Introduction to Quantum Information and Quantum Machine Learning

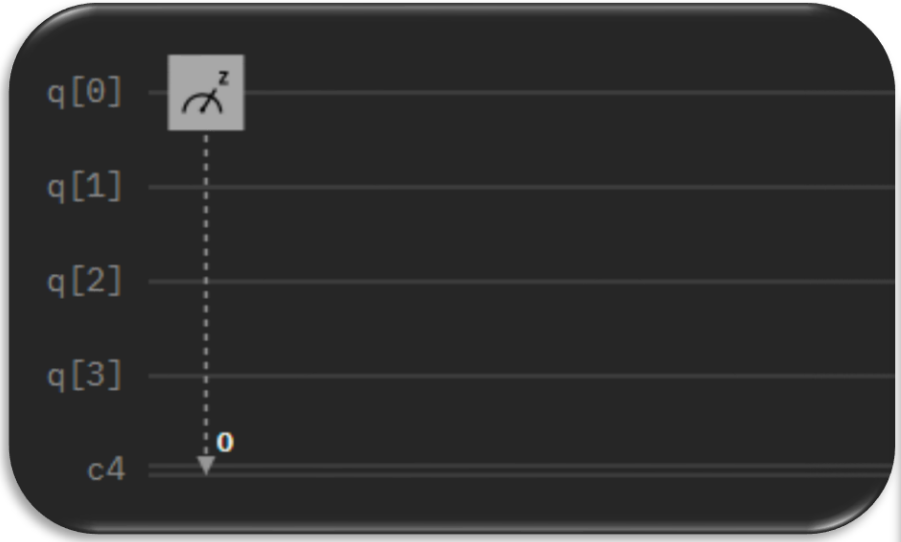
Project 1

Dr Gustaw Szawioła, docent PUT
Dr Sci. Eng. Przemysław Głowacki

1. Task 1

IQI AND QML
D. Sc. Eng. Przemysław Głowacki

Z-type projection measurement – reading of the qubit state



QUANTUM COMPOSER CODE
Graphical code

Qiskit

Read only

Open in Quantum Lab

```python
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.measure(qreg_q[0], creg_c[0])
```

code (Qiskit)

OpenQASM 2.0

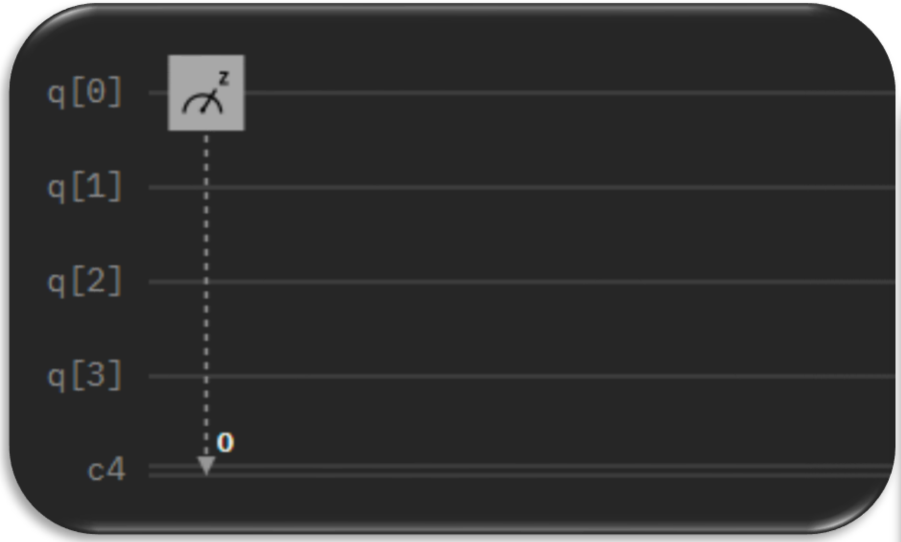
Open in Quantum Lab

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[4];
creg c[4];
measure q[0] -> c[0];

```

code (OpenQASM 2.0)

Z-type projection measurement – reading of the qubit state

Python 3.13 Qiskit 2.2.1



QUANTUM COMPOSER CODE
Graphical code

Qiskit

Read only

Open in Quantum Lab

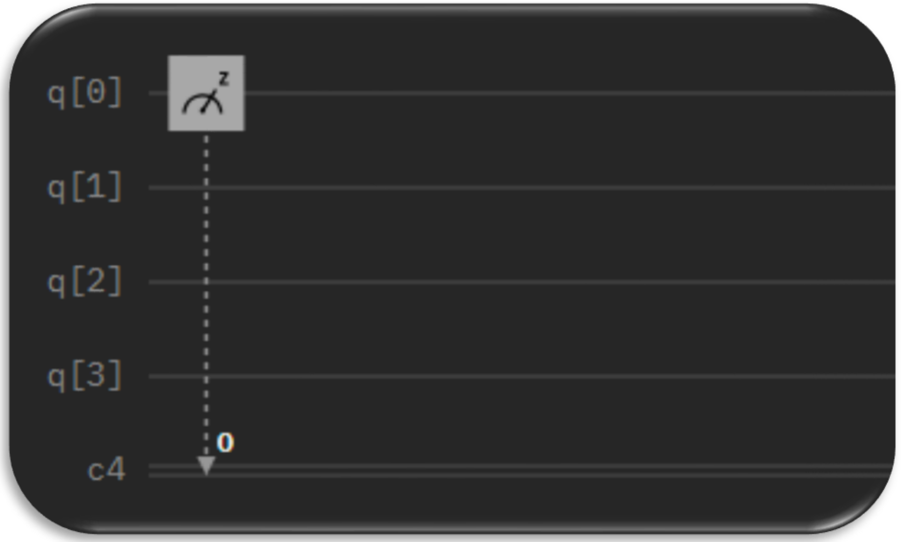
```python
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.measure(qreg_q[0], creg_c[0])
```

code (Qiskit)

```python
from qiskit import QuantumRegister,
ClassicalRegister, QuantumCircuit
from qiskit_aer import Aer
from qiskit.compiler import transpile
from qiskit.visualization import *
from numpy import pi
# selection of quantum simulator (or processor)
backend =
Aer.get_backend('qasm_simulator')
```

code (Qiskit) additional code: import, „provider” and „beckend”

Z-type projection measurement – reading of the qubit state




