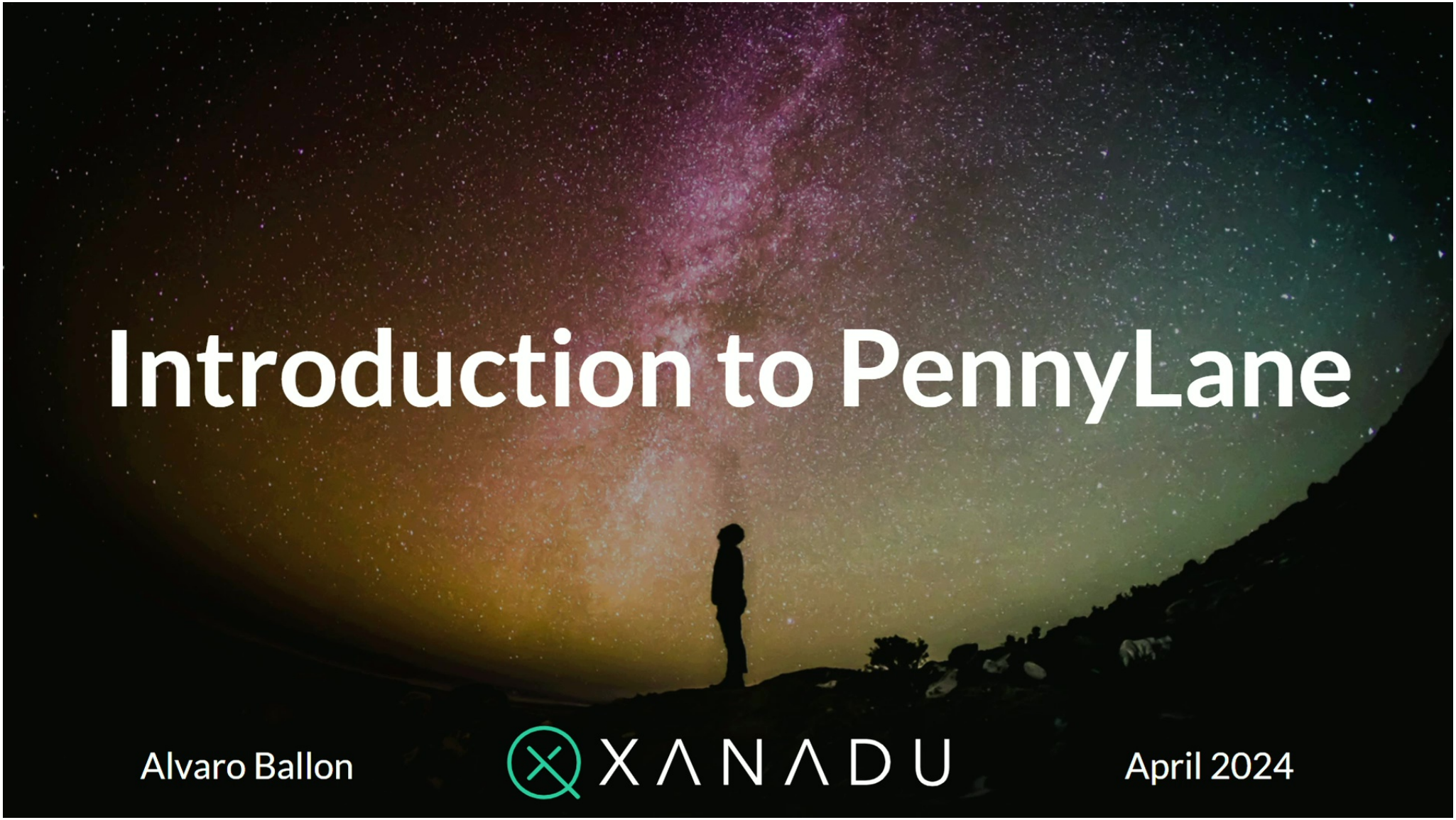
Title: Quantum Machine Learning

Speakers: Alvaro Ballon Bordo

Collection: Navigating Quantum and AI Career Trajectories: A Beginnerâ€™s Mini-Course on Computational Methods and their Applications

Date: May 21, 2024 - 9:30 AM

URL: https://pirsa.org/24050082

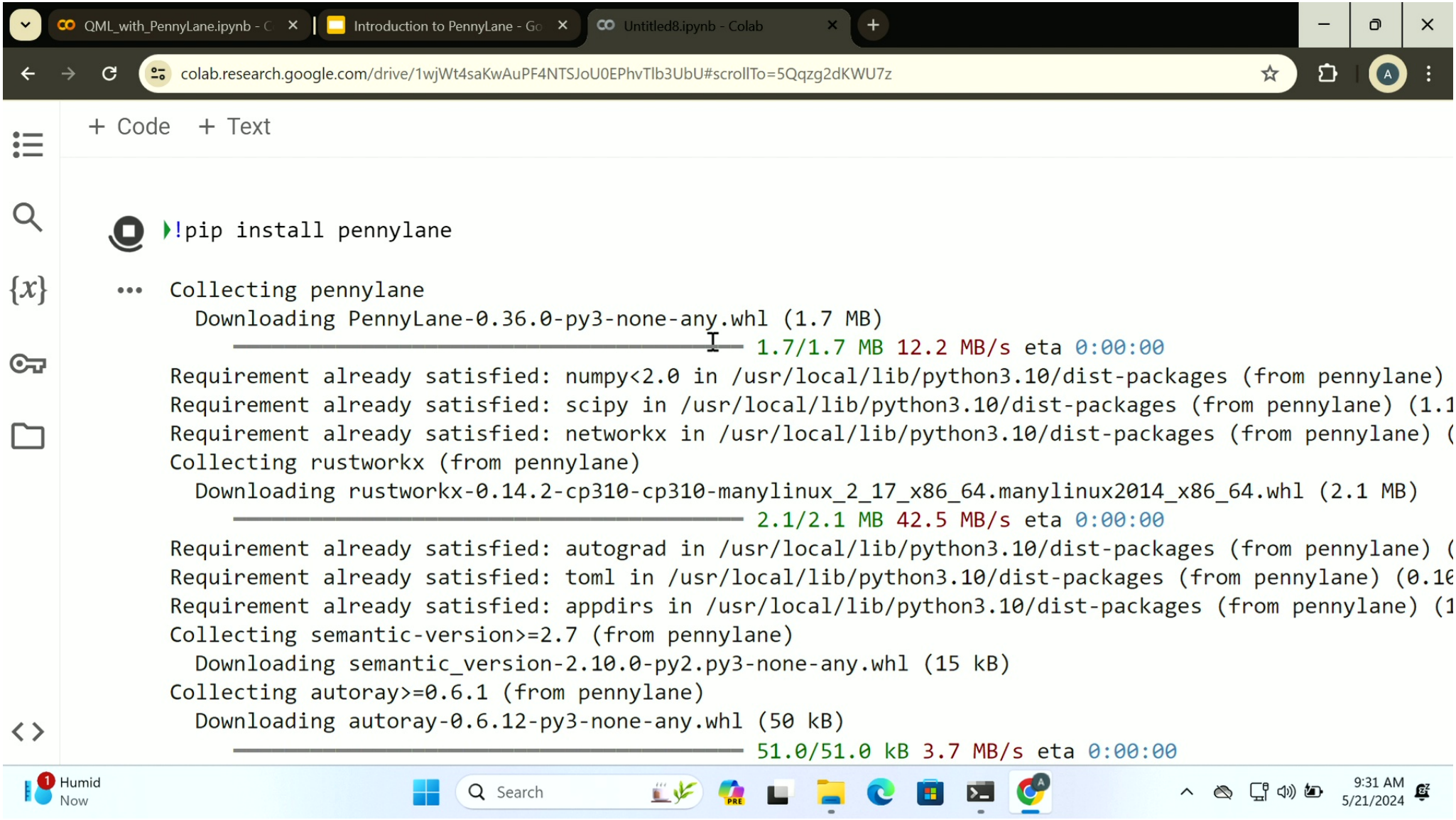

Introduction to PennyLane
Alvaro Ballon
XANADU
April 2024

Vision

Emphasis on fundamental **building blocks** to provide **flexibility** to users.

We target **user-friendliness** and **educational material**.

The library makes full use of native support for **differentiable quantum computations**


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=5Qqzg2dKWU7z
+ Code
+ Text
!pip install pennylane
Collecting pennylane
Downloading PennyLane-0.36.0-py3-none-any.whl (1.7 MB)
1.7/1.7 MB 12.2 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0 in /usr/local/lib/python3.10/dist-packages (from pennylane)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from pennylane) (1.1
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from pennylane) (
Collecting rustworkx (from pennylane)
Downloading rustworkx-0.14.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
2.1/2.1 MB 42.5 MB/s eta 0:00:00
Requirement already satisfied: autograd in /usr/local/lib/python3.10/dist-packages (from pennylane) (
Requirement already satisfied: toml in /usr/local/lib/python3.10/dist-packages (from pennylane) (0.10
Requirement already satisfied: appdirs in /usr/local/lib/python3.10/dist-packages (from pennylane) (1
Collecting semantic-version>=2.7 (from pennylane)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Collecting autoray>=0.6.1 (from pennylane)
Downloading autoray-0.6.12-py3-none-any.whl (50 kB)
51.0/51.0 kB 3.7 MB/s eta 0:00:00
Humid
Now
Search
9:31 AM
5/21/2024

