In [1]:
```
backend_publicism = Public_Provider.get_backend('ibmq_qasm_simulator')
backend_publiclon = Public_Provider.get_backend('ibmq_ourense')
print(backend_publicism)
print(backend_publiclon)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-8b5f7c7096cd> in <module>
----> 1 backend_publicism = Public_Provider.get_backend('ibmq_qasm_simulator')
      2 backend_publiclon = Public_Provider.get_backend('ibmq_ourense')
      3 print(backend_publicism)
      4 print(backend_publiclon)
      5

NameError: name 'Public_Provider' is not defined
```

In [2]:
```
from qiskit import IBMQ
API_TOKEN ='bf24308d4b9628ea5d8f4a21341653f23eb280986828dbe5d3b691a9a9cd0b40b0b40a2e7741cc22abc447b4feb2ba88a4d51f4d2512c7bca3aa964ef6b1a2c'
IBMQ.save_account(API_TOKEN)

Public_Provider = IBMQ.load_account()
Public_Provider.backends()
```

```
configrc.store_credentials:WARNING:2020-09-10 19:25:30,250: Credentials already present. Set overwrite=True to overwrite.
```

Out[2]:
```
[<IBMQSimulator('ibmq_qasm_simulator') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmqx2') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_16_melbourne') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_vigo') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_ourense') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_valencia') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_london') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_burlington') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_essex') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_armonk') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_santiago') from IBMQ(hub='ibm-q', group='open', project='main')>]
```

In [3]:
```
backend_publicism = Public_Provider.get_backend('ibmq_qasm_simulator')
backend_publiclon = Public_Provider.get_backend('ibmq_ourense')
print(backend_publicism)
print(backend_publiclon)
```

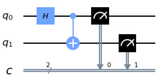
```
ibmq_qasm_simulator
ibmq_ourense
```

In [4]:
```
from qiskit import QuantumRegister, QuantumCircuit, ClassicalRegister
%matplotlib inline
```

In [5]:
```
q = QuantumRegister(2, name = 'q')
c = ClassicalRegister(2, name = 'c')
bell_state = QuantumCircuit(q,c)
```

In [6]:
```
bell_state.h(q[0])
bell_state.cx(q[0], q[1])
bell_state.measure(q,c)
bell_state.draw(output = 'mpl')
```

Out[6]:



In [7]:
```
from qiskit import BasicAer, execute
```

In [8]:
```
from qiskit.tools.monitor import job_monitor
```

In [9]:
```
job = execute(bell_state,backend_publiclon)
```

In [10]:
```
job_monitor(job)
```

```
Job Status: job has successfully run
```

In [11]:
```
result = job.result()
```

In [12]:
```
count = result.get_counts()
```

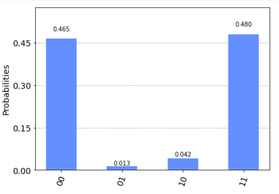
In [13]:
```
print(count)
```

```
{'01': 13, '11': 492, '10': 43, '00': 476}
```

In [14]:
```
from qiskit.tools.visualization import plot_histogram
```

In [15]:
```
plot_histogram(count)
```

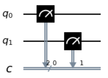
Out[15]:



In [21]:
```
q2 = QuantumRegister(2, name = 'q')
c2 = ClassicalRegister(2, name = 'c')
Entangle = QuantumCircuit(q,c)
```

In [22]:
```
Entangle.measure(q2,c2)
Entangle.draw(output = 'mpl')
```

Out[22]:



In [23]:
```
job = execute(Entangle,backend_publiclon)
```

In [24]:
```
job_monitor(job)
```

```
Job Status: job has successfully run
```

In [25]:
```
result = job.result()
```

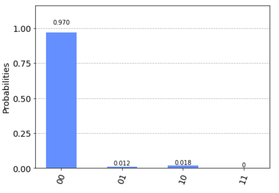
In [26]:
```
count = result.get_counts()
```

In [27]:
```
print(count)
```

```
{'01': 12, '11': 1, '10': 18, '00': 993}
```

In [28]:
```
plot_histogram(count)
```

Out[28]:



In [29]:
```
import numpy as np
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from qiskit.extensions import Initialize
from qiskit_textbook.tools import random_state, array_to_latex
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-29-b4b7e1c61d6c> in <module>
      2 from qiskit.visualization import plot_histogram, plot_bloch_multivector
      3 from qiskit.extensions import Initialize
----> 4 from qiskit_textbook.tools import random_state, array_to_latex

ModuleNotFoundError: No module named 'qiskit_textbook'
```

In [30]:
```
import numpy as np
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from qiskit.extensions import Initialize
```

In [31]:
```
qr = QuantumRegister(3, name="q")
crz = ClassicalRegister(1, name="crz")
crx = ClassicalRegister(1, name="crx")
teleportation_circuit = QuantumCircuit(qr, crz, crx)
```

In [32]:
```
teleportation_circuit.h(q[1])
teleportation_circuit.cx(q[1], q[2])
bell_state.draw(output = 'mpl')
```

```
---------------------------------------------------------------------------
CircuitError                              Traceback (most recent call last)
<ipython-input-32-5392d188c892> in <module>
----> 1 teleportation_circuit.h(q[1])
      2 teleportation_circuit.cx(q[1], q[2])
      3 bell_state.draw(output = 'mpl')

/opt/anaconda3/envs/venv/lib/python3.8/site-packages/qiskit/util.py in wrapper(*args, **kwargs)
    107             if kwargs:
    108                 _rename_kwargs(func.__name__, kwargs, kwarg_map)
--> 109             return func(*args, **kwargs)
    110         return wrapper
    111     return decorator

/opt/anaconda3/envs/venv/lib/python3.8/site-packages/qiskit/circuit/quantumcircuit.py in h(self, qubit, q)
   1582         """Apply :class:`~qiskit.circuit.library.HGate`."""
   1583         from .library.standard_gates.h import HGate
-> 1584         return self.append(HGate(), [qubit], [])
   1585
   1586     @deprecate_arguments({'ctl': 'control_qubit', 'tgt': 'target_qubit'})

/opt/anaconda3/envs/venv/lib/python3.8/site-packages/qiskit/circuit/quantumcircuit.py in append(self, instruction, qargs, cargs)
    544         instructions = InstructionSet()
    545         for (qarg, carg) in instruction.broadcast_arguments(expanded_qargs, expanded_cargs):
--> 546             instructions.add(self._append(instruction, qarg, carg), qarg, carg)
    547         return instructions
    548

/opt/anaconda3/envs/venv/lib/python3.8/site-packages/qiskit/circuit/quantumcircuit.py in _append(self, instruction, qargs, cargs)
    568         # do some compatibility checks
    569         self._check_dups(qargs)
--> 570         self._check_qargs(qargs)
    571         self._check_cargs(cargs)
    572

/opt/anaconda3/envs/venv/lib/python3.8/site-packages/qiskit/circuit/quantumcircuit.py in _check_qargs(self, qargs)
    643             raise CircuitError("qarg is not a Qubit")
    644         if not all(self.has_register(i.register) for i in qargs):
--> 645             raise CircuitError("register not in this circuit")
    646
    647     def _check_cargs(self, cargs):

CircuitError: 'register not in this circuit'
```

In [ ]:
```
```