```
n=4 # determining the number of quantum and classical registers
seria=2048 # number of trials in a series of measurements of a given type
# Example of measuring the state of qubit q[0]
nx = n # Number of qubits and bits
qx = QuantumRegister(nx) #
cx = ClassicalRegister(nx) #
circuitX = QuantumCircuit(qx, cx) # Quantum algorithm - quantum circuit
circuitX.measure(qx[0], cx[0]) # Checking the states of qubits - quantum measurement
```

QUANTUM COMPOSER CODE

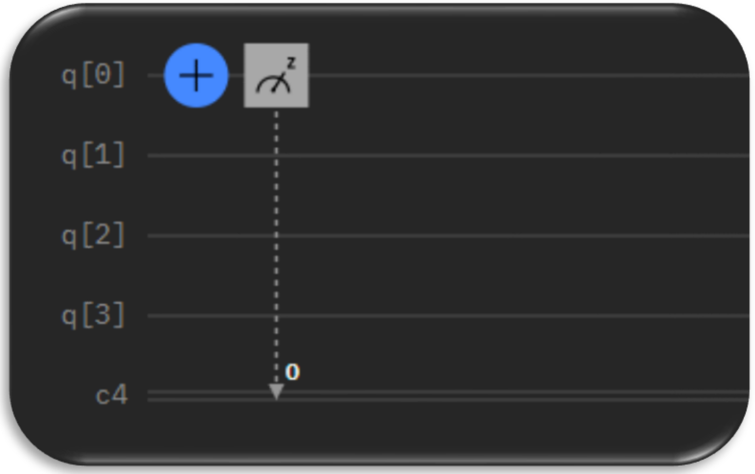
Graphical code

```
circuitX.draw(output='mpl’)
# Drawing a quantum circuit
```

code (Qiskit)

3. Task 2

Operation and reading of the result of quantum gate X



QUANTUM COMPOSER CODE
Graphical code

Qiskit ⌄ Read only

Open in Quantum Lab

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.x(qreg_q[0])
circuit.measure(qreg_q[0], creg_c[0])
```

code (Qiskit)

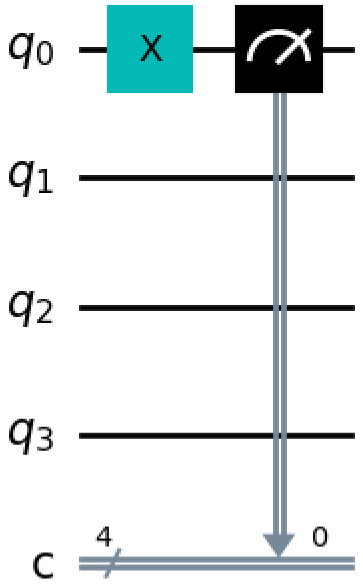
OpenQASM 2.0 ⌄

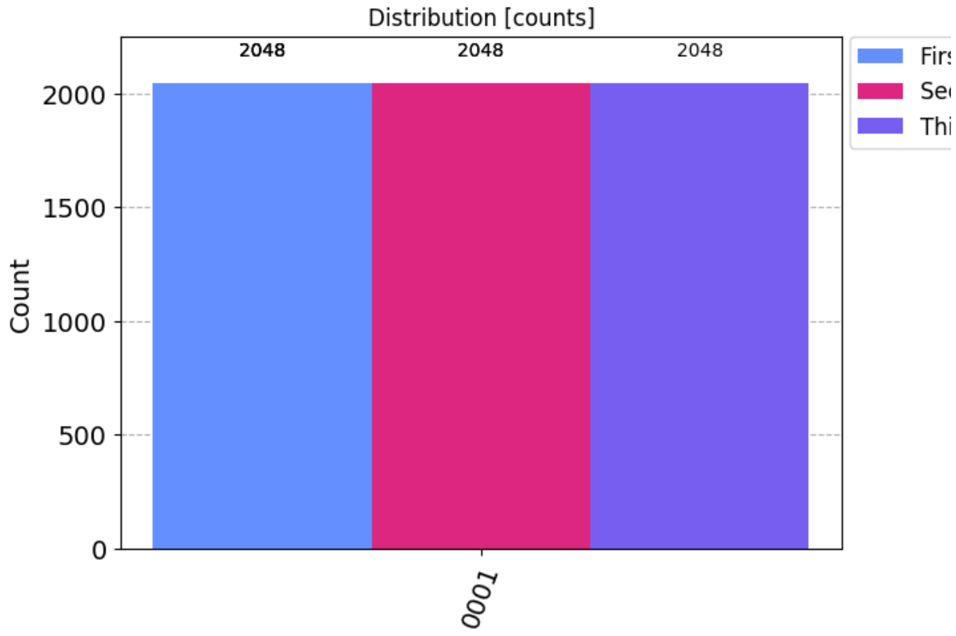
Open in Quantum Lab

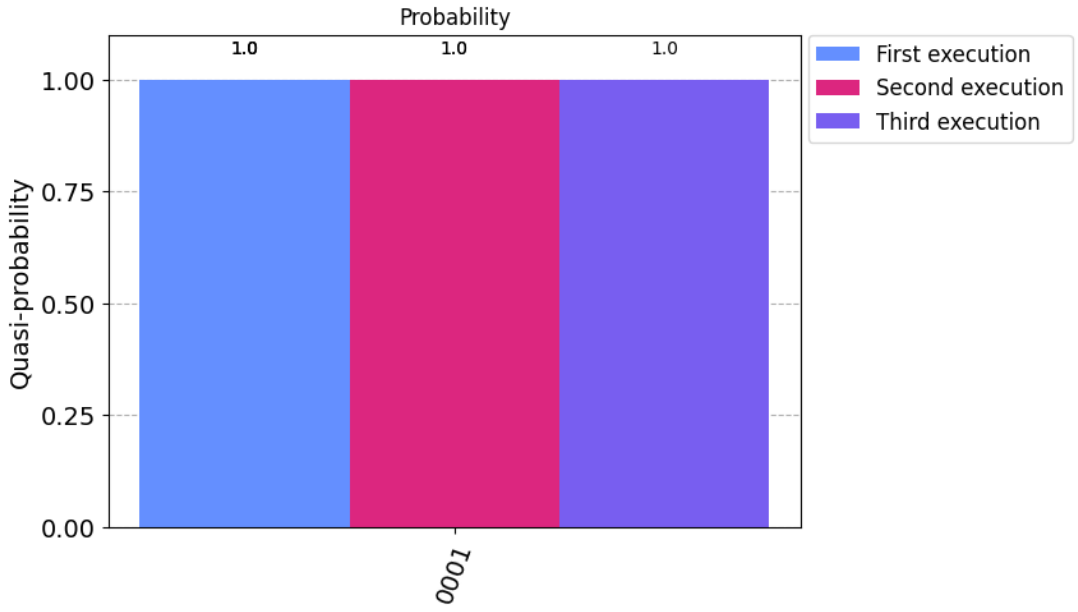
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[4];
creg c[4];
x q[0];
measure q[0] -> c[0];
```

code (OpenQASM 2.0)

Classic measurement simulation







Qiskit output show the quantum circuit

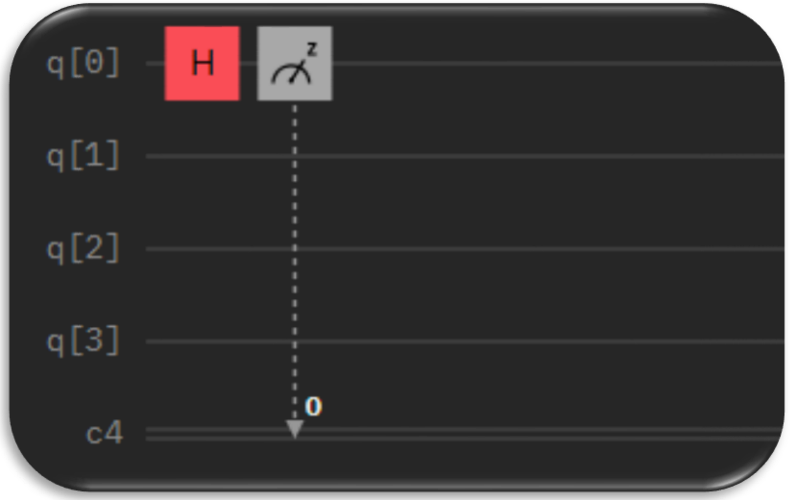
Qiskit output results as Plot 1 „Counts”

Plot 2 „Probability”

4. Task 3

SUPERPOSITION OF STATES

Operation and reading of the result of Hadamard quantum gate (H gate)



QUANTUM COMPOSER CODE
Graphical code

Qiskit
Read only
Open in Quantum Lab

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.h(qreg_q[0])
circuit.measure(qreg_q[0], creg_c[0])
```

