# Quantum Programming Languages: Qiskit

Jens Palsberg

May 12, 2020

# Outline

**Hook:** The next quantum programming homework is in Qiskit. Qiskit is a mix of Python and of calls to operations on a quantum machine. The homework is to program and run four well-known algorithms.

**Purpose:** Persuade you that you can get this to work.

**Preview:**

1. We will go over the text of the homework.
2. We will show how to get Qiskit to run on a laptop.
3. We will go over full details of how to write a program in Qiskit.

**Transition to Body:** Let us begin with the task that is due in two weeks.

**Main Point 1:** We will go over the text of the homework.
[Design: $U_f$, parameterization in $n$, and sharing of code]
[Evaluation: testing and scalability]
[Instructions: how to input, how to run, and how to understand the output]

**Transition to MP2:** Now let us look at how how to get started.

**Main Point 2:** We will show how to get Qiskit to run on a laptop.
[Installation]
[Run a Qiskit program]
[Run on IBM quantum computers]

**Transition to MP3:** Now let us get into what programs look like.

**Main Point 3:** We will go over full details of how to write a program in Qiskit.
[Import]
[A program is a sequence of instructions, each of which addresses one or more qubits]
[We can measure the qubits, print the results, and repeat runs]

**Transition to Close:** So there you have it.

**Review:** The homework has both programming and report writing, the language will run on your laptop, and the programs look roughly as one could expect after reading a circuit diagram.

**Strong finish:** Qiskit is an attempt to make quantum programming as simple as possible, but not simpler. (Thank you, Mr. Einstein!)

**Call to action:** While you do this assignment, think about the core ideas that may carry over when we move on to the next language.

## Detailed presentation

**Hook:** The next quantum programming homework is in Qiskit. Qiskit is a mix of Python and of calls to operations on a quantum machine. The homework is to program and run four well-known algorithms.

**Purpose:** Persuade you that you can get this to work.

**Preview:**

1. We will go over the text of the homework.
2. We will show how to get Qiskit to run on a laptop.
3. We will go over full details of how to write a program in Qiskit.

**Transition to Body:** Let us begin with the task that is due in two weeks.

**Main Point 1:** We will go over the text of the homework.

[Design: $U_f$, parameterization in $n$, and sharing of code]

In Qiskit, implement the Deutsch-Jozsa algorithm, the Bernstein-Vazirani algorithm, Simon's algorithm, and Grover's algorithm. Write detailed comments in the code about why it works. Run the programs on the simulator. Write a report that covers the following three points.

The first point is about design and evaluation.

- Present the design of how you implemented the black-box function $U_f$. Assess how visually neat and easy to read it is.
- Present the design for how you prevent the user of $U_f$ from accessing the implementation of $U_f$. Assess how well you succeeded.
- Present the design of how you parameterized the solution in $n$.
- Discuss the number of lines and percentage of code that your four programs share. Assess how well you succeeded in reusing code from one program to the next.
- Discuss your effort to test the four programs and present results from the testing. Discuss whether different cases of $U_f$ lead to different execution times.
- What is your experience with scalability as $n$ grows? Present a diagram that maps n to execution time.

[Instructions: how to input, how to run, and how to understand the output]

- Present a README file that describes how to input the function $f$, how to run the program, and how to understand the output.

[Qiskit: reflections]

As a reflection on both homeworks on programming in Qiskit, address the following points.

- List three aspects of quantum programming in Qiskit that turned out to be easy to learn and list three aspects that were difficult to learn.

- List three aspects of quantum programming in Qiski that the language supports well and list three aspects that Qiskit supports poorly.
- Which feature would you like Qiskit to support to make the quantum programming easier?
- List three positives and three negatives of the documentation of Qiskit.
- In some cases, Qiskit has its own names for key concepts in quantum programming. Give a dictionary that maps key concepts in quantum programming to the names used in Qiskit.
- How much type checking does the Qiskit implementation do at run time when a program passes data from the classical side to the quantum side and back?

Submit five files, one for each program and one with the report.

**Transition to MP2:** Now let us look at how how to get started.

**Main Point 2:** We will show how to get Qiskit to run on a laptop.

[Installation]
Qiskit's pip package:

```
pip install qiskit
```

[Run a Qiskit program]
Now we can run Qiskit programs like any other Python program. Try one like the following (from Getting Started with Qiskit):

```
import numpy as np
from qiskit import(QuantumCircuit,execute,Aer)
from qiskit.visualization import plot_histogram
# Use Aers qasm_simulator
simulator = Aer.get_backend(qasm_simulator)
# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)
# Add a H gate on qubit 0
circuit.h(0)
# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)
# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])
# Execute the circuit on the qasm simulator
job = execute(circuit, simulator, shots=1000)
# Grab results from the job
result = job.result()
# Returns counts
counts = result.get_counts(circuit)
print("\nTotal count for 00 and 11 are:",counts)
```

[Run on IBM quantum computers]
One can run on the IBM quantum computers from any Python script. First, get your API token from the account page for your IBM Quantum Experience account. Then, in your Python code, connect and run on the quantum computer with the following:

```
from qiskit import IBMQ, compile
IBMQ.save_account(MY_API_TOKEN)
provider = IBMQ.load_account(MY_API_TOKEN)
backend = provider.get_backend(one_of_backends_from_provider.backends())
qobj = compile(your_quantum_circuit, backend, shots=num_shots)
job = backend.run(qobj)
result = job.result()
counts = result.get_counts()
```

Your job may be queued and the queue may be quite long. You can access your job's result later through the web portal.

**Transition to MP3:** Now let us get into what programs look like.

**Main Point 3:** We will go over full details of how to write a program in Qiskit.

[Import]
We will examine the program listed earlier. The program creates a fully entangled state between two qubits, called a Bell State. This state is in an equal superposition between $|00\rangle$ and $|11\rangle$, meaning that it is equally likely that a measurement will result in measuring both qubits as 0 or both qubits as 1.

```
import numpy as np
from qiskit import(QuantumCircuit,execute,Aer)
from qiskit.visualization import plot_histogram
# Use Aers qasm_simulator
simulator = Aer.get_backend(qasm_simulator)
```

[A program is a sequence of instructions, each of which addresses one or more qubits]
Next, let's construct our Bell State.

```
# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)
# Add a H gate on qubit 0
circuit.h(0)
# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)
# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])
```

We've accomplished this by driving qubit 0 into a superposition state (that's what the `h` gate does), and then creating an entangled state between qubits 0 and 1 (that's what the `cx` gate does).

[We can measure the qubits, print the results, and repeat runs]
Finally, we will run the program:

```
# Execute the circuit on the qasm simulator
job = execute(circuit, simulator, shots=1000)
# Grab results from the job
result = job.result()
# Returns counts
counts = result.get_counts(circuit)
print("\nTotal count for 00 and 11 are:",counts)
```

Compare the two arrays of measurement results. The results will be correlated between the qubits and random from shot to shot.

The `simulator` is a simulated quantum computer. By specifying we want to `execute`, we have told our simulator to run the program specified above, collapse the state with a measurement, and return the result to us. The variable `shots` refers to the number of times we run the whole program.

**Transition to Close:** So there you have it.

**Review:** The homework has both programming and report writing, the language will run on your laptop, and the programs look roughly as one could expect after reading a circuit diagram.

**Strong finish:** Qiskit is an attempt to make quantum programming as simple as possible, but not simpler. (Thank you, Mr. Einstein!)

**Call to action:** While you do this assignment, think about the core ideas that may carry over when we move on to other languages.