Quantum devices

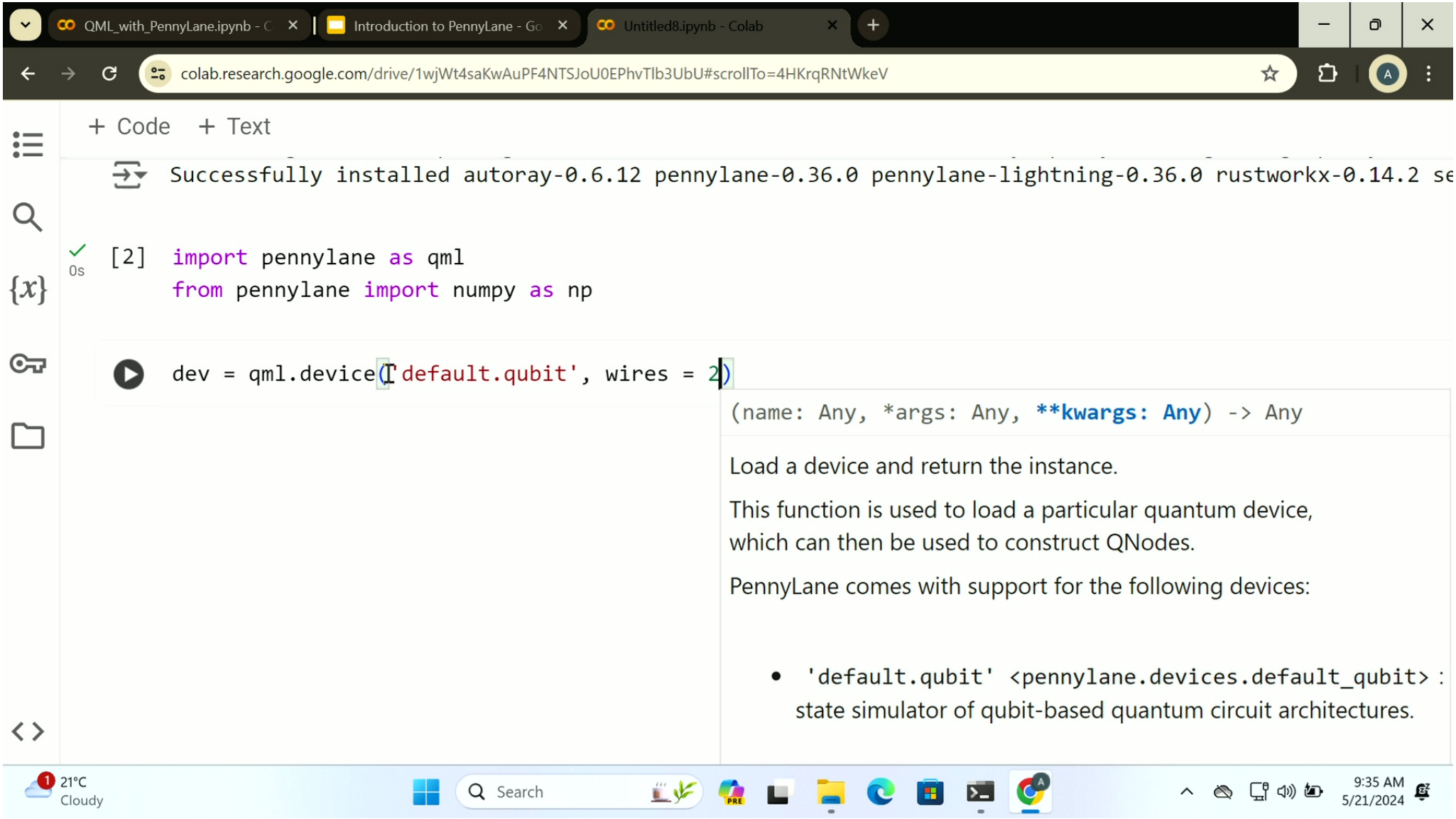
To simulate quantum circuits, we must specify a device

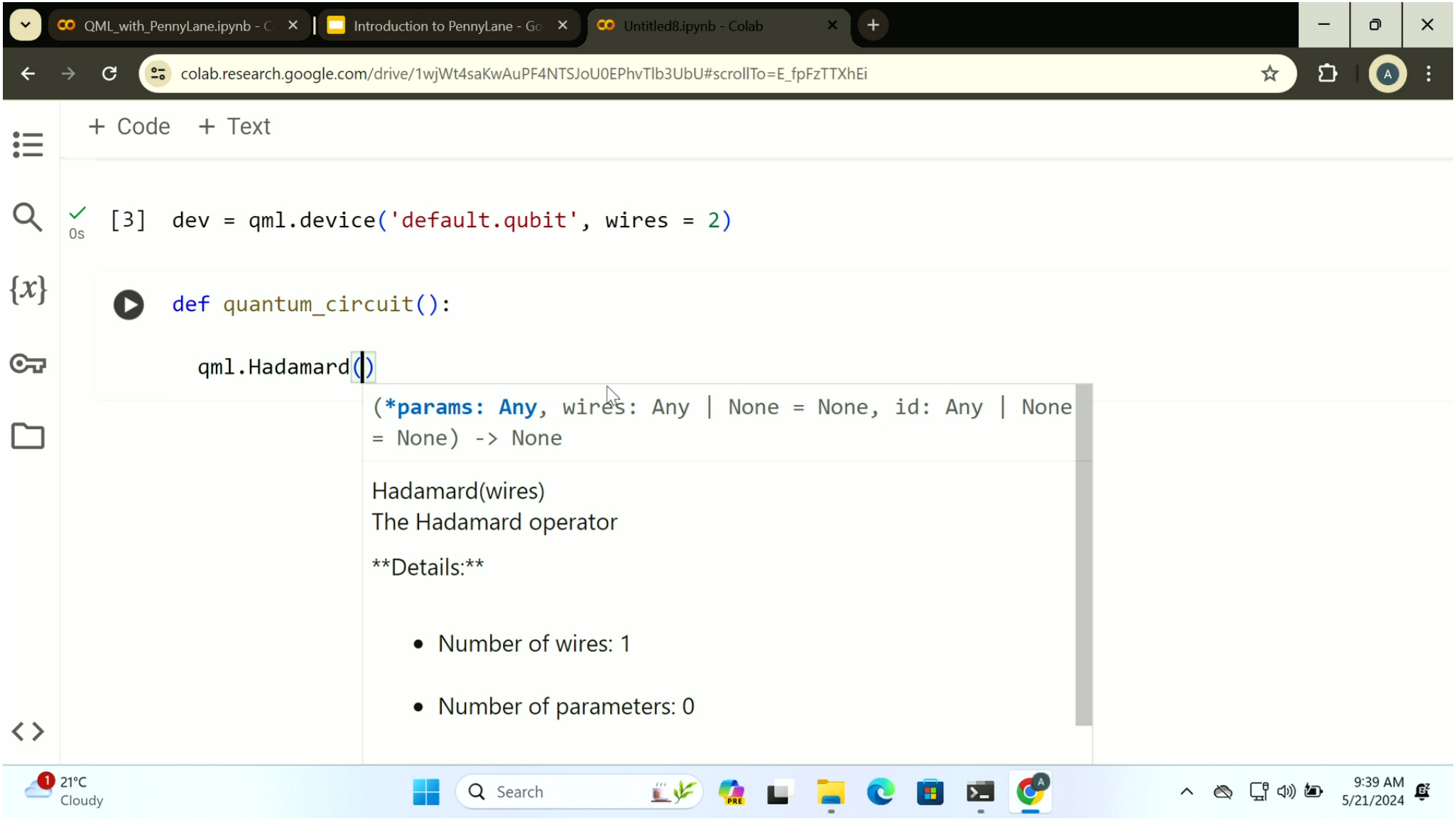
- default.qubit: Run-of-the-mill device that works with pure qubit states.
- default.mixed: Device that works with mixed states.
- lightning.qubit: Faster simulator, with a backend coded in C++ for efficiency.
- default.gaussian: For continuous variable systems

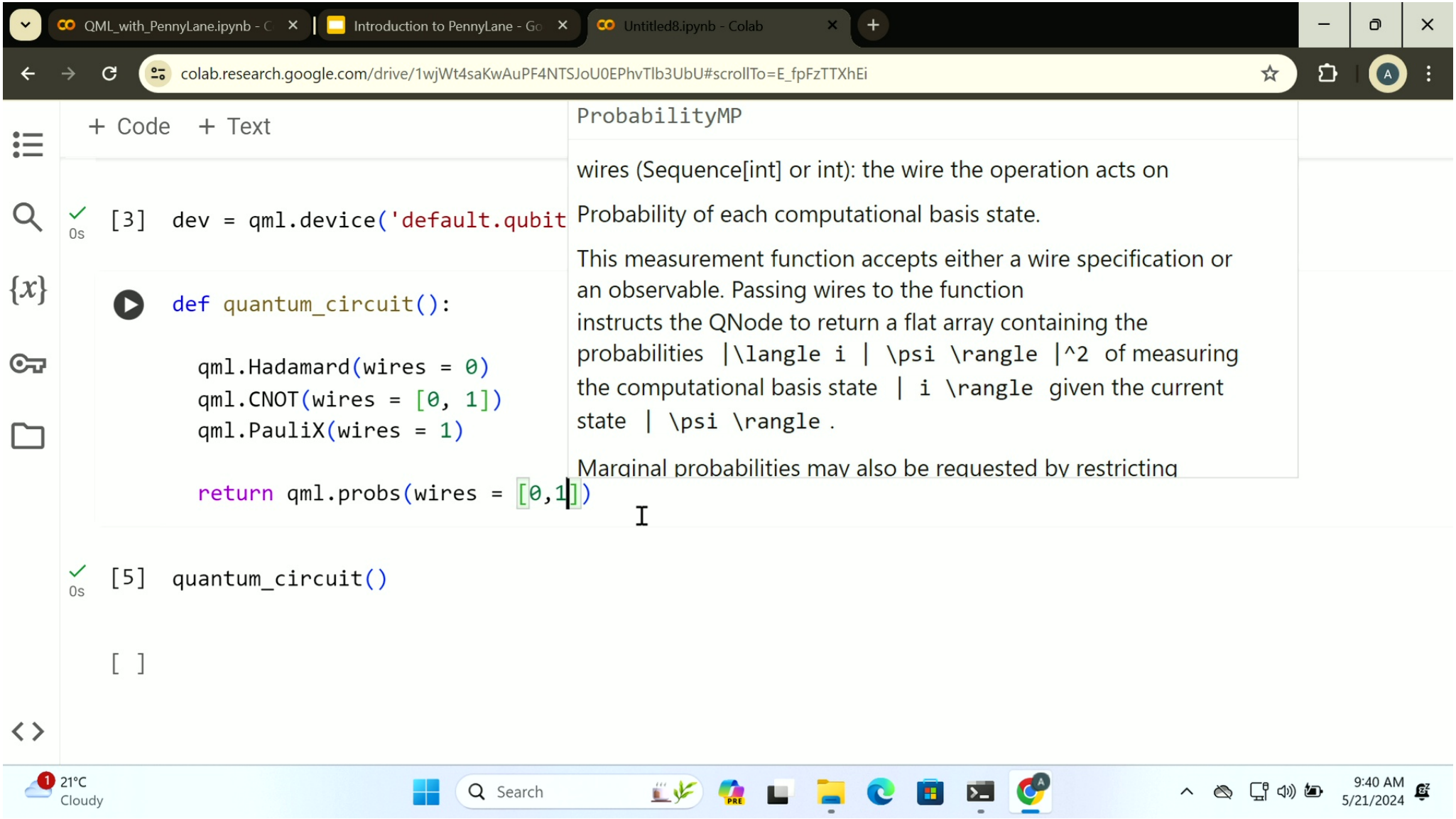
We define a device in PennyLane as follows

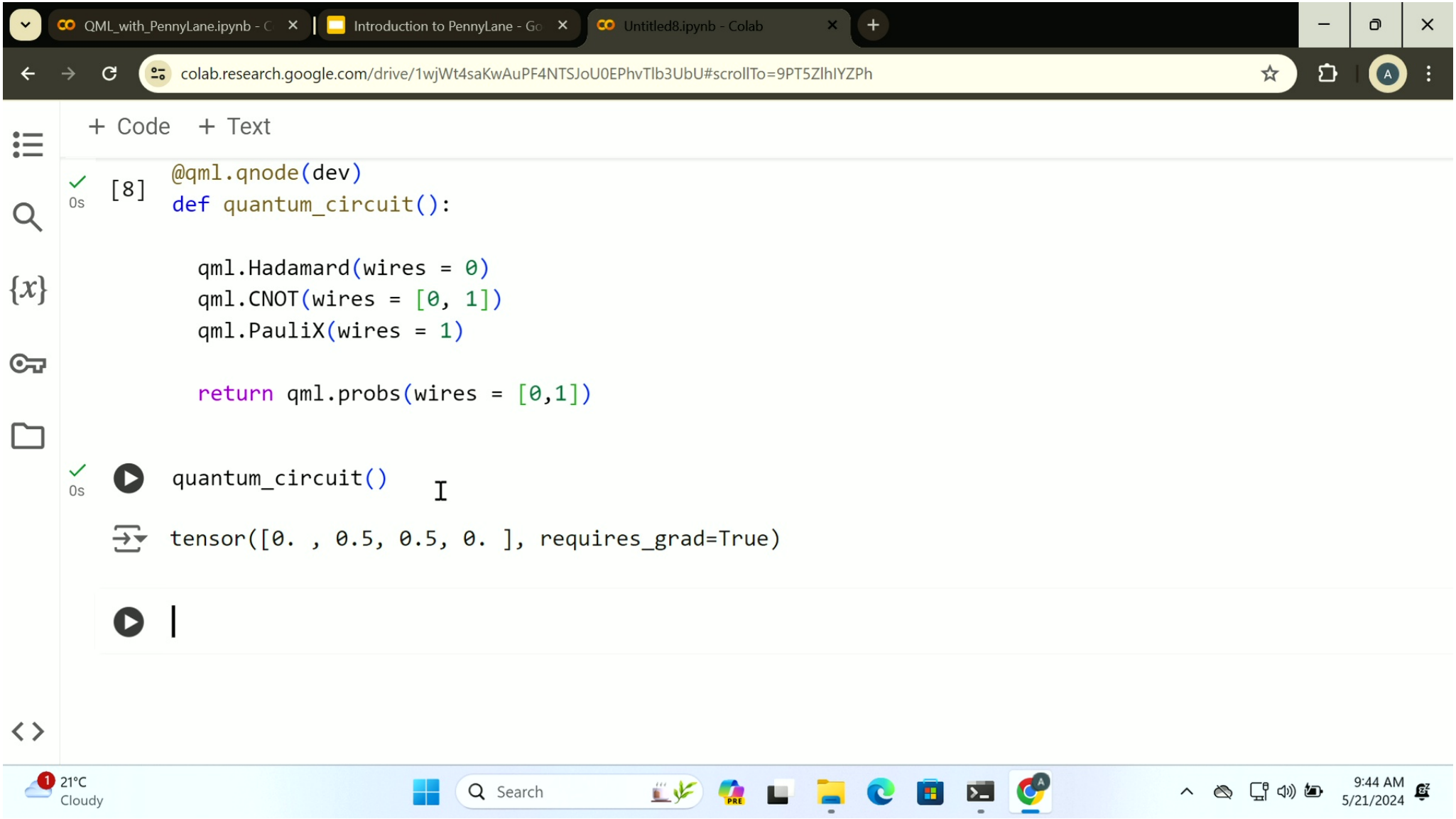
```
dev = qml.device('default.qubit', wires = 2)
```

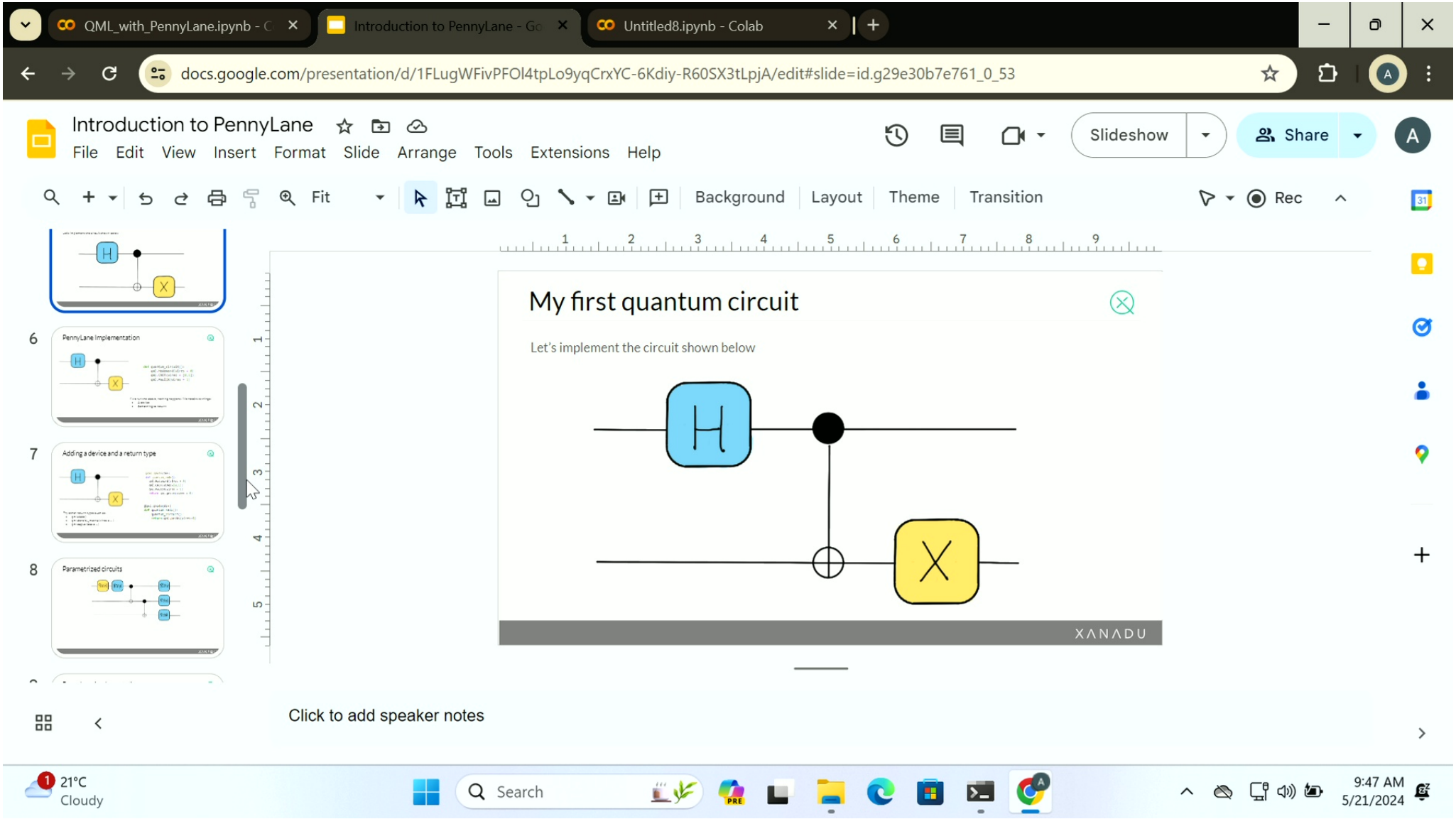
(number of wires optional in default.qubit)


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=4HKrqRNtWkeV
+ Code
+ Text
Successfully installed autoray-0.6.12 pennylane-0.36.0 pennylane-lightning-0.36.0 rustworkx-0.14.2 se
[2]
0s
import pennylane as qml
from pennylane import numpy as np
dev = qml.device('default.qubit', wires = 2)
(name: Any, *args: Any, **kwargs: Any) -> Any
Load a device and return the instance.
This function is used to load a particular quantum device, which can then be used to construct QNodes.
PennyLane comes with support for the following devices:
'default.qubit' <pennylane.devices.default_qubit> : state simulator of qubit-based quantum circuit architectures.
21°C
Cloudy
Search
9:35 AM
5/21/2024

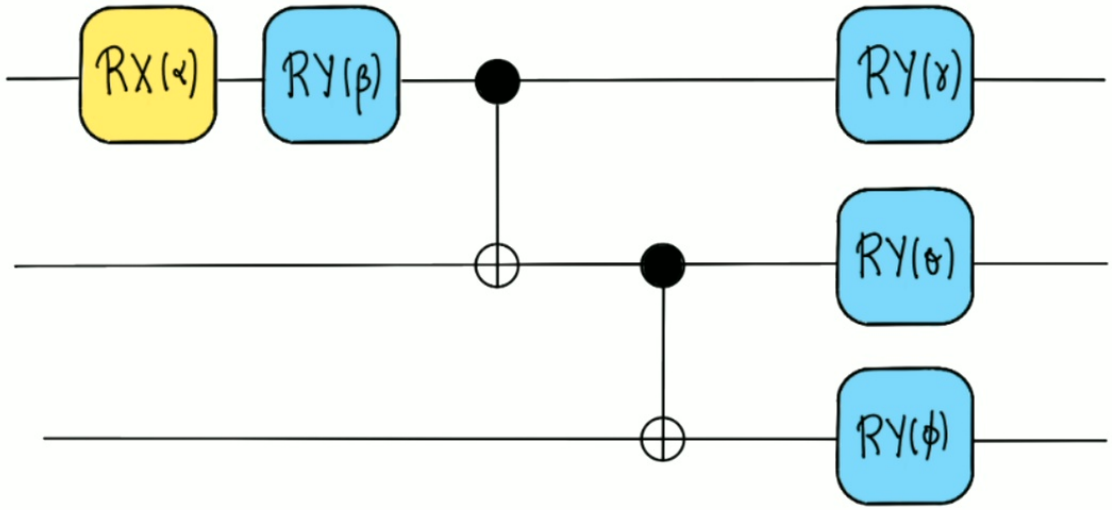

QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=E_fpFzTTXhEi
+ Code
+ Text
[3] dev = qml.device('default.qubit', wires = 2)
0s
def quantum_circuit():
qml.Hadamard()
(*params: Any, wires: Any | None = None, id: Any | None = None) -> None
Hadamard(wires)
The Hadamard operator
Details:
Number of wires: 1
Number of parameters: 0
21°C
Cloudy
Search
9:39 AM
5/21/2024

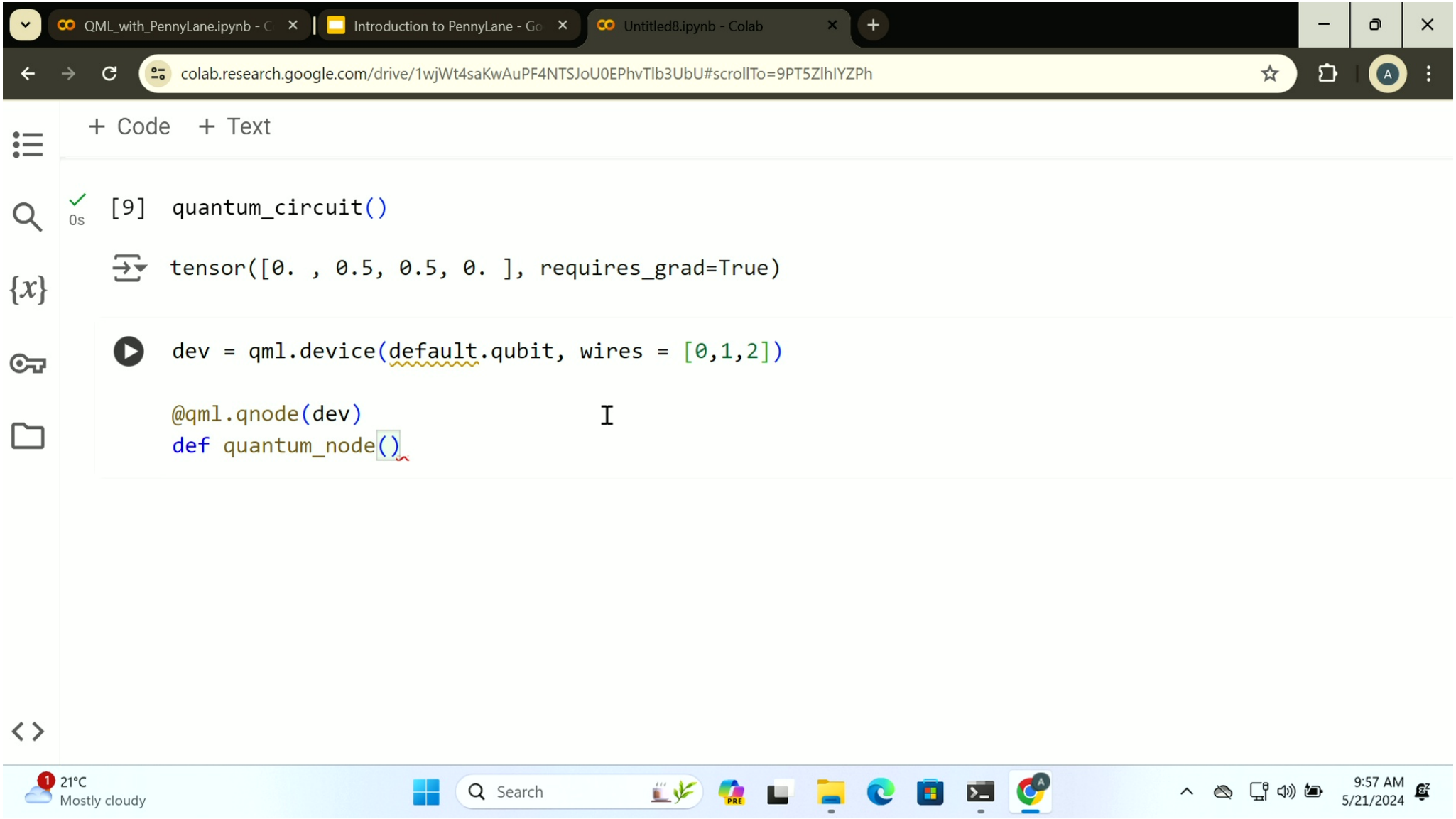

QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=E_fpFzTTXhEi
+ Code
+ Text
ProbabilityMP
wires (Sequence[int] or int): the wire the operation acts on
Probability of each computational basis state.
This measurement function accepts either a wire specification or an observable. Passing wires to the function instructs the QNode to return a flat array containing the probabilities |\langle i | \psi \rangle |^2 of measuring the computational basis state | i \rangle given the current state | \psi \rangle .
Marginal probabilities may also be requested by restricting
[3] dev = qml.device('default.qubit
0s
def quantum_circuit():
qml.Hadamard(wires = 0)
qml.CNOT(wires = [0, 1])
qml.PauliX(wires = 1)
return qml.probs(wires = [0,1])
[5] quantum_circuit()
0s
[ ]
21°C
Cloudy
Search
9:40 AM
5/21/2024

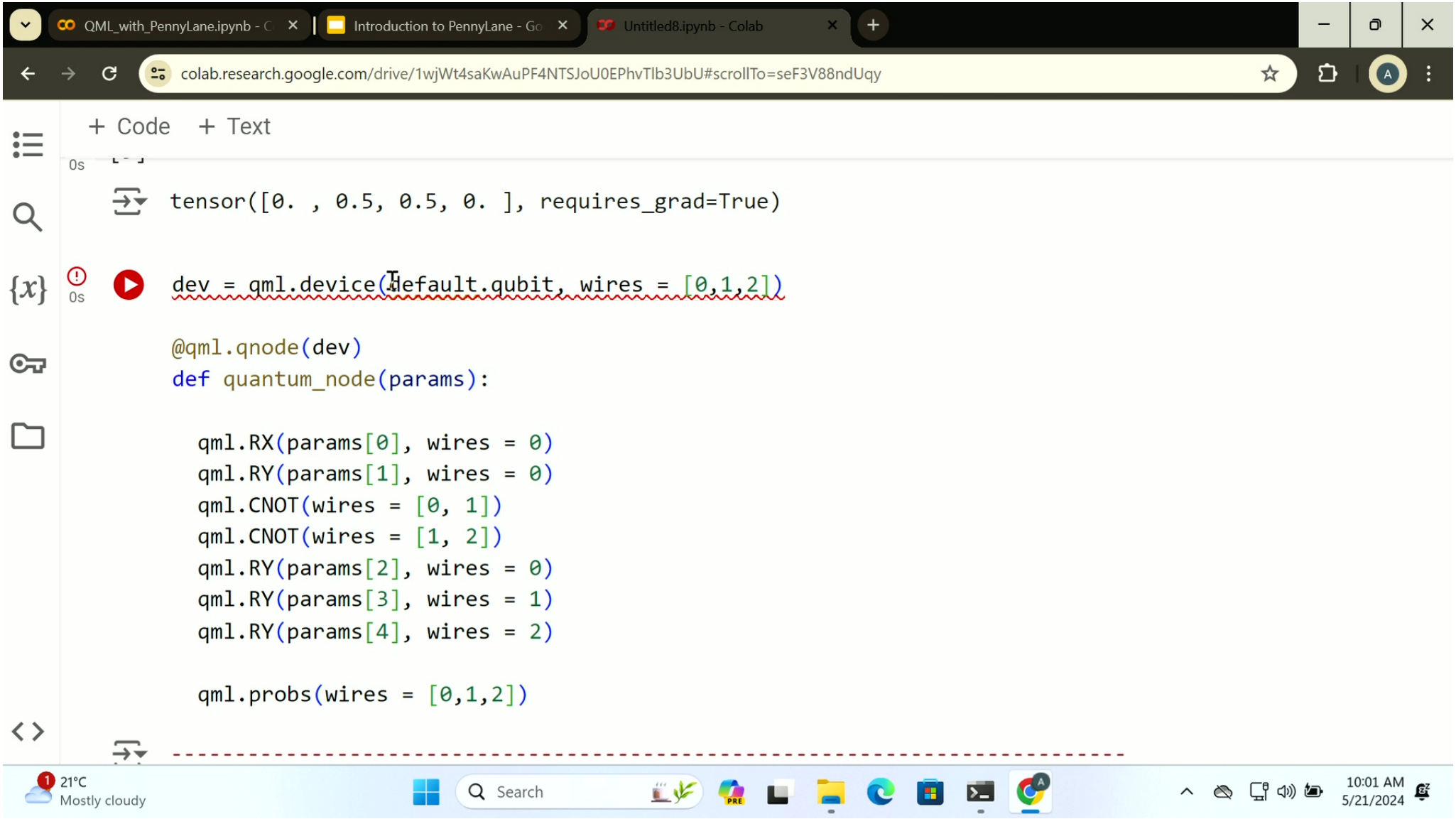

QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=9PT5ZlhIYZPh
+ Code
+ Text
[8]
0s
@qml.qnode(dev)
def quantum_circuit():
qml.Hadamard(wires = 0)
qml.CNOT(wires = [0, 1])
qml.PauliX(wires = 1)
return qml.probs(wires = [0,1])
0s
quantum_circuit()
tensor([0. , 0.5, 0.5, 0.], requires_grad=True)
21°C
Cloudy
Search
9:44 AM
5/21/2024