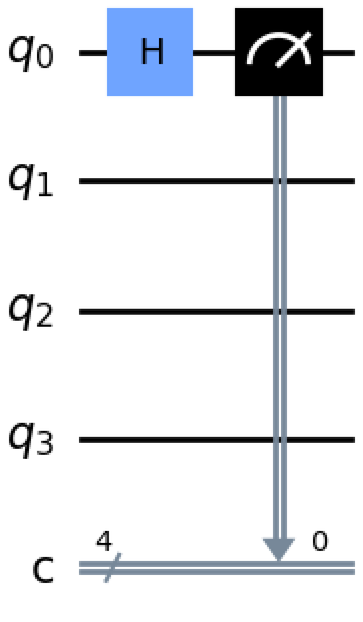
code (Qiskit)

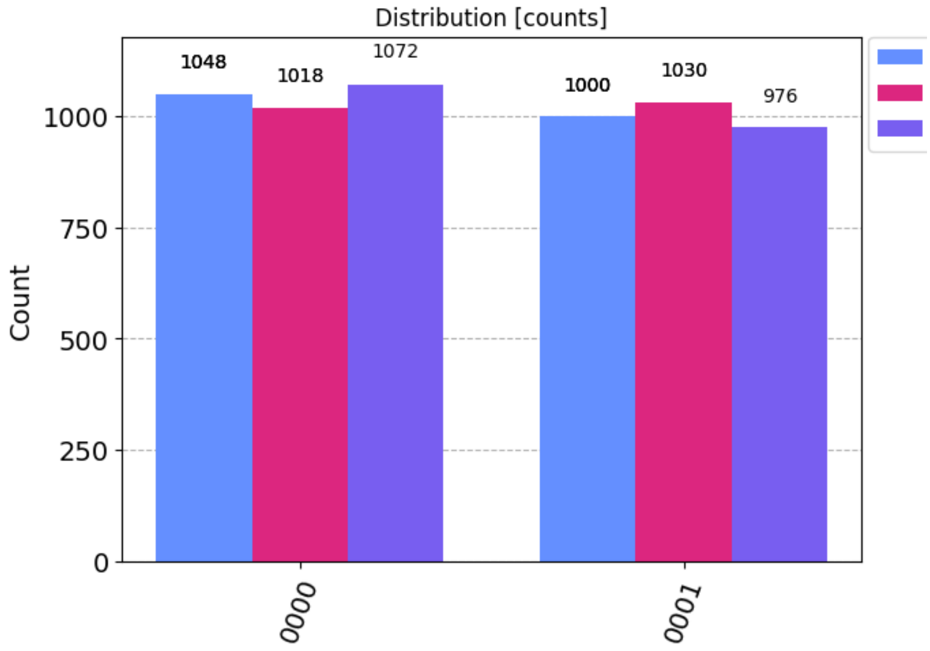
OpenQASM 2.0
Open in Quantum Lab

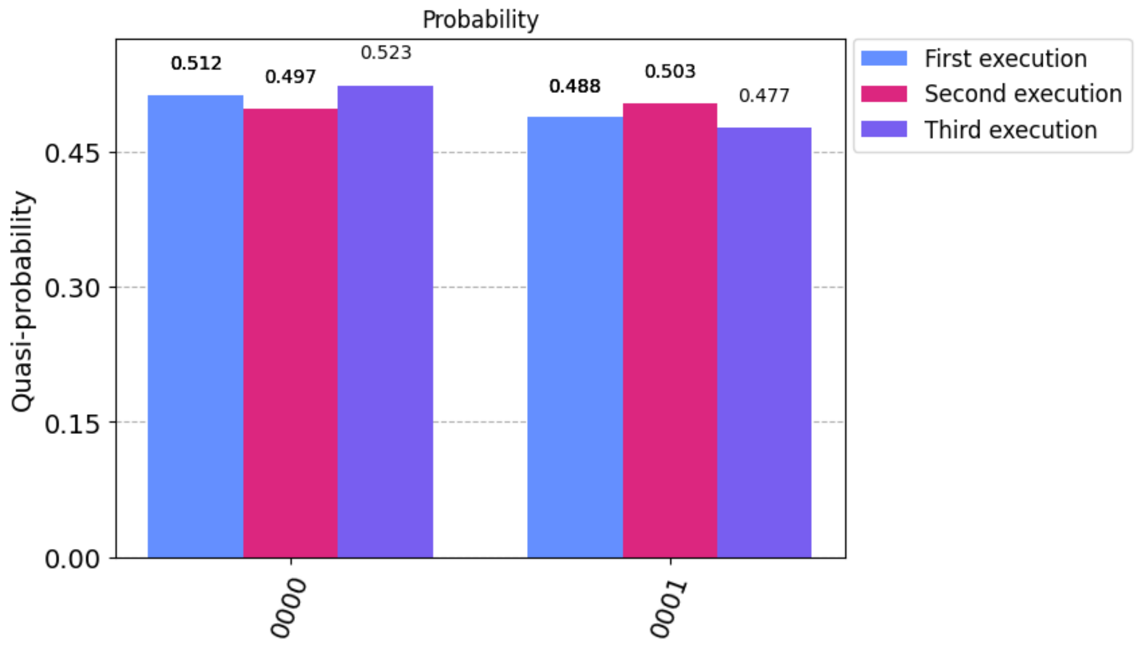
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[4];
creg c[4];
h q[0];
measure q[0] -> c[0];
```

code (OpenQASM 2.0)

Classic measurement simulation – Qiskit and Python







Qiskit output show the quantum circuit

Qiskit output results as Plot 1 „Counts”

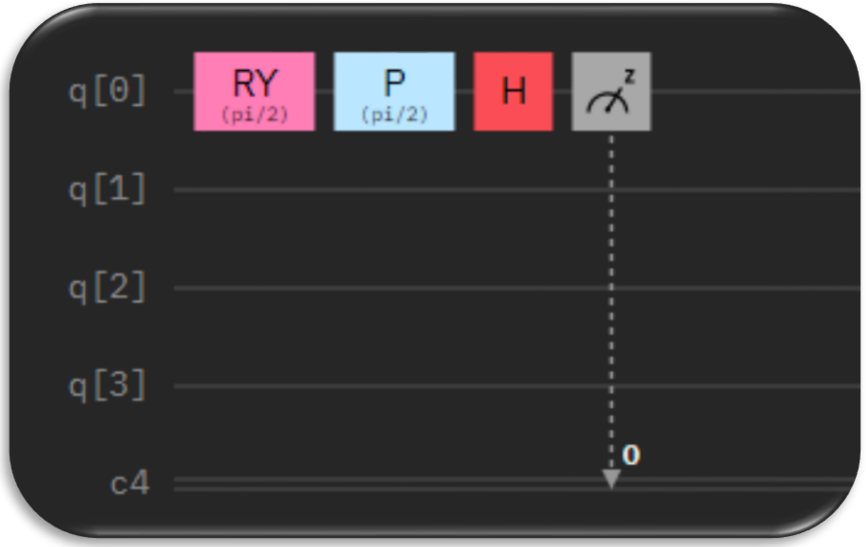
Plot 2 „Probability”

5. Task 4

State tomography of one qubit – implementations on a quantum computer

Measurement in the X base

$$\sigma_1 = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$



QUANTUM COMPOSER CODE
Graphical code

Qiskit
Read only
Open in Quantum Lab

```python
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.ry(pi / 2, qreg_q[0])
circuit.p(pi / 2, qreg_q[0])
circuit.h(qreg_q[0])
circuit.measure(qreg_q[0], creg_c[0])
```

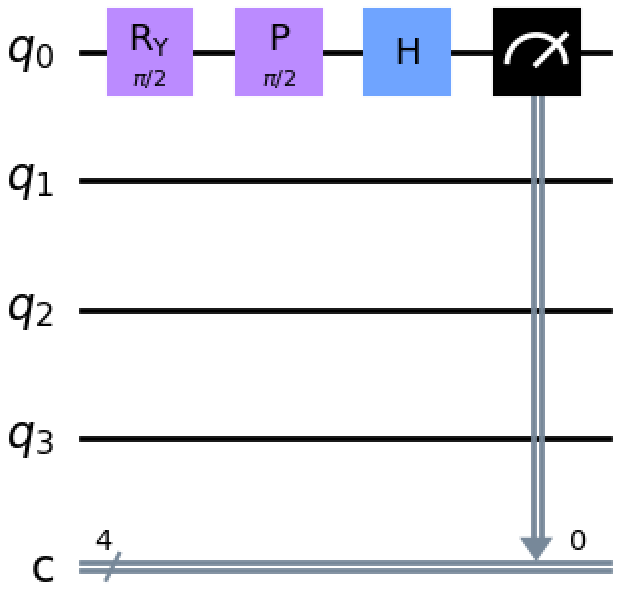
code (Qiskit)

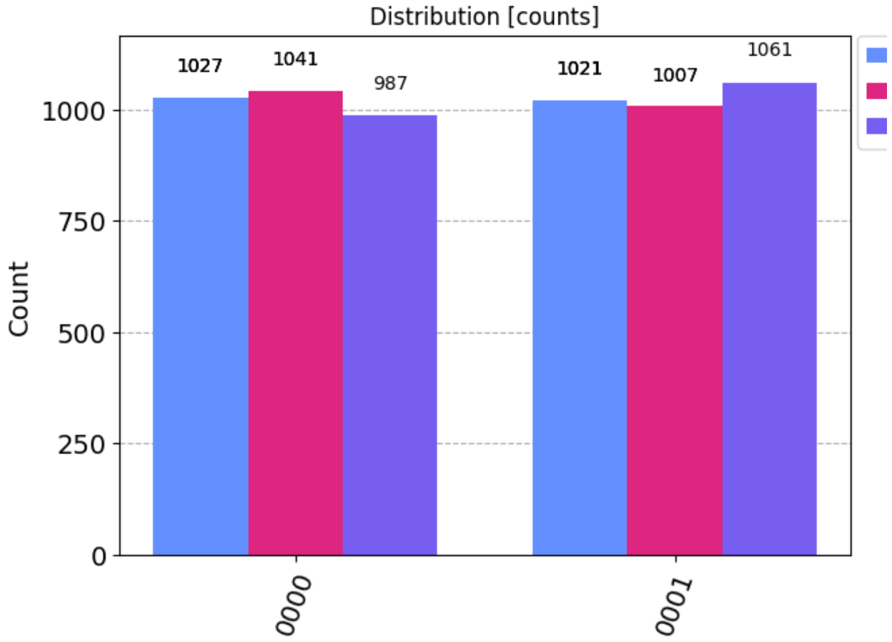
OpenQASM 2.0
Open in Quantum Lab

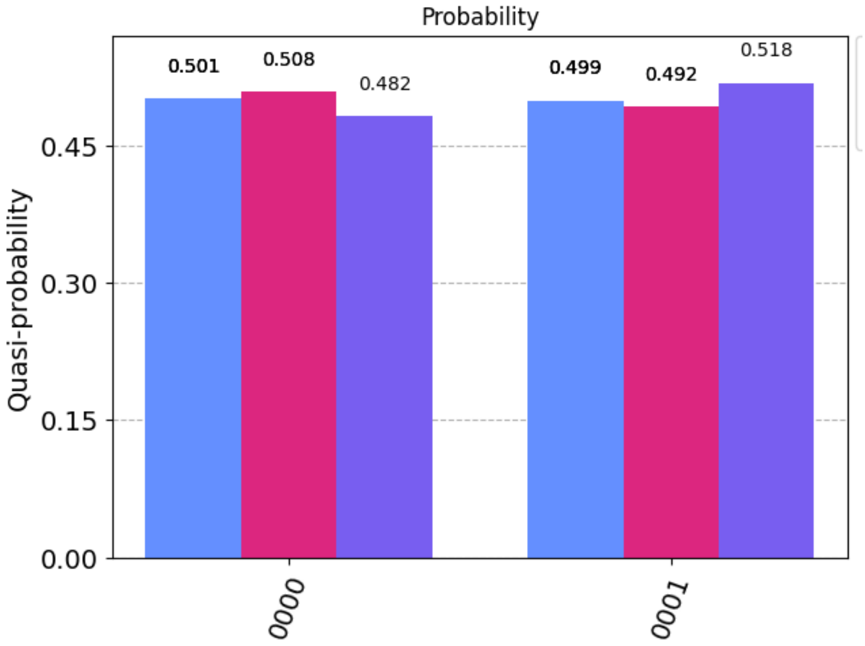
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[4];
creg c[4];
ry(pi/2) q[0];
p(pi/2) q[0];
h q[0];
measure q[0] -> c[0];
```

code (OpenQASM 2.0)

Classic measurement simulation – Qiskit & Python







Qiskit output show
the quantum circuit

Qiskit output results as Plot 1 „Counts”

Plot 2 „Probability”

Measurement in the Y base

$$\sigma_2 = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

QUANTUM COMPOSER CODE
Graphical code

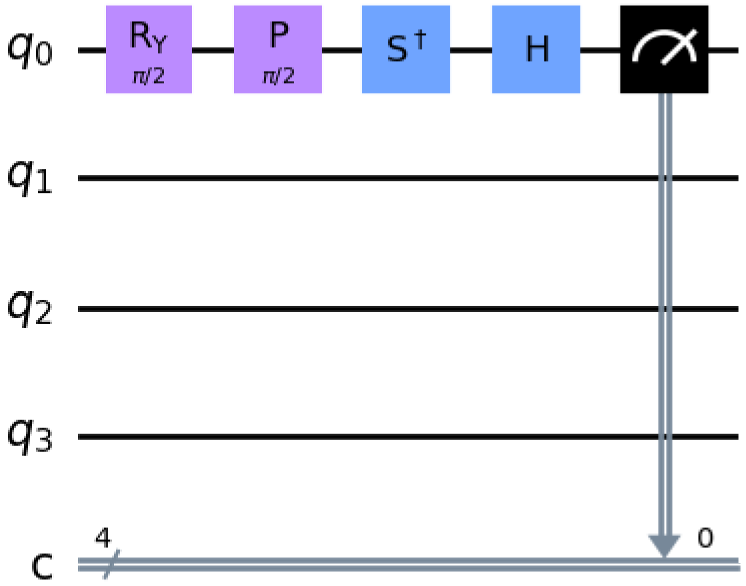
```
Qiskit                                   Read only
Open in Quantum Lab

from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.ry(pi / 2, qreg_q[0])
circuit.p(pi / 2, qreg_q[0])
circuit.sdg(qreg_q[0])
circuit.h(qreg_q[0])
circuit.measure(qreg_q[0], creg_c[0])
```

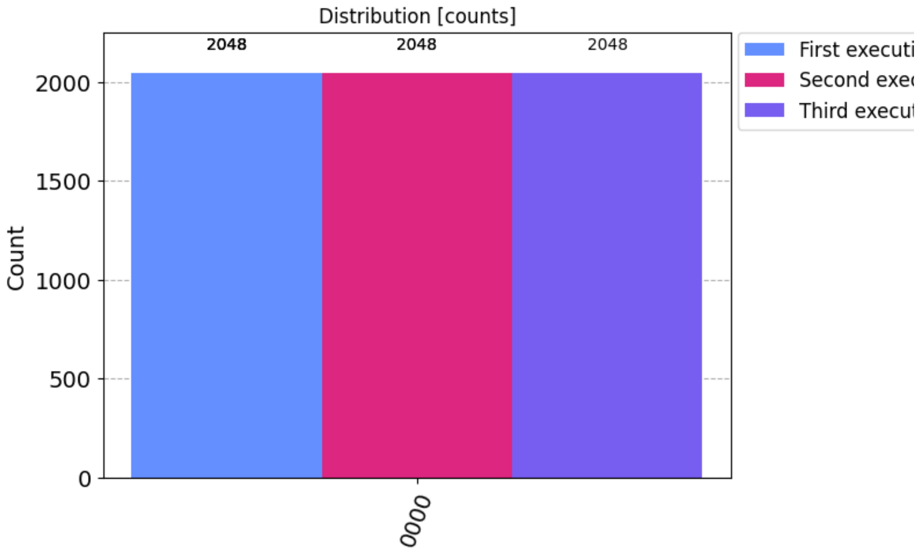
code (Qiskit)

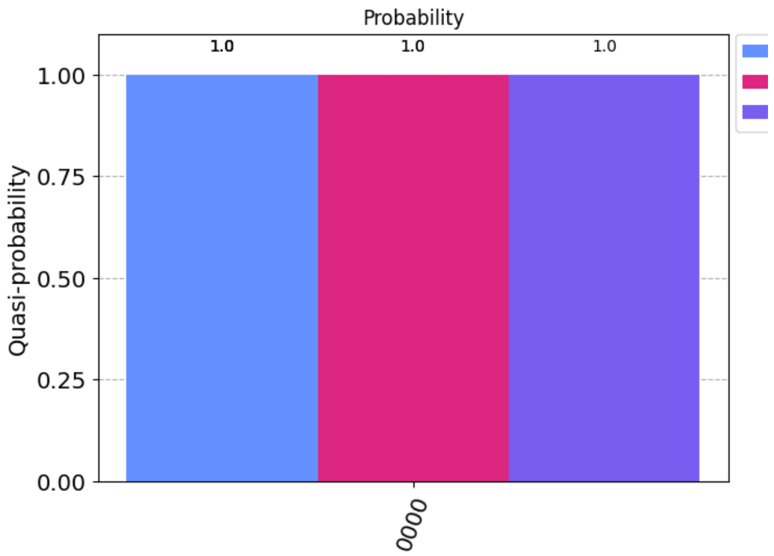
```
OpenQASM 2.0
Open in Quantum Lab

OPENQASM 2.0;
include "qelib1.inc";

qreg q[4];
creg c[4];
ry(pi/2) q[0];
p(pi/2) q[0];
sdg q[0];
h q[0];
measure q[0] -> c[0];
```

code (OpenQASM 2.0)

Classic measurement simulation – Qiskit and Python







Qiskit output show
the quantum circuit.

