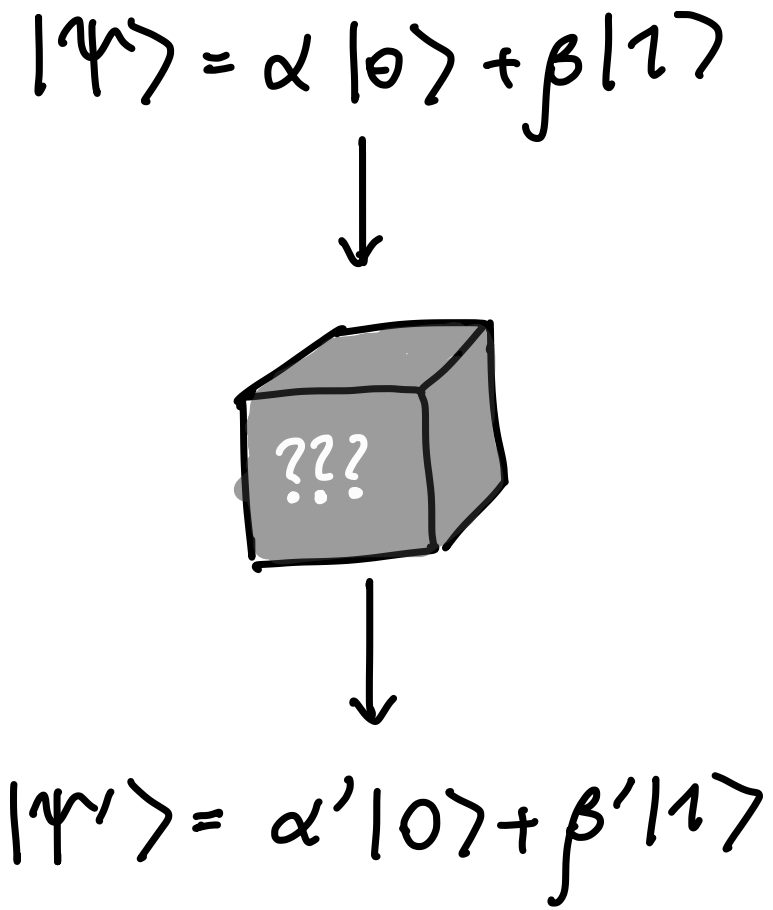
Using qubits for computation

1. **Prepare** qubits in a **superposition**
2. Apply **operations** that **entangle** the qubits and manipulate the amplitudes
3. **Measure** qubits to extract an answer
4. Profit

...how do we do this?

Manipulating qubits

Manipulating a qubit changes its state:



We need a mechanism for sending valid quantum state vectors to other valid quantum state vectors.

Manipulating qubits: unitary operations

Single-qubit states are manipulated by 2×2 unitary matrices. A matrix U is unitary if

$$U^\dagger U = U U^\dagger = \mathbb{1}.$$

The $\dagger$ means "conjugate transpose":

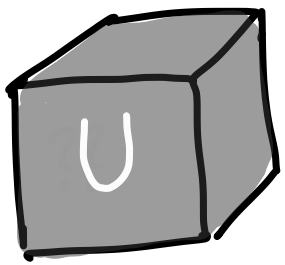
$$U^\dagger = (U^*)^T.$$

Unitary operations, or **gates**, are **reversible**: if we apply U, we can undo it by applying $U^\dagger$.

Manipulating qubits: unitary operations

Unitary operations preserve the length of vectors, i.e., *maintain the normalization of qubit states*.

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ $\qquad |\alpha|^2 + |\beta|^2 = 1$



$UU^\dagger = U^\dagger U = \mathbb{1}$

$|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$ $\qquad |\alpha'|^2 + |\beta'|^2 = 1$

Manipulating qubits: unitary operations

Unitary operations act *linearly* on superpositions:

$$U(\alpha|0\rangle + \beta|1\rangle) = \alpha U|0\rangle + \beta U|1\rangle$$

This is a key contributor to the power of quantum computing!