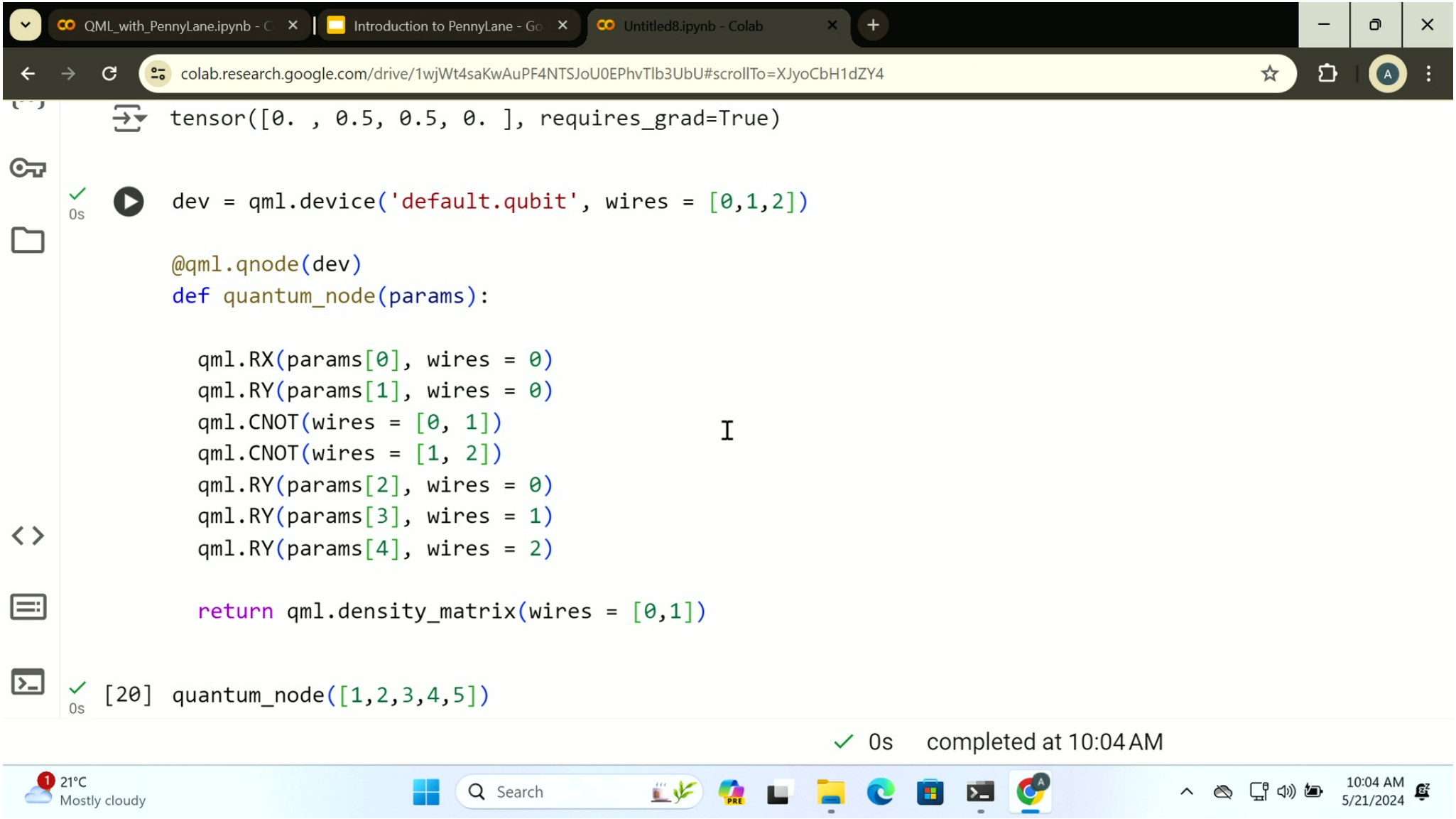

QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
docs.google.com/presentation/d/1FLugWFivPFOl4tpLo9yqCrxYC-6Kdiy-R60SX3tLpjA/edit#slide=id.g29e30b7e761_0_53
Introduction to PennyLane
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Share
Fit
Background
Layout
Theme
Transition
Rec
6 PennyLane implementation
7 Adding a device and a return type
8 Parametrized circuits
My first quantum circuit
Let's implement the circuit shown below
H
X
XANADU
Click to add speaker notes
21°C
Cloudy
Search
9:47 AM
5/21/2024

Parametrized circuits




QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=9PT5ZlhIYZPh
+ Code
+ Text
[9] quantum_circuit()
0s
tensor([0. , 0.5, 0.5, 0.], requires_grad=True)
dev = qml.device(default.qubit, wires = [0,1,2])
@qml.qnode(dev)
def quantum_node()
21°C
Mostly cloudy
Search
9:57 AM
5/21/2024


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=seF3V88ndUqy
+ Code
+ Text
0s
tensor([0. , 0.5, 0.5, 0.], requires_grad=True)
0s
dev = qml.device(default.qubit, wires = [0,1,2])
@qml.qnode(dev)
def quantum_node(params):
qml.RX(params[0], wires = 0)
qml.RY(params[1], wires = 0)
qml.CNOT(wires = [0, 1])
qml.CNOT(wires = [1, 2])
qml.RY(params[2], wires = 0)
qml.RY(params[3], wires = 1)
qml.RY(params[4], wires = 2)
qml.probs(wires = [0,1,2])
21°C
Mostly cloudy
Search
10:01 AM
5/21/2024


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=XJyoCbH1dZY4
tensor([0. , 0.5, 0.5, 0.], requires_grad=True)
0s
dev = qml.device('default.qubit', wires = [0,1,2])
@qml.qnode(dev)
def quantum_node(params):
qml.RX(params[0], wires = 0)
qml.RY(params[1], wires = 0)
qml.CNOT(wires = [0, 1])
qml.CNOT(wires = [1, 2])
qml.RY(params[2], wires = 0)
qml.RY(params[3], wires = 1)
qml.RY(params[4], wires = 2)
return qml.density_matrix(wires = [0,1])
0s
[20] quantum_node([1,2,3,4,5])
0s completed at 10:04 AM
21°C
Mostly cloudy
Search
10:04 AM
5/21/2024

QML_with_PennyLane.ipynb - C | Introduction to PennyLane - Gc | Untitled8.ipynb - Colab

docs.google.com/presentation/d/1FLugWFivPFOl4tpLo9yqCrxYC-6Kdiy-R60SX3tLpjA/edit#slide=id.g2cb64d55baa_0_35

Introduction to PennyLane

File Edit View Insert Format Slide Arrange Tools Extensions Help

Slideshow | Share

Fit | Background | Layout | Theme | Transition | Rec

10 Circuits as models

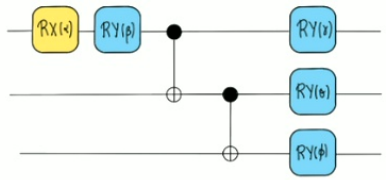
11 Cost function minimization

12 Master PennyLane

13 Thank you

13 total slides

PennyLane implementation

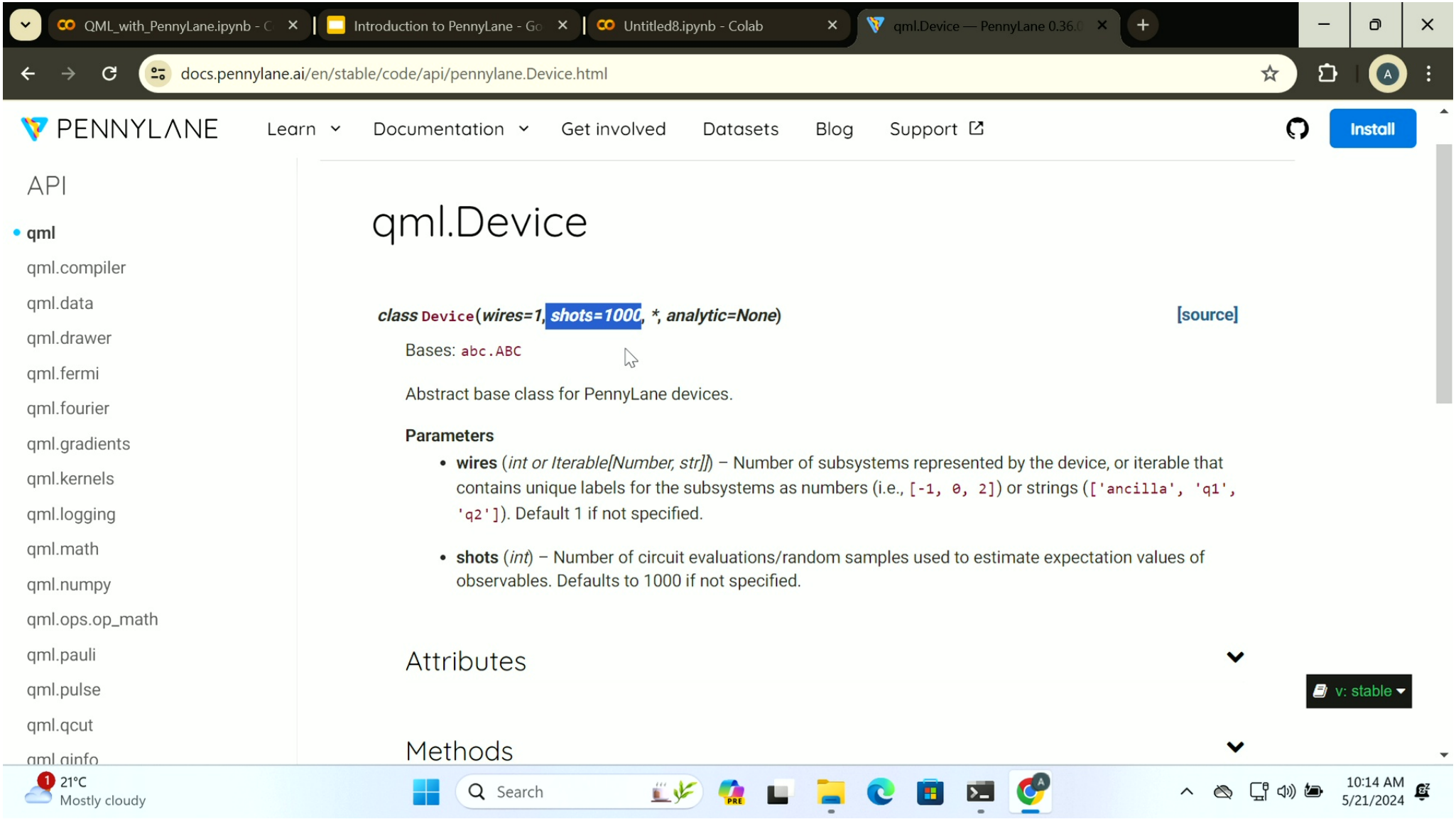



```
dev = qml.device('default.qubit', wires = 3)

@qml.qnode(dev)
def quantum_node(params):
  qml.RY(params[0], wires = 0)
  qml.RY(params[1], wires = 0)
  qml.CNOT(wires=[0,1])
  qml.CNOT(wires=[1,2])
  qml.RY(params[2], wires = 0)
  qml.RY(params[3], wires = 1)
  qml.RY(params[4], wires = 2)

  return qml.expval(qml.PauliZ(0))
```

