Hands-on Labs:

Running Qiskit Programs on IBMQ

Introduction

This hands-on lab provides step-by-step instructions for running quantum circuits using Qiskit on IBM Quantum Experience (IBMQ). By the end of this lab, students will be able to:

- Set up an IBMQ account and access quantum processors.
- Write and execute quantum circuits using Qiskit.
- Submit jobs to real quantum hardware and analyze results.

---

1. Setting Up IBMQ

Step 1: Create an IBM Quantum Experience Account

1. Go to IBM Quantum Experience.
2. Click on **Sign Up** and create an account.
3. Verify your email and log in to the dashboard.
4. Navigate to **API Tokens** under Account Settings.
5. Copy your API token for later use.

Step 2: Install Qiskit

Before proceeding, ensure that Python and Qiskit are installed. Open a terminal or command prompt and run:

```
pip install qiskit
pip install qiskit-ibm-runtime
```

Step 3: Authenticate with IBMQ

1. Open a Python script or Jupyter Notebook.
2. Run the following code to save your API token:

```
from qiskit import IBMQ
IBMQ.save_account('YOUR_API_TOKEN')
IBMQ.load_account()
```

3. This step enables access to IBMQ’s quantum processors.

---

2. Writing and Running a Quantum Circuit

Step 4: Create a Simple Quantum Circuit

Open a Python script or Jupyter Notebook and enter:

```
from qiskit import QuantumCircuit, transpile, assemble, Aer, execute
from qiskit.providers.ibmq import IBMQ

# Load IBMQ account
IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q')
backend = provider.get_backend('ibmq_lima')

# Create a simple quantum circuit
qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0, 1)
qc.measure([0,1], [0,1])

# Transpile and execute the circuit
transpiled_qc = transpile(qc, backend)
job = backend.run(assemble(transpiled_qc))
print("Job ID:", job.job_id())
```

Step 5: Monitor Job Status

After submitting the job, you can check its status:

```
from qiskit.tools.monitor import job_monitor
job_monitor(job)
```

Step 6: Retrieve and Analyze Results

Once the job is complete, retrieve results and visualize output:

```
result = job.result()
counts = result.get_counts()
print("Measurement Results:", counts)

from qiskit.visualization import plot_histogram
```

```
plot_histogram(counts)
```

---

3. Submitting Custom Circuits

Step 7: Modify and Submit a Custom Circuit

1. Modify the circuit by adding different quantum gates:

```
qc = QuantumCircuit(3, 3)
qc.h(0)
qc.cx(0, 1)
qc.cx(1, 2)
qc.measure([0,1,2], [0,1,2])
```

2. Re-run the execution steps above.

---

4. Summary and Further Exploration

- Experiment with different quantum gates and circuit depths.
- Try executing circuits on different IBMQ backends.
- Explore Qiskit's advanced features, such as noise models and pulse programming.

This lab provides a practical foundation for running quantum programs on IBMQ. Future labs will explore more complex quantum algorithms and hybrid quantum-classical workflows.