Example: X

The matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

is unitary.

Exercise: Determine what it does by evaluating its action on the basis states.

$$\begin{aligned} X|0\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle, \\ X|1\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle. \end{aligned}$$

Example: X

Exercise: What does X do to the state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$\begin{aligned} X\left(\alpha|0\rangle + \beta|1\rangle\right) &= \alpha \cdot X|0\rangle + \beta \cdot X|1\rangle \\ &= \alpha|1\rangle + \beta|0\rangle. \end{aligned}$$

X is also known as Pauli X, the bit flip, or NOT gate. It is *self-inverse*.

Example: H

Perhaps the most important unitary in quantum computing is the Hadamard gate (H):

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

This gate creates a *uniform superposition*:

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) = |+\rangle \\ H|1\rangle &= \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = |-\rangle \end{aligned}$$

Example: H

The Hadamard is also self-inverse:

$$\begin{aligned} H|+\rangle &= H \cdot \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\left(H|0\rangle + H|1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) + \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\right) \\ &= |0\rangle \end{aligned}$$

Similarly, $H|-\rangle = |1\rangle$.

Example: Z

The gate

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

does something a little different.

Apply to basis states:

$$\begin{aligned} Z|0\rangle &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle, \\ Z|1\rangle &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = - \begin{pmatrix} 0 \\ 1 \end{pmatrix} = -|1\rangle. \end{aligned}$$

Just changes the sign. (This will be important later!)

Single-qubit gates: applying sequences of gates

We apply *products* of single-qubit operations to represent performing gates in sequence.

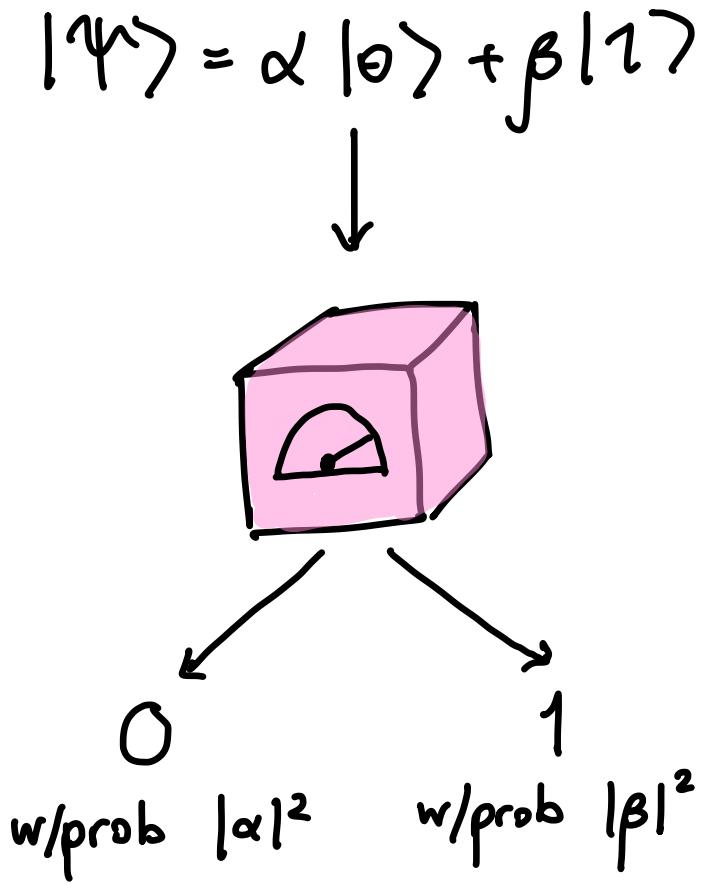
Exercise: Apply H, then Z, then X to $|0\rangle$ without doing any matrix multiplication.

$$\begin{aligned} XZH|0\rangle &= XZ\left(\frac{|0\rangle}{\sqrt{2}} + \frac{|1\rangle}{\sqrt{2}}\right) \\ &= X\left(\frac{|0\rangle}{\sqrt{2}} - \frac{|1\rangle}{\sqrt{2}}\right) \\ &= \frac{|1\rangle}{\sqrt{2}} - \frac{|0\rangle}{\sqrt{2}} \end{aligned}$$

Measuring qubits

Manipulating qubits changes their amplitudes. These determine probability of observing the qubit in $|0\rangle$ or $|1\rangle$ when we measure (it's **probabilistic**!).