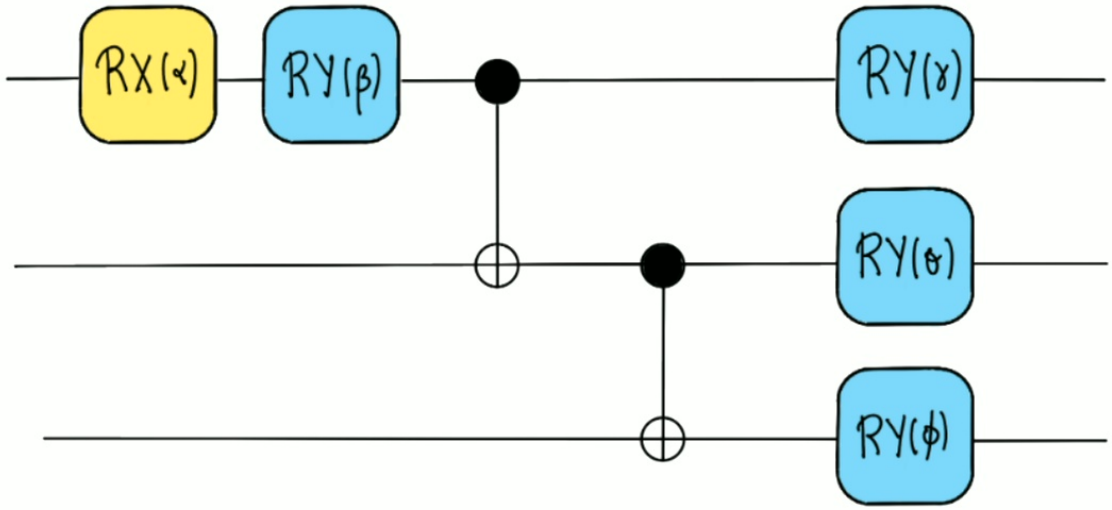
XANADU

Click to add speaker notes

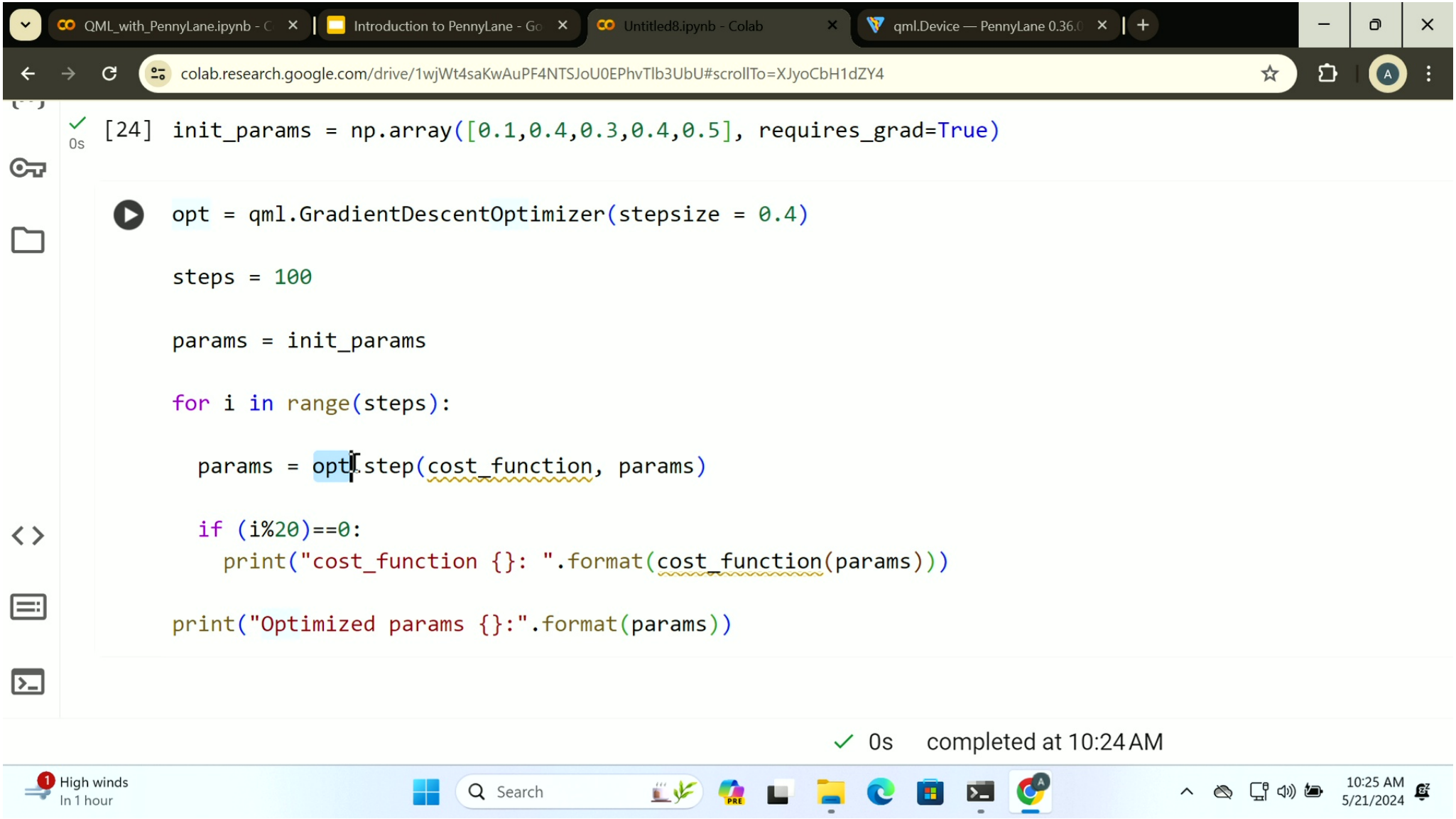
21°C Mostly cloudy | Search | 10:11 AM 5/21/2024

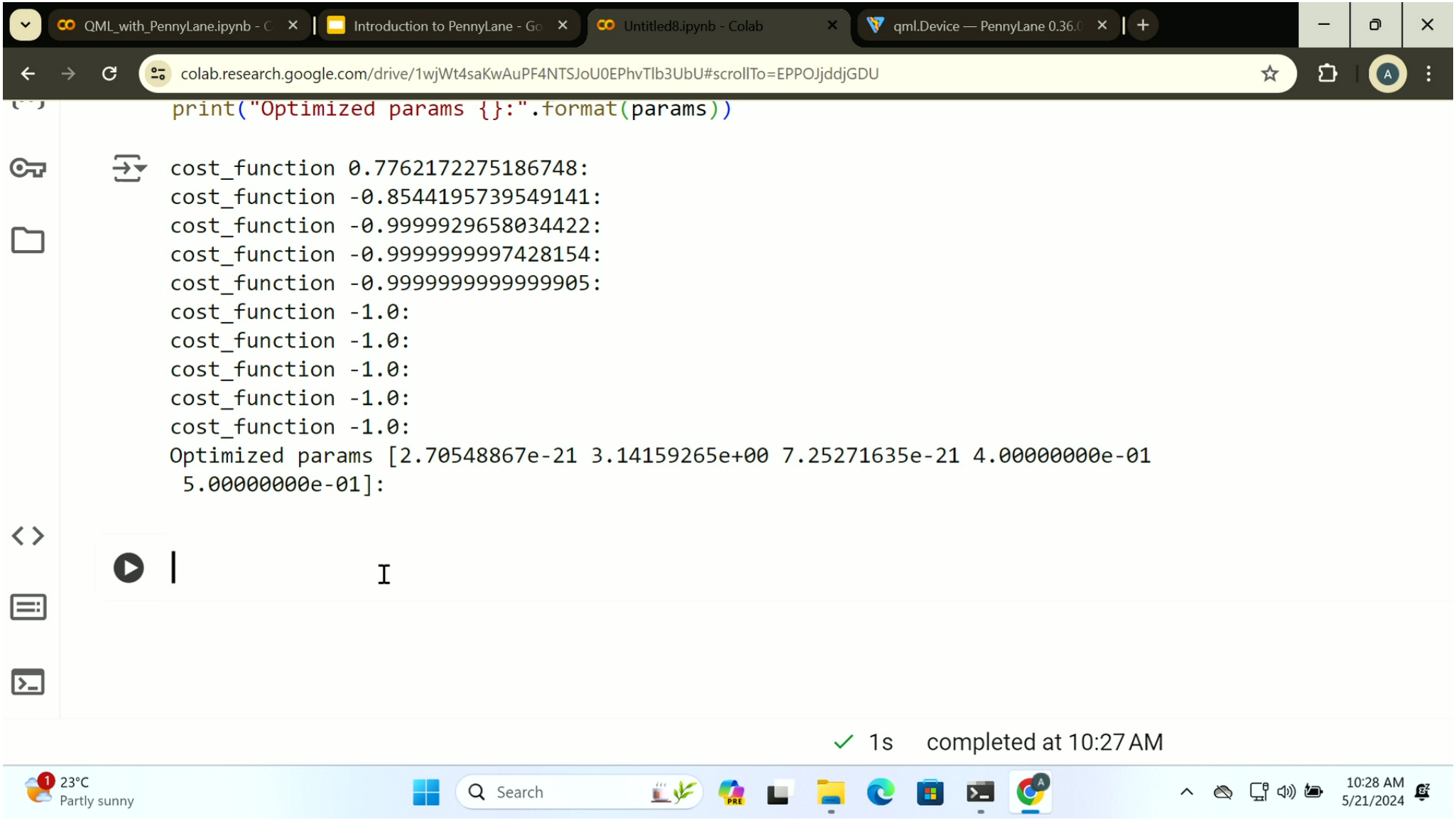

QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
docs.pennylane.ai/en/stable/code/api/pennylane.Device.html
PENNYLANE
Learn
Documentation
Get involved
Datasets
Blog
Support
Install
API
qml
qml.compiler
qml.data
qml.drawer
qml.fermi
qml.fourier
qml.gradients
qml.kernels
qml.logging
qml.math
qml.numpy
qml.ops.op_math
qml.pauli
qml.pulse
qml.qcut
qml.qinfo
qml.Device
class Device(wires=1, shots=1000, *, analytic=None)
[source]
Bases: abc.ABC
Abstract base class for PennyLane devices.
Parameters
wires (int or Iterable[Number, str]) – Number of subsystems represented by the device, or iterable that contains unique labels for the subsystems as numbers (i.e., [-1, 0, 2]) or strings (['ancilla', 'q1', 'q2']). Default 1 if not specified.
shots (int) – Number of circuit evaluations/random samples used to estimate expectation values of observables. Defaults to 1000 if not specified.
Attributes
Methods
v: stable
21°C
Mostly cloudy
Search
10:14 AM
5/21/2024

