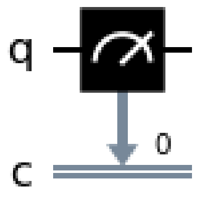
Task 2 - Drawing Circuits (20 pts) We can visualize circuits using the QuantumCircuit's draw method. Draw your circuit from Task 1 using the matplotlib format.

```
In [7]: # Draw your circuit in this cell

        # BEGIN SOLUTION
        simpleCircuit().draw(output='mpl')
        # END SOLUTION
```

```
Out[7]:
```



Task 4 - Running Your Circuit on a Quantum Computer (20 pts) Now let's compare our results from the simulator with the results from a real quantum device. *(What should we expect to see?)*

Create an account with IBM Quantum and paste your API token into the code block below. After running the `save_account` method, you may remove your token to keep it private from Gradescope. Credentials will be saved to your computer and calling `load_account` is sufficient to retrieve them.

```
In [10]: # IBMQ.save_account('replace with your token and uncomment the first time')
```

```
In [11]: IBMQ.load_account()
```

```
Out[11]: <AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>
```

The code block below lists some info about the available IBM quantum devices and queues.

```
In [12]: provider = IBMQ.get_provider(hub='ibm-q')
         for backend in provider.backends():
             status = backend.status().to_dict()
             if status['operational'] and status['status_msg'] == 'active':
                 if 'simulator' not in status['backend_name']:
                     print(pprint.pformat(status))
```

```
{'backend_name': 'ibmq_armonk',
 'backend_version': '2.4.26',
 'operational': True,
 'pending_jobs': 1,
 'status_msg': 'active'}
{'backend_name': 'ibmq_bogota',
 'backend_version': '1.6.16',
 'operational': True,
 'pending_jobs': 38,
 'status_msg': 'active'}
{'backend_name': 'ibmq_lima',
 'backend_version': '1.0.27',
 'operational': True,
 'pending_jobs': 29,
 'status_msg': 'active'}
{'backend_name': 'ibmq_belem',
 'backend_version': '1.0.30',
 'operational': True,
 'pending_jobs': 8,
 'status_msg': 'active'}
{'backend_name': 'ibmq_quito',
 'backend_version': '1.1.20',
 'operational': True,
 'pending_jobs': 72,
 'status_msg': 'active'}
{'backend_name': 'ibmq_manila',
 'backend_version': '1.0.22',
 'operational': True,
 'pending_jobs': 122,
 'status_msg': 'active'}
```

Choose one of the backends from above and insert its name into the code block below. Running this code block will execuete your circuit on an IBM quantum device. **Note: It may take a while for your job to complete based on queue times.** Use the generated link to check your job's status.

```
In [13]: ibmqc = provider.get_backend('replace with a backend_name')
         job = execute(simpleCircuit(), ibmqc, shots=468)
         print("Check job status here:", "https://quantum-computing.ibm.com/jobs/" + job.job_id())
         res = job.result()
         counts = res.get_counts()
         plot_histogram(counts)
```

```
---------------------------------------------------------------------------
QiskitBackendNotFoundError                Traceback (most recent call last)
<ipython-input-13-07ee3753e4aa> in <module>
----> 1 ibmqc = provider.get_backend('replace with a backend_name')
      2 job = execute(simpleCircuit(), ibmqc, shots=468)
      3 print("Check job status here:", "https://quantum-computing.ibm.com/jobs/" + job job_id())
      4 res = job.result()
      5 counts = res.get_counts()

~/opt/miniconda3/envs/cse468/lib/python3.7/site-packages/qiskit/providers/provider.py i. get_backend(s
     53             raise QiskitBackendNotFoundError("More than one backend matches the criteria")
     54         if not backends:
---> 55             raise QiskitBackendNotFoundError("No backend matches the criteria")
     56 
     57         return backends[0]

QiskitBackendNotFoundError: 'No backend matches the criteria'
```

Do you see the same results as the qasm simulator? Why or why not?

Type your answer here, replacing this text.

No. There is error induced by the quantum computer which causes some measurements to yield $|1\rangle$.