QISKIT CHEAT SHEET

Step 1: Creating Quantum Circuits	
Creating a quantum circuit with **1 qubit**	`qc = QuantumCircuit(1)`
Creating a quantum circuit with **n qubits**	`qc = QuantumCircuit(n)`
Creating a quantum circuit with **n qubits and m classical bits**	`qc = QuantumCircuit(n, m)`

Step 2: Quantum Gates	
X gate on qubit 0	`qc.x(0)`
X gate on qubit a (indexed by 0)	`qc.x(a)`
Z gate on qubit a (indexed by 0)	`qc.z(a)`
H gate on qubit a (indexed by 0)	`qc.h(a)`
CX gate with qubit a as the control and qubit b as the target	`qc.cx(a, b)`

Step 2: Measurement	
Measuring the state of qubit 0 and storing the result in classical bit 0	`qc.measure(0, 0)`
Measuring the states of qubits 0 and 1 and storing the results in classical bits 0 and 1	`qc.measure([0, 1], [0, 1]`
Measuring the states of qubits 0, 1, 2 and storing the results in classical bits 0, 1, 2	`qc.measure([0, 1, 2], [0, 1, 2]`
Measuring the states of qubits 0 on and storing the results in classical bits 0 on	`qc.measure([0, 1, 2, ...], [0, 1, 2, ...]`

Step 3: Visualizing and Simulating Circuits

Draw the circuit. If `idle_wires` is `False` then any extra/unused wires will be ignored, which is particularly helpful for transpiled circuits where extra wires may be created that are never actually used	`qc.draw(idle_wires = False)`
Visualize the rotation on the Bloch sphere. NOTE: This works for 1 qubit circuits only.	`visualize_transition(qc, trace = True)`

Simulating the quantum circuit with the QASM simulator. Specifically,
 	
`qc`: quantum circuit to run
`backend`: which backend to run the circuit on
`shots`: how many times to repeat and measure the circuit
	`# Simulate your specific circuit, make sure`
	`backend = Aer.get_backend('qasm_simulator')`
	`job = execute(qc, backend=backend, shots=1024)`
	
	`# Get the counts from the simulation result`
	`result = job.result()`
	`counts = result.get_counts()`
	
	`# Plot the results as a histogram`
	`plot_histogram(counts)`

Step 3: Running Circuits with any Backend

List of available backends	`backends = provider.backends()`
Get the number of jobs in line for a given backend (useful for finding a less busy backend). NOTE: This can be used for any backend, including the two listed below.	`backends[i].status().pending_jobs`
Get the QASM simulator as a backend	`backend = Aer.get_backend('qasm_simulator')`
Quantum hardware of given name as a backend	`backend = provider.get_backend('ibm_oslo')`
See how your intended circuit is transpiled on a given backend	`tc = transpile(qc, backend = backend)`
Send the circuit to be executed on whatever backend you specify	`job = execute(qc, backend=backend, shots=1024)`

Check the status of the job. This is important for running on quantum hardware that may take a while to get back, depending on the size of the queue. The main statuses of interest: QUEUED/'JOB IS QUEUED' and DONE/'JOB HAS SUCCESSFULLY RUN'. If the job is not DONE, then you need to give it more time.	`job.status()`

Once the job is DONE, the results can be unpacked and visualized	`# Get the results as counts of outcomes`
	`result = job.result()`
	`counts = result.get_counts()`
	
	`# Plot the results as a histogram`
	`plot_histogram(counts)`