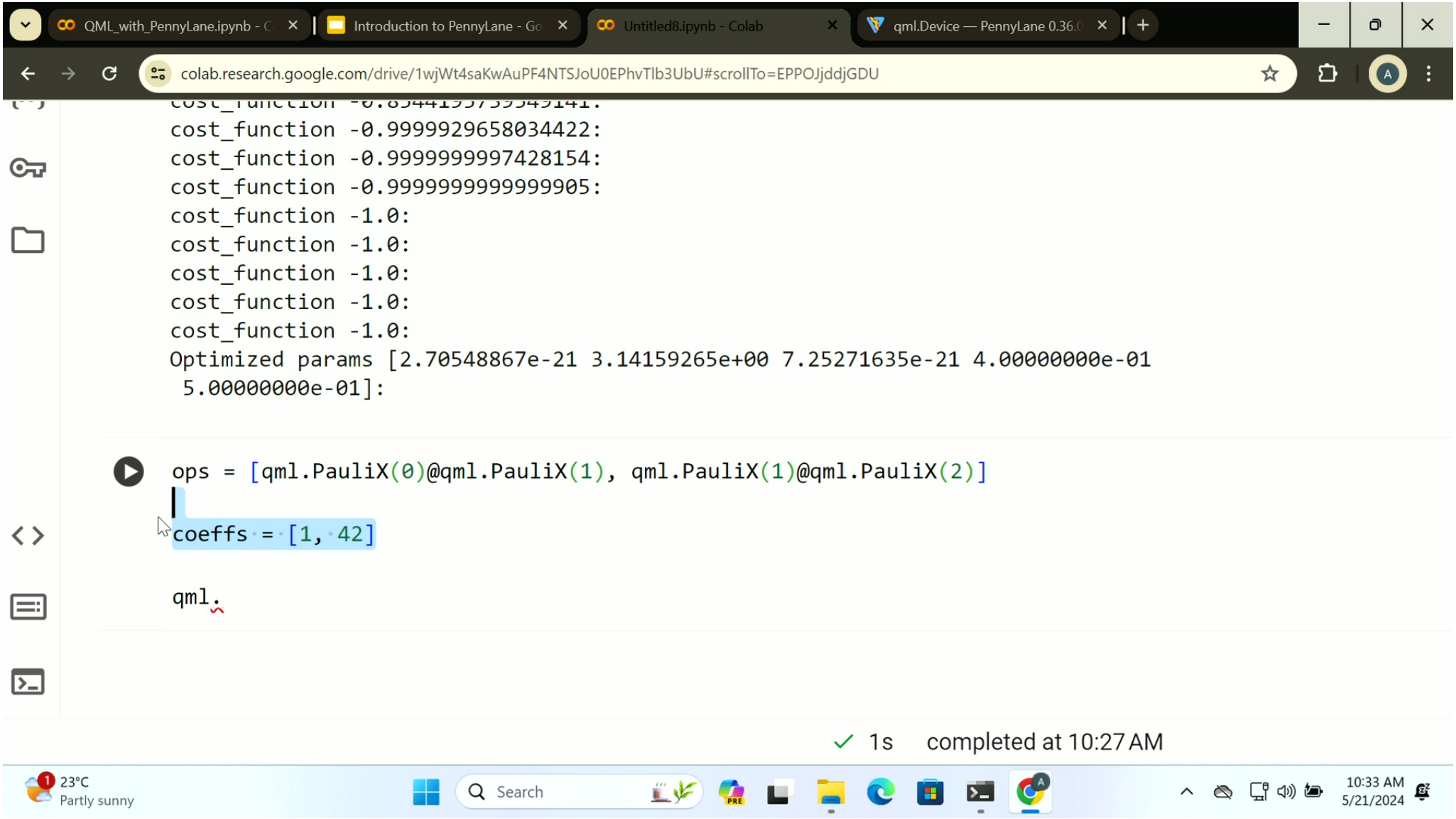
Circuits as models

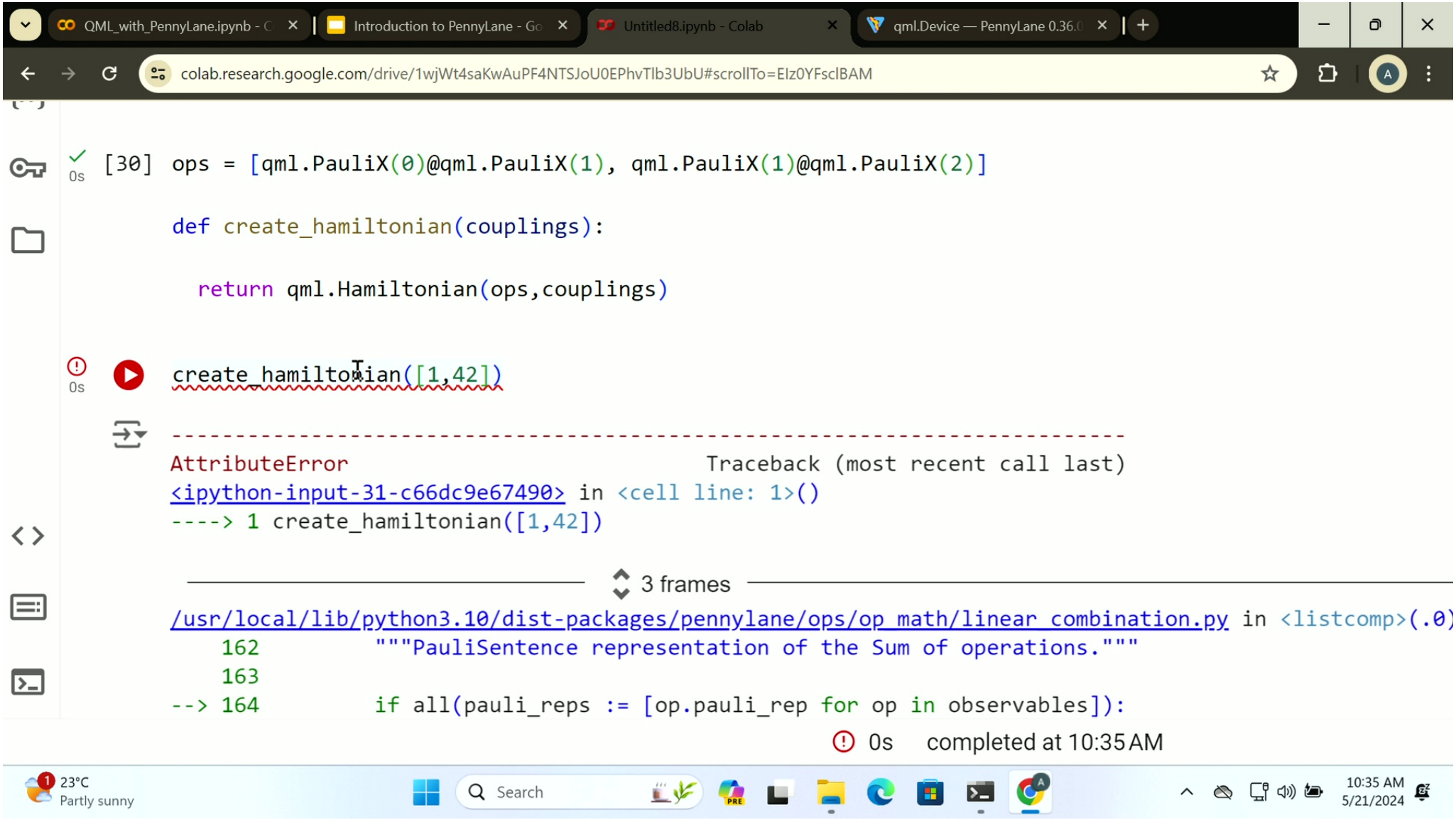


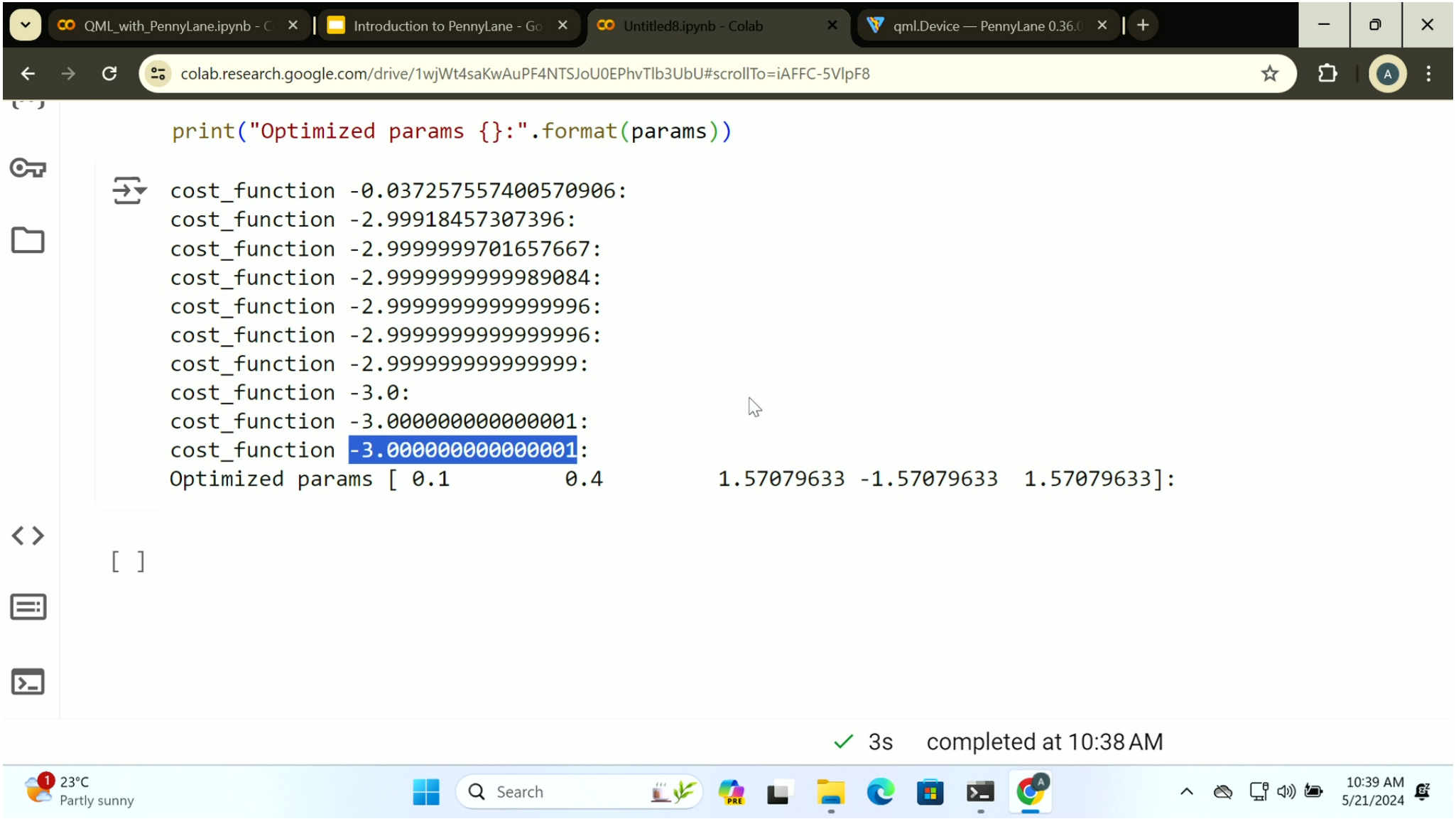
If we return e.g. qml.expval(obs) we can interpret this as a function f: $R^5 \rightarrow R$.
Let's do what we usually like to do with functions and try to find extrema!