Qiskit output results as Plot 1 „Counts”

Plot 2 „Probability”

Measurement in the Z base

$$\sigma_3 = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

QUANTUM COMPOSER CODE
Graphical code

Qiskit

Read only

Open in Quantum Lab

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.ry(pi / 2, qreg_q[0])
circuit.p(pi / 2, qreg_q[0])
circuit.measure(qreg_q[0], creg_c[0])
```

code (Qiskit)

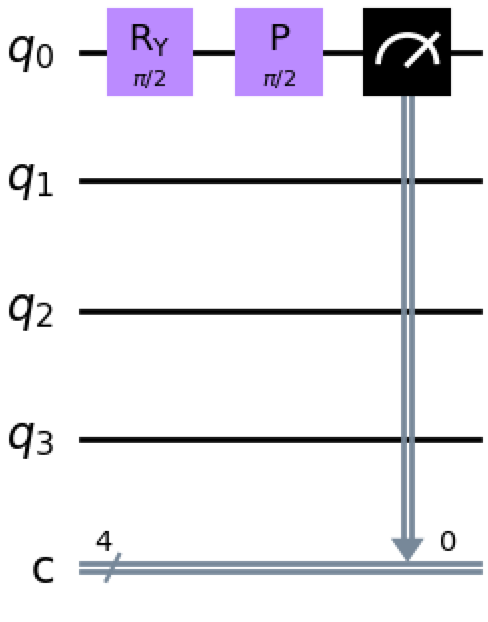
OpenQASM 2.0

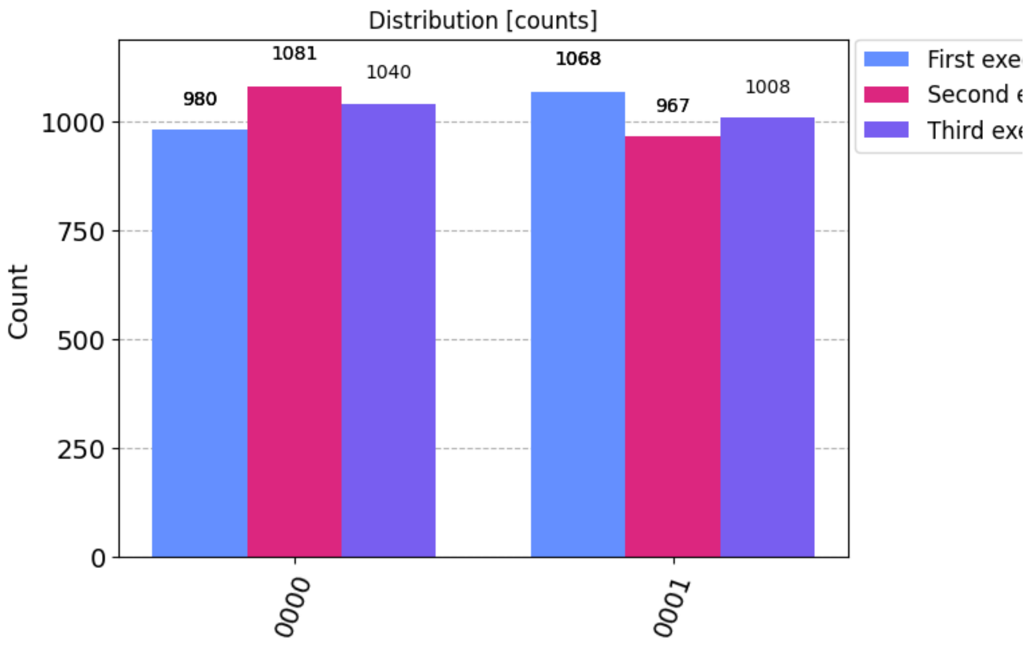
Open in Quantum Lab

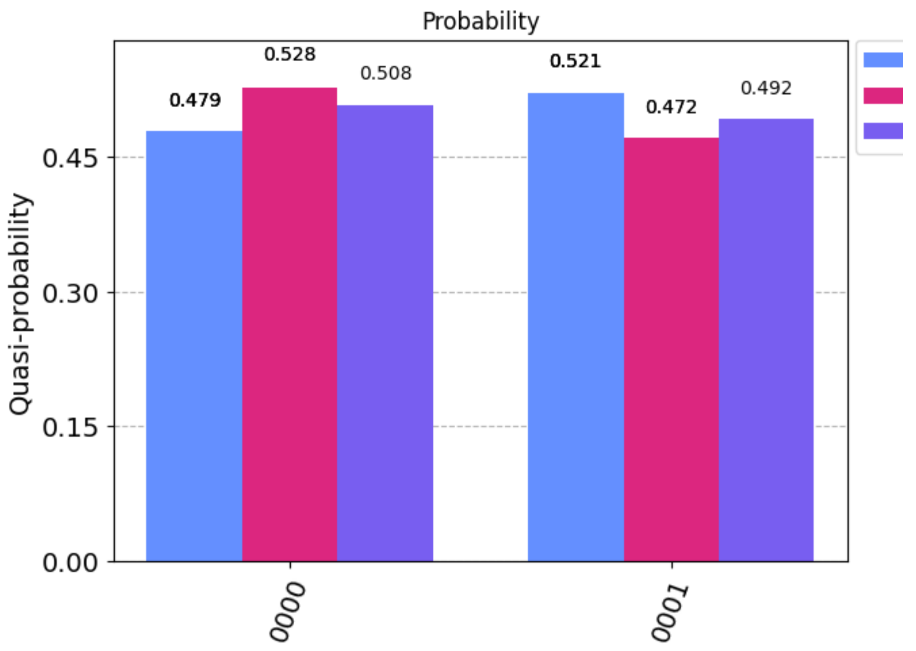
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[4];
creg c[4];
ry(pi/2) q[0];
p(pi/2) q[0];
measure q[0] -> c[0];
```

code (OpenQASM 2.0)

Classic measurement simulation – Qiskit & Python







Qiskit output show the quantum circuit

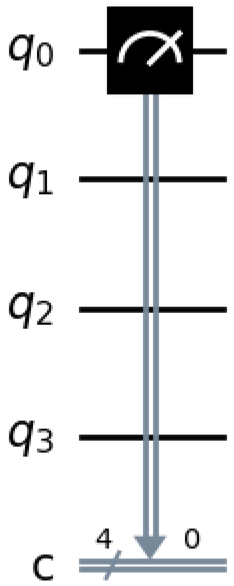
Qiskit output results as Plot 1 „Counts"

Plot 2 „Probability"

For ambitious and interested students

Additional tasks to complete

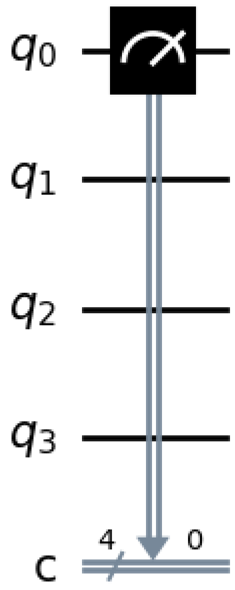
Visualization in Qiskit – quantum circuit (task 1)



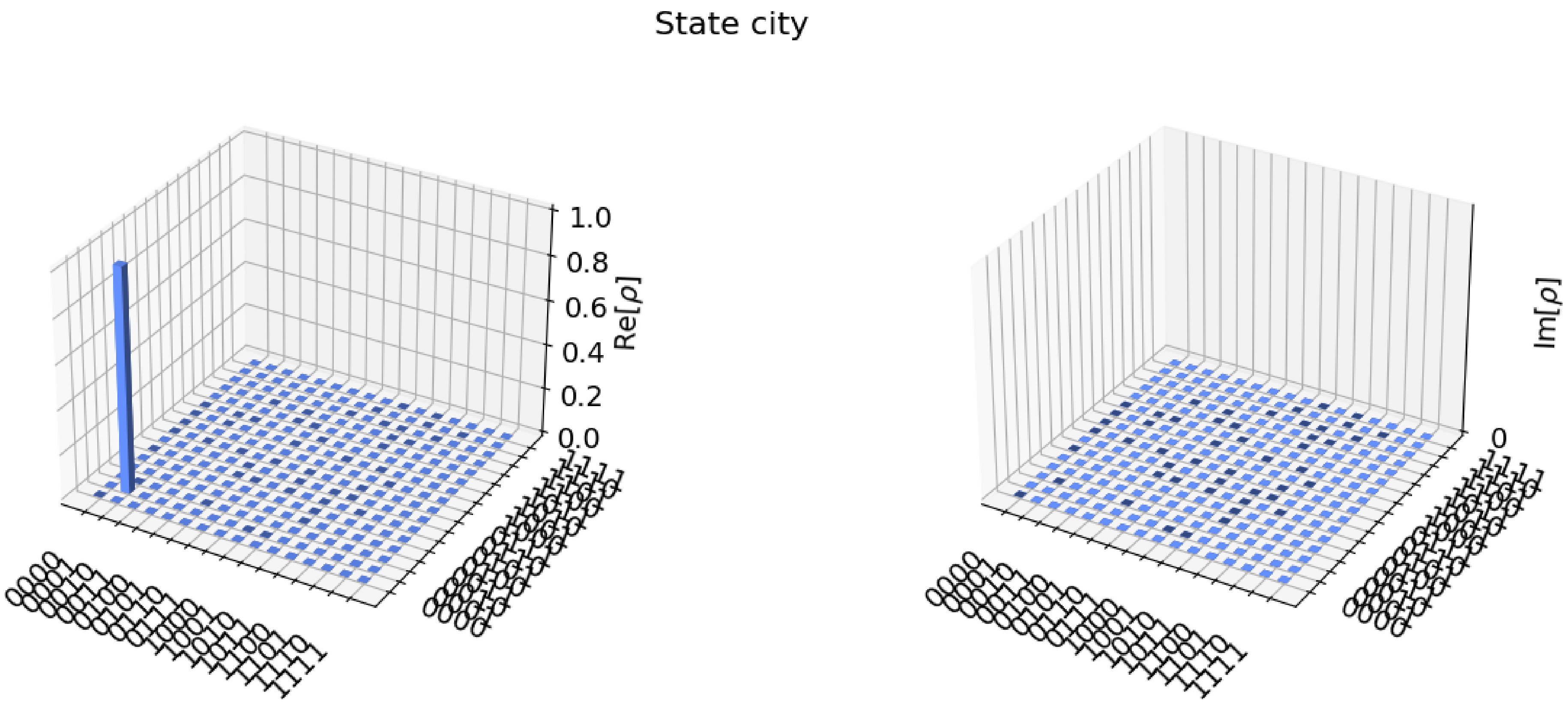

```
from qiskit.visualization import plot_state_city, plot_bloch_multivector
from qiskit.visualization import plot_state_paulivec, plot_state_hinton
from qiskit.visualization import plot_state_qsphere
# execute the quantum circuit
backend = BasicAer.get_backend('statevector_simulator') # the device to run on
result = backend.run(transpile(circuit, backend)).result()
psi = result.get_statevector(circuit)
```

Qiskit output show
the quantum circuit

Visualization in Qiskit: plot_state_city(psi)

