Qiskit 2 - Various measurements

In [4]:

```python
from qiskit import (
    ClassicalRegister,
    QuantumRegister,
    QuantumCircuit,
    Aer,
    execute
    )

from qiskit.tools.visualization import circuit_drawer, plot_histogram
```

1 - measure a single qubit

- **In:** one qubit, initialized as 0
- **Operations:** None
- **Hypothesis:** always measure a 0

In [3]:

```python
s1 = QuantumRegister(1, name='s1')
r1 = ClassicalRegister(1, name='r1')
c1 = QuantumCircuit(s1, r1)
c1.measure(s1, r1)
circuit_drawer(c1)
```

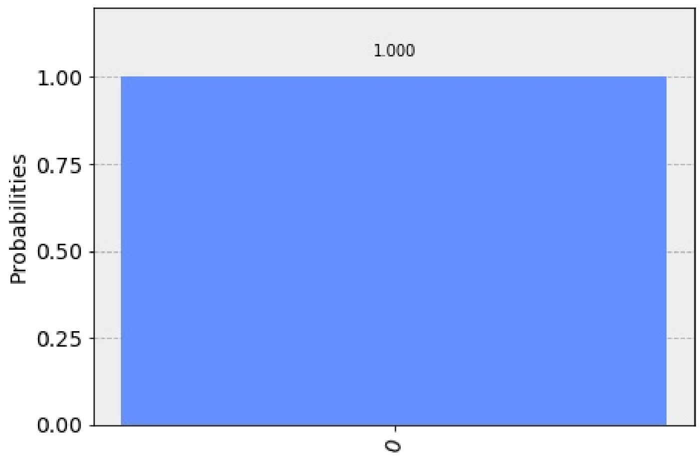
Out[3]:

```
s1_0: |0>┤M├
          ╥
 r1_0: 0 ═╩═
```

In [6]:

```python
sim = Aer.get_backend('qasm_simulator')
job1 = execute(c1, sim, shots=1000)
res1 = job1.result()
count1 = res1.get_counts(c1)
plot_histogram(count1)
```

Out[6]:



2 - swap from 0 to 1

- **In:** one qubit, initialized as 0
- **Operations:** apply a pauli x operator and perform a bitflip
- **Hypothesis:** measure the qubit and find it as 1 all the times, besides errors ofc

In [21]:

```
c2 = QuantumCircuit(s1, r1)
c2.x(s1)
c2.measure(s1, r1)
circuit_drawer(c2)
```

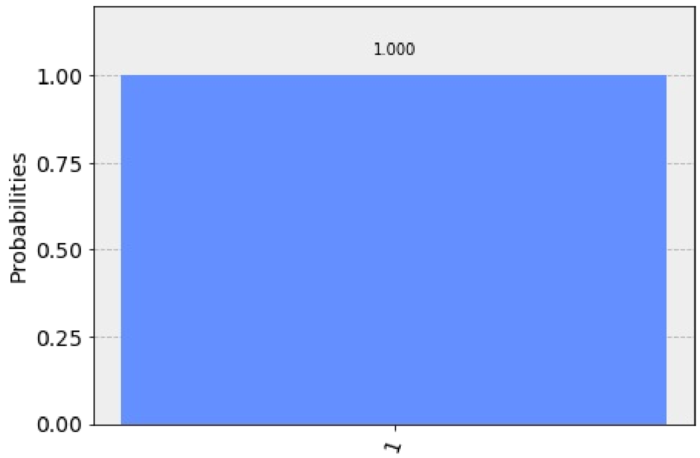
Out[21]:

```
        ┌───┐┌─┐
s1_0: |0>┤ X ├┤M├
        └───┘└╥┘
 r1_0: 0 ═════╩═
```

In [22]:

```
job2 = execute(c2, sim, shots=1000)
res2 = job2.result()
count2 = res2.get_counts()
plot_histogram(count2)
```

Out[22]:



3 - hadamard on input

- **In:** one qubit, initialized as 0
- **Operations:** apply a hadamard
- **Hypothesis:** measure the QuantumRegister and find it half in 0 and half in 1

In [23]:

```
c3 = QuantumCircuit(s1, r1)
c3.h(s1)
c3.measure(s1, r1)
circuit_drawer(c3)
```