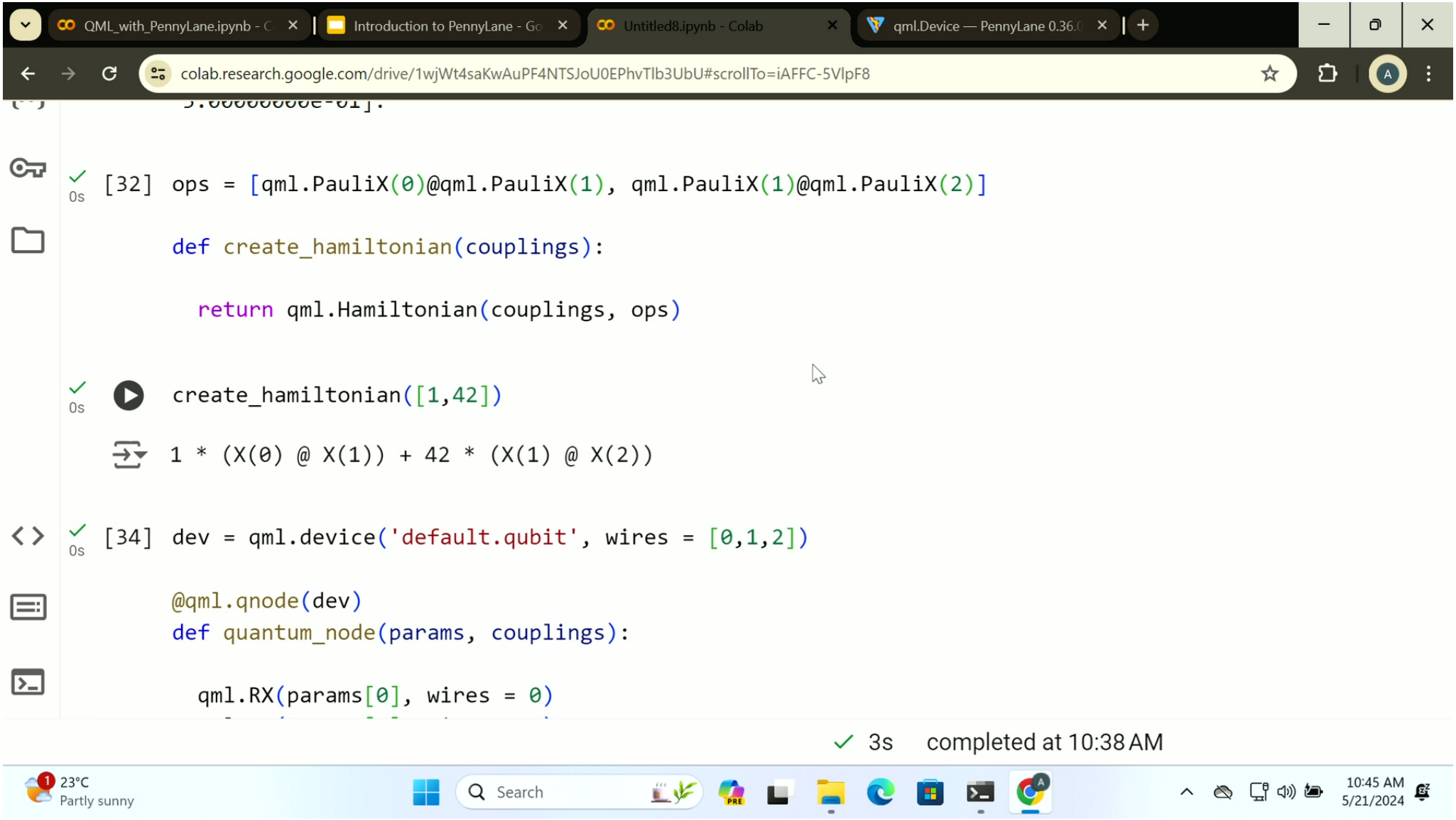

QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=XJyoCbH1dZY4
[24] init_params = np.array([0.1,0.4,0.3,0.4,0.5], requires_grad=True)
0s
opt = qml.GradientDescentOptimizer(stepsize = 0.4)
steps = 100
params = init_params
for i in range(steps):
params = opt.step(cost_function, params)
if (i%20)==0:
print("cost_function {}: ".format(cost_function(params)))
print("Optimized params {}:".format(params))
0s completed at 10:24 AM
High winds
In 1 hour
Search
10:25 AM
5/21/2024

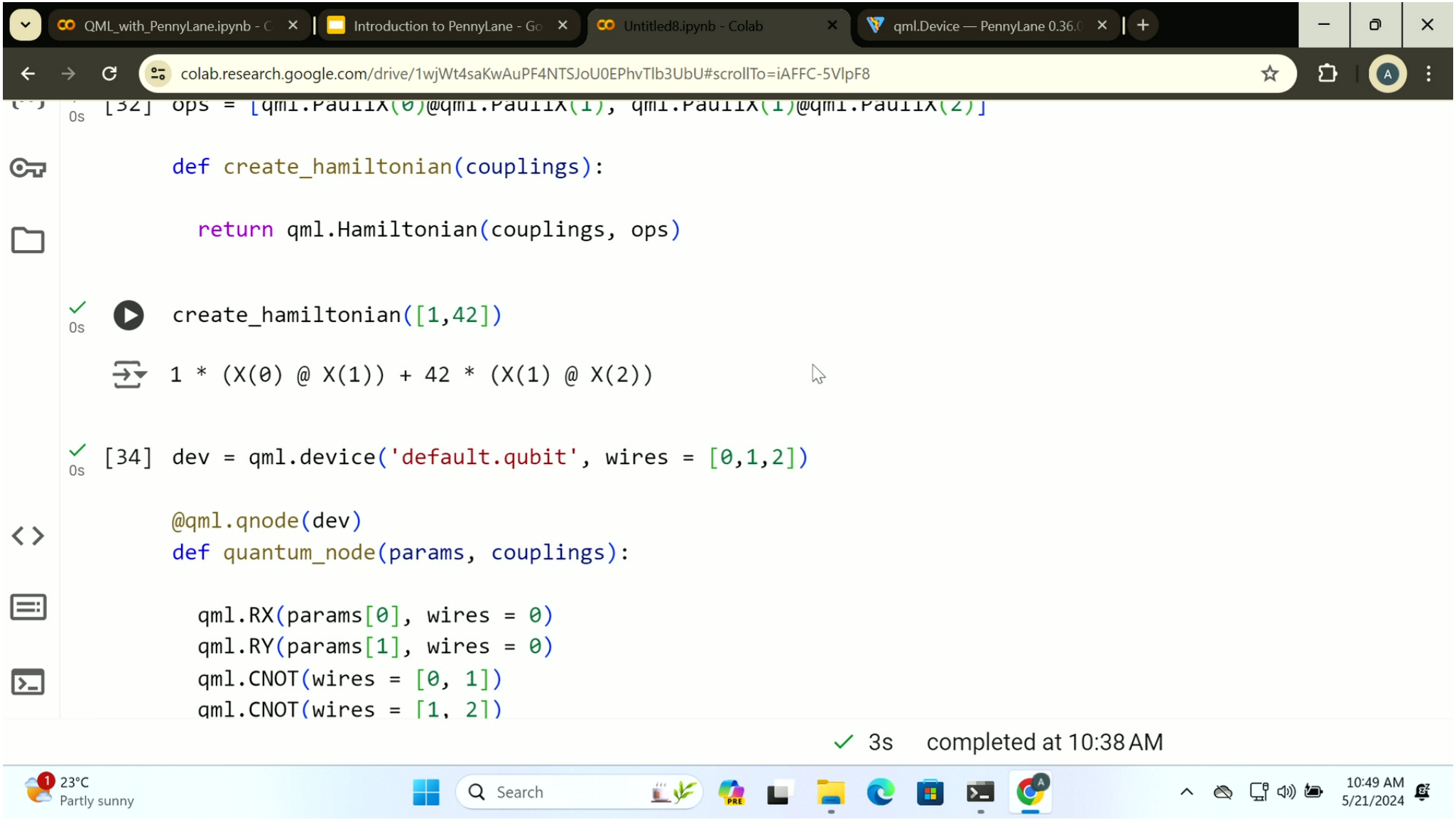

QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=EPPOJjddjGDU
print("Optimized params {}:".format(params))
cost_function 0.7762172275186748:
cost_function -0.8544195739549141:
cost_function -0.9999929658034422:
cost_function -0.9999999997428154:
cost_function -0.9999999999999905:
cost_function -1.0:
cost_function -1.0:
cost_function -1.0:
cost_function -1.0:
cost_function -1.0:
Optimized params [2.70548867e-21 3.14159265e+00 7.25271635e-21 4.00000000e-01
5.00000000e-01]:
1s completed at 10:27 AM
23°C
Partly sunny
Search
10:28 AM
5/21/2024


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=EPPOJjddjGDU
cost_function -0.9999929658034422:
cost_function -0.9999999997428154:
cost_function -0.9999999999999905:
cost_function -1.0:
cost_function -1.0:
cost_function -1.0:
cost_function -1.0:
cost_function -1.0:
Optimized params [2.70548867e-21 3.14159265e+00 7.25271635e-21 4.00000000e-01
5.00000000e-01]:
ops = [qml.PauliX(0)@qml.PauliX(1), qml.PauliX(1)@qml.PauliX(2)]
coeffs = [1, 42]
qml.
1s completed at 10:27 AM
23°C
Partly sunny
Search
10:33 AM
5/21/2024


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=Elz0YFsclBAM
[30] ops = [qml.PauliX(0)@qml.PauliX(1), qml.PauliX(1)@qml.PauliX(2)]
def create_hamiltonian(couplings):
return qml.Hamiltonian(ops,couplings)
create_hamiltonian([1,42])
AttributeError Traceback (most recent call last)
<ipython-input-31-c66dc9e67490> in <cell line: 1>()
----> 1 create_hamiltonian([1,42])
3 frames
/usr/local/lib/python3.10/dist-packages/pennylane/ops/op_math/linear_combination.py in <listcomp>(.0)
162 """PauliSentence representation of the Sum of operations."""
163
--> 164 if all(pauli_reps := [op.pauli_rep for op in observables]):
0s completed at 10:35 AM
23°C
Partly sunny
Search
10:35 AM
5/21/2024


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=iAFFC-5VlpF8
print("Optimized params {}:".format(params))
cost_function -0.037257557400570906:
cost_function -2.99918457307396:
cost_function -2.9999999701657667:
cost_function -2.9999999999989084:
cost_function -2.9999999999999996:
cost_function -2.9999999999999996:
cost_function -2.999999999999999:
cost_function -3.0:
cost_function -3.000000000000001:
cost_function -3.000000000000001:
Optimized params [0.1 0.4 1.57079633 -1.57079633 1.57079633]:
[ ]
3s completed at 10:38 AM
23°C
Partly sunny
Search
10:39 AM
5/21/2024


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=iAFFC-5VlpF8
[32] ops = [qml.PauliX(0)@qml.PauliX(1), qml.PauliX(1)@qml.PauliX(2)]
def create_hamiltonian(couplings):
return qml.Hamiltonian(couplings, ops)
create_hamiltonian([1,42])
1 * (X(0) @ X(1)) + 42 * (X(1) @ X(2))
[34] dev = qml.device('default.qubit', wires = [0,1,2])
@qml.qnode(dev)
def quantum_node(params, couplings):
qml.RX(params[0], wires = 0)
3s completed at 10:38 AM
23°C
Partly sunny
Search
10:45 AM
5/21/2024


QML_with_PennyLane.ipynb - C
Introduction to PennyLane - Go
Untitled8.ipynb - Colab
qml.Device — PennyLane 0.36.0
colab.research.google.com/drive/1wjWt4saKwAuPF4NTSJoU0EPhvTlb3UbU#scrollTo=iAFFC-5VlpF8
0s
[32] ops = [qml.PauliX(0)@qml.PauliX(1), qml.PauliX(1)@qml.PauliX(2)]
def create_hamiltonian(couplings):
return qml.Hamiltonian(couplings, ops)
0s
create_hamiltonian([1,42])
1 * (X(0) @ X(1)) + 42 * (X(1) @ X(2))
0s
[34] dev = qml.device('default.qubit', wires = [0,1,2])
@qml.qnode(dev)
def quantum_node(params, couplings):
qml.RX(params[0], wires = 0)
qml.RY(params[1], wires = 0)
qml.CNOT(wires = [0, 1])
qml.CNOT(wires = [1, 2])
3s completed at 10:38 AM
23°C
Partly sunny
Search
10:49 AM
5/21/2024