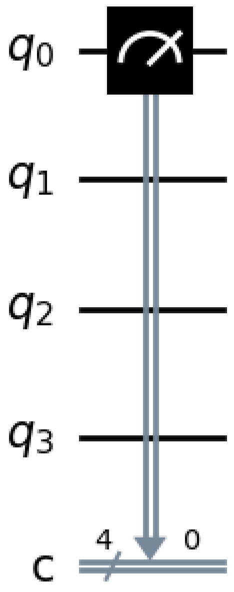


Qiskit output show
the quantum circuit

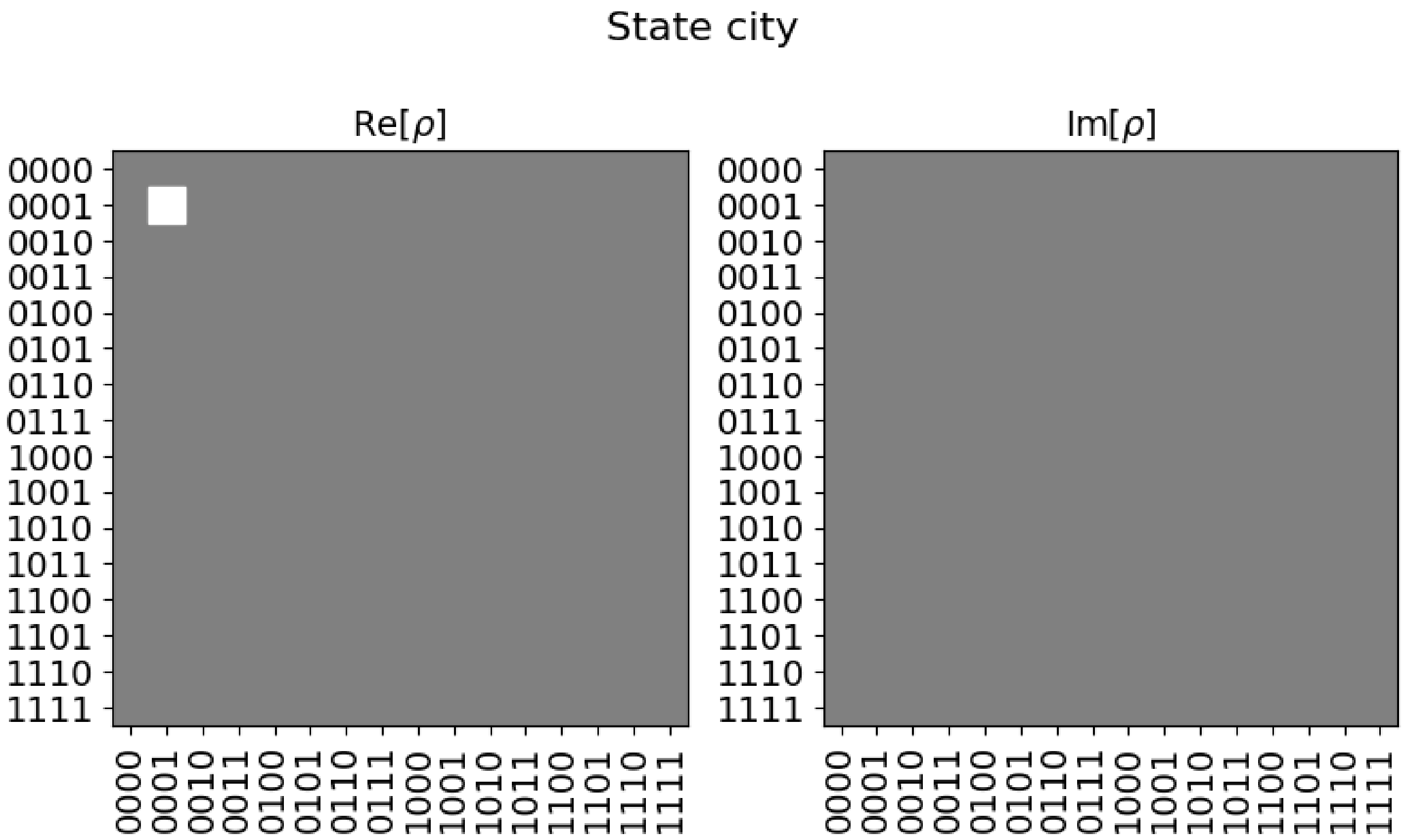


Plot after use state_city function for Quantum Circuit

Visualization in Qiskit: plot_state_hinton(psi)

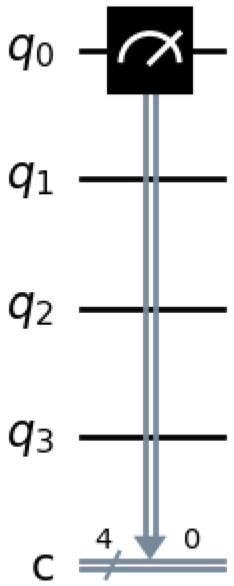


Qiskit output show the quantum circuit

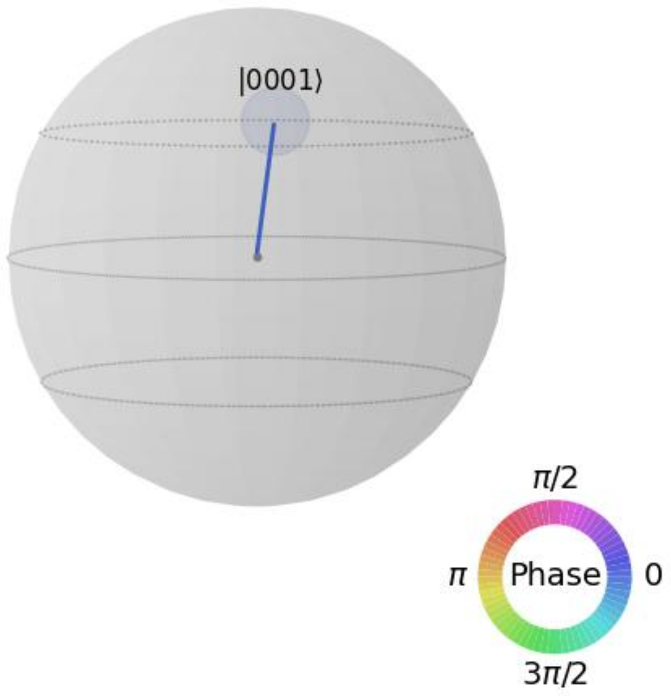


Plot after use state_hinton function for Quantum Circuit

Visualization in Qiskit: plot_state_qsphere(psi)

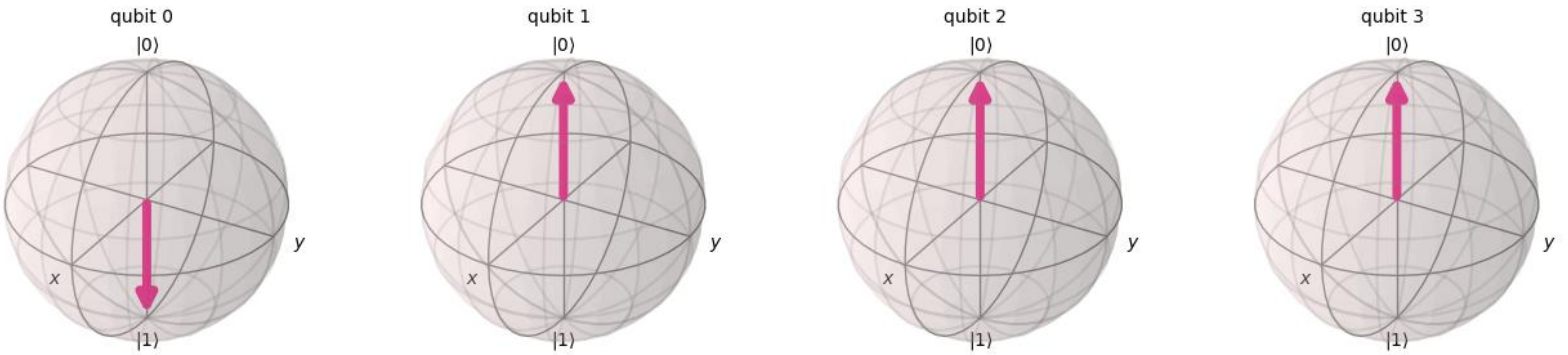


Qiskit output show
the quantum circuit



Plot after use state_qsphere function for Quantum Circuit

Visualization in Qiskit: plot_bloch_multivector(psi)



Plot after use state_multivector function for Quantum Circuit

The End