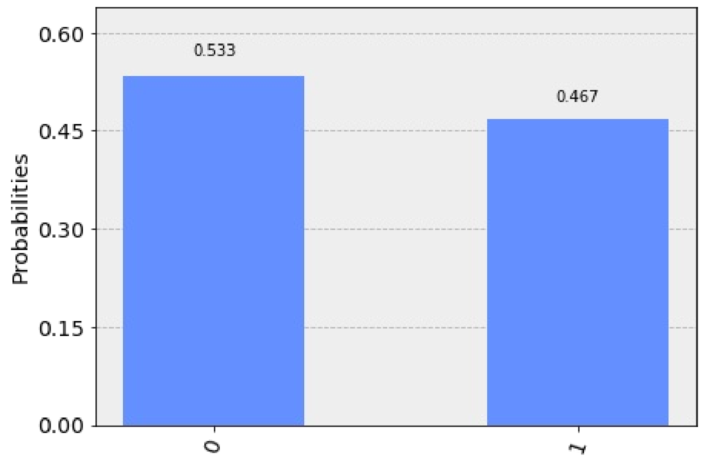
Out[23]:

```
        ┌───┐┌─┐
s1_0: |0>┤ H ├┤M├
        └───┘└╥┘
 r1_0: 0 ═════╩═
```

In [30]:

```
job3 = execute(c3, sim, shots=1000)
count3 = job3.result().get_counts()
plot_histogram(count3)
```

Out[30]:



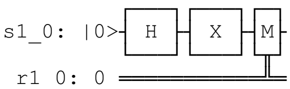
4 - hadamard and pauli x

- **In:** one qubit, initialized as 0
- **Operations:** apply hadamard and then a pauli x operator
- **Hypothesis:** measurements will be same as above

In [27]:

```
c4 = QuantumCircuit(s1, r1)
c4.h(s1)
c4.x(s1)
c4.measure(s1, r1)
circuit_drawer(c4)
```

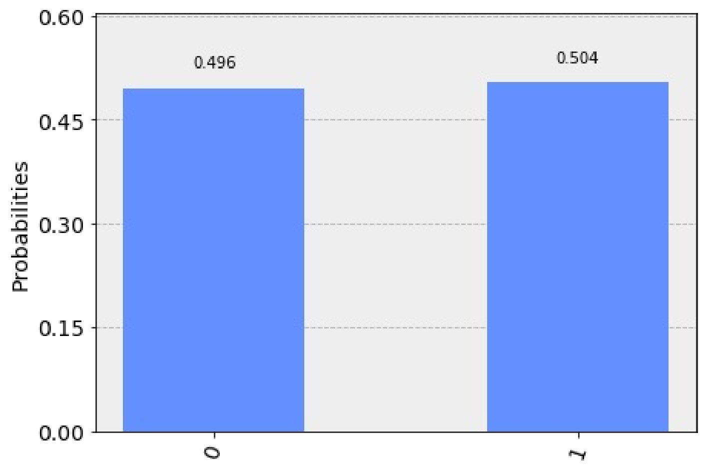
Out[27]:



In [31]:

```
job4 = execute(c4, sim)
plot_histogram(job4.result().get_counts())
```

Out[31]:



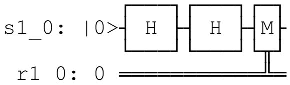
5 - double Hadamards

- **In:** one qubit, initialized as 0
- **Operations:** apply Hadamard and then another hadamard
- **Hypothesis:** we will always measure a zero

In [32]:

```
c5 = QuantumCircuit(s1, r1)
c5.h(s1)
c5.h(s1)
c5.measure(s1, r1)
circuit_drawer(c5)
```