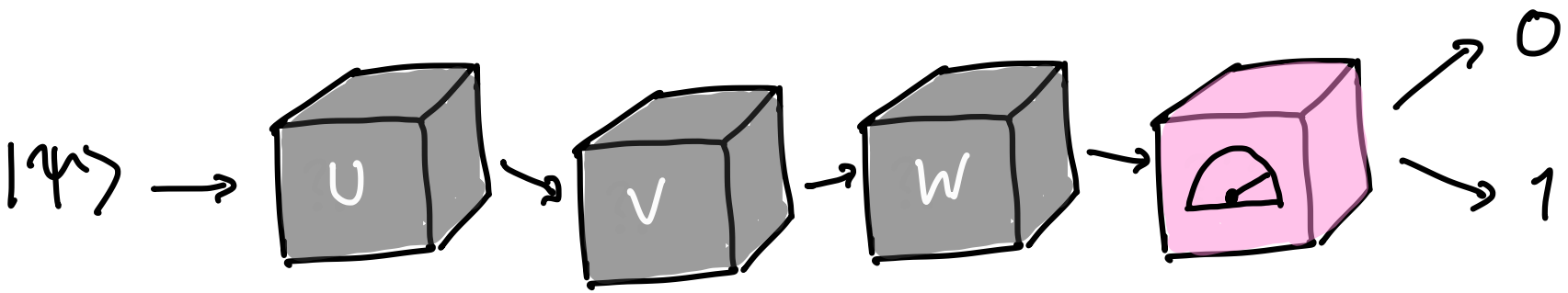
Running an algorithm and measuring gives us a single bit of information (0 or 1). To get more, we need to run the algorithm multiple times (multiple **shots**).



Quantum computing

Quantum computing is the act of manipulating the state of qubits in a way that represents solution of a computational problem:

1. **Prepare** qubits in a **superposition**
2. Apply **operations** that **entangle** the qubits and manipulate the amplitudes
3. **Measure** qubits to extract an answer
4. Profit



Let's simulate this using NumPy.

Programming quantum computers

That was tedious—let's never do that again.

Let's use some real quantum software instead: there is a fantastic ecosystem of open-source tools.


```python
import pennylane as qml

H = qml.Hamiltonian(...)

dev = qml.device('default.qubit', wires=2)

@qml.qnode(dev)
def quantum_circuit(params):
    qml.RY(params[0], wires=0)
    qml.RY(params[1], wires=1)
    qml.CNOT(wires=[0, 1])
    qml.RY(params[2], wires=0)
    return qml.expval(H)

quantum_circuit([0.1, 0.2, 0.3])
```

PennyLane

We will use PennyLane, a Python framework developed by **Xanadu** (a Toronto-based quantum startup).

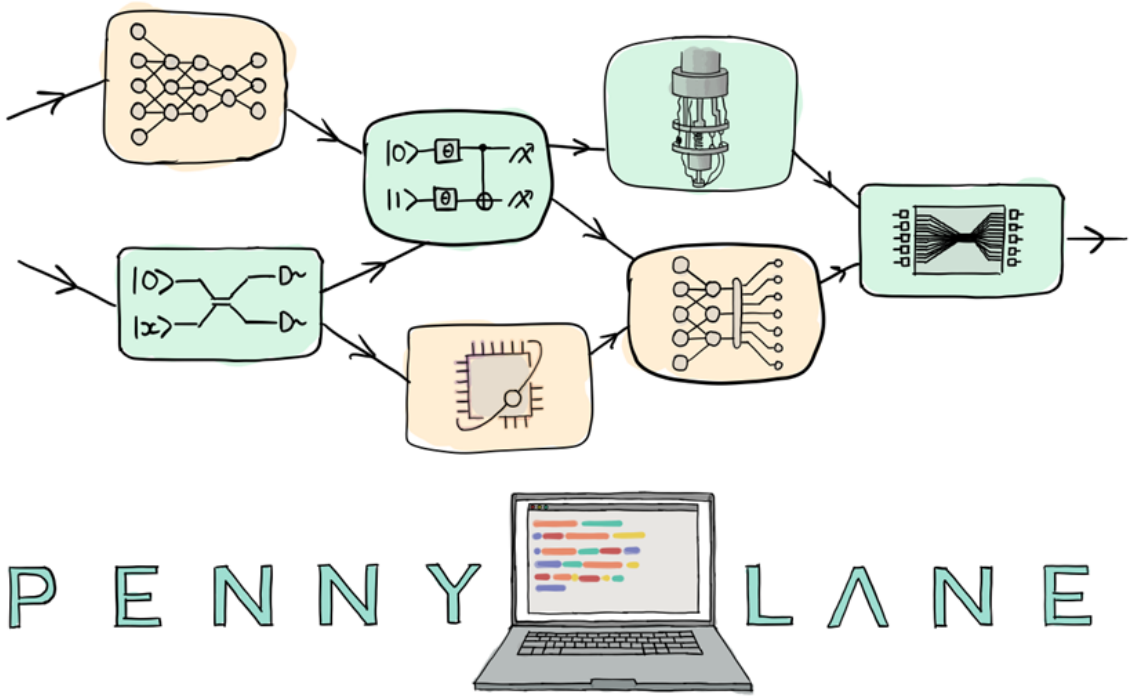


Image credit: `https://pennylane.ai/`

PennyLane

GitHub: `https://github.com/PennyLaneAI/PennyLane`
Documentation: `https://pennylane.readthedocs.io/en/stable/`
Demonstrations: `https://pennylane.ai/qml/demonstrations.html`
Discussion Forum: `https://discuss.pennylane.ai/`

Its key use case is **differentiable quantum programming** and quantum machine learning; it is also a valuable tool for quantum computing algorithms and applications.

PennyLane

```python
def ket_0():
    return np.array([1.0, 0.0])

def apply_ops(ops, state):
    for op in ops:
        state = np.dot(op, state)
    return state

def measure(state, num_samples=100):
    prob_0 = state[0] * state[0].conj()
    prob_1 = state[1] * state[1].conj()

    samples = np.random.choice(
        [0, 1], size=num_samples, p=[prob_0, prob_1]
    )
    return samples

H = (1/np.sqrt(2)) * np.array([[1, 1], [1, -1]])
X = np.array([[0, 1], [1, 0]])
Z = np.array([[1, 0], [0, -1]])

input_state = ket_0()
output_state = apply_ops([H, X, Z], input_state)
results = measure(output_state, num_samples=10)

print(results)
```

```
[1 0 0 1 0 0 0 1 1 0]
```

Sample NumPy

```python
dev = qml.device('default.qubit', wires=1, shots=10)

@qml.qnode(dev)
def my_circuit():
    qml.Hadamard(wires=0)
    qml.PauliX(wires=0)
    qml.PauliZ(wires=0)
    return qml.sample()

print(my_circuit())
```

```
[1 0 1 0 1 1 0 1 0 0]
```

PennyLane

Recap

Today's learning outcomes were:

1. Define quantum computing, and explain its societal importance
2. Define a *qubit* both conceptually and mathematically
3. Simulate quantum computing on a single-qubit

For next time

Content:

- Getting started with PennyLane
- Quantum circuits
- More unitary operations

Action items:

1. Sign up for Xanadu Quantum Cloud, or install and configure your programming environment with PennyLane v0.28 prior to next class (come to office hours if you need help)
2. Follow instructions on Canvas to sign up for PrairieLearn and Piazza
3. Work on Assignment 0

Recommended reading: Codebook nodes I.1-I.8.