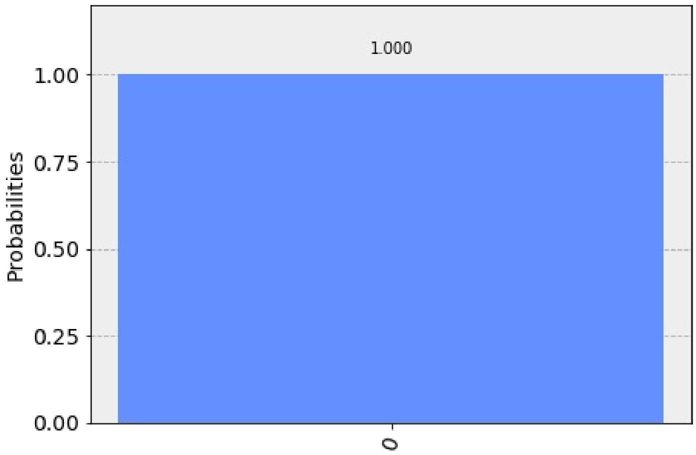
Out[32]:



In [34]:

```
job5 = execute(c5, sim, shots=1000)
plot_histogram(job5.result().get_counts())
```

Out[34]:



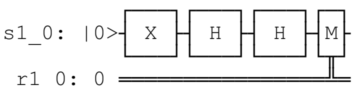
6 - double hadamards after flip

- **In:** one qubit, initialized as 0
- **Operations:** apply a pauli x and flip to 1, then double hadamards
- **Hypothesis:** always measure a 1

In [35]:

```
c6 = QuantumCircuit(s1, r1)
c6.x(s1)
c6.h(s1)
c6.h(s1)
c6.measure(s1, r1)
circuit_drawer(c6)
```

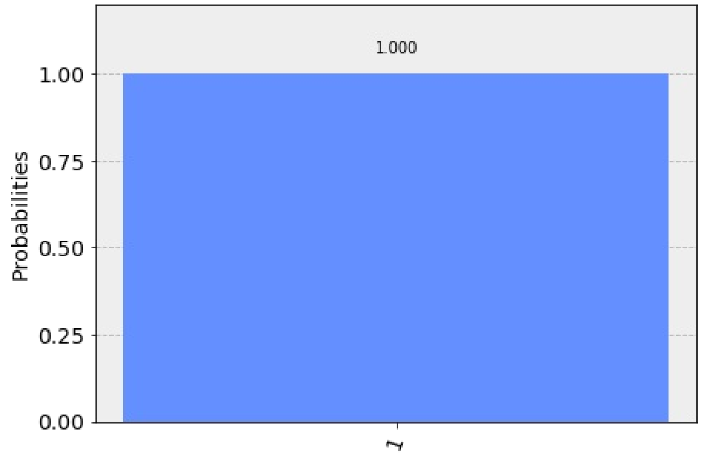
Out[35]:



In [36]:

```
job6 = execute(c6, sim)
res6 = job6.result()
count6 = res6.get_counts()
plot_histogram(count6)
```

Out[36]:



7 - bloch plot

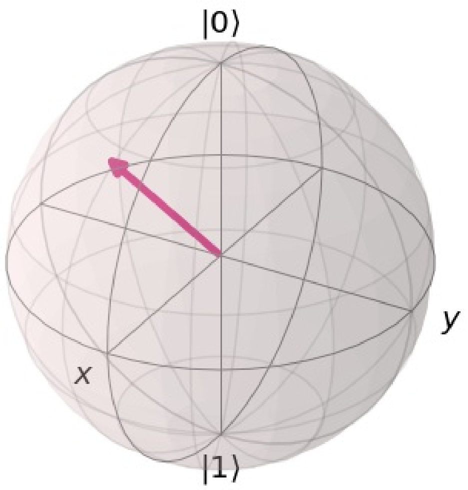
In [38]:

```
from qiskit.tools.visualization import plot_bloch_vector
```

In [44]:

```
plot_bloch_vector([1, 0, 1])
```

Out[44]:



8 - multiple results in histogram

In [45]:

```
cm = QuantumCircuit(s1, r1)
cm.h(s1)
cm.x(s1)
cm.measure(s1, r1)
circuit_drawer(cm)
```

Out[45]:

```
s1_0: |0>─┤ H ├─┤ X ├─┤M├─
                       ║
 r1_0: 0 ══════════════╩══
```

In [47]:

```
jobm1 = execute(cm, sim)
jobm2 = execute(cm, sim)
resm1 = jobm1.result().get_counts()
resm2 = jobm2.result().get_counts()
plot_histogram([resm1, resm2], legend=['first run', 'second run'])
```

Out[47]: