IBM qiskit certification exam cheat sheet

Finq Tech
Quantum Lab

(1) Creating QuantumCircuit:
qr = QuantumRegister(3, 'q')
anc = QuantumRegister(1, 'ancilla')
cr = ClassicalRegister(3, 'c')
qc = QuantumCircuit(qr, anc, cr)
OR:
qc = QuantumCircuit(4, 3)

(2) 1-qubit gates:
(All rotation follow right hand rule)
qc.x([0,1]) or .y(), .z()
Rotate π by x,y,z axis
qc.rx(Θ) qc.ry(Θ), qc.rz(Θ)
Rotate Θ by axis.
p = rz = u1 are equivalent
s = rz(π /2), t = rz(π /4)
h Rotate π by axis (x + z)/$\sqrt{2}$
HZH = X, HXH = Z, $SXS^{\dagger} = Y$, $SYS^{\dagger} = X$
u3(Θ, ϕ, λ) = rz(λ) ry (Θ) rz(ϕ)
u2(ϕ, λ) = u3(Θ=π/2)

(3) Control gates:
qc.cz(ctrl, target). Or any other c-gates
qc.cx(0,1) = qc.cnot(0,1)
= qc.append(cx1, [0,1])
where cx1 = XGate().control()
ccx(0, 1, 2) = mct([0,1], 2)
mcx = mct = multi control x gate
qc.crx(theta, 0, 1) etc require theta as first input
ctrl_state = 1 by defaults
qc.to_gate() make qc to gate
swap = cx(0,1) + cx(1,0) + cx(0,1)

(4) Other methods for qc:
qc.initialize([1/sqrt(2), 0, 0, 1/sqrt(2)] ,[0,1])
qc.inverse()
qc.barrier() or qc.barrier([0,1])
qc.depth() return int
qc.draw()
qc.from_qasm_str()

(5) Measurement:
qc.measure([0,1,2], [0,1,2]) or qc.measure(qr, cr)
qc.measure_all() create new cr to store all result.

(6) Execute (Auto transpile & assemble)
backend = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
res = execute(qc, backend= backend, shots=1024,
coupling_map=couple_map)
Use memory=True if need individual results.

(7) Aer simulators:
Aer.get_backend('qasm_simulator') → get_counts(qc)
Only qasm allows noise
Aer.get_backend('statevector_simulator') →
get_statevector(qc)
Aer.get_backend('unitary_simulator') → get_ unitary(qc)
Backend.set_options(devices='GPU') → Nvidia

(8) Run (NO transpile & assemble)
job = backend.run(transpile(qc, backend))
job.result().get_xxx(qc)

(9) Operator (Allow noise compare to qc and gate)
op = Operator(Xgate()) or Operator(qc)

(10) Plot result:
counts = res.get_counts(qc)
 # {'01': 512} etc. Note that first qubit is last in string, sequence follow binary number from small to large
plot_histogram(counts)
OR:
res = Statevector.from_instruction(qc)
state = res. get_statevector(qc)
plot_bloch_multivector(state)
plot_state_city(state)
plot_state_qsphere(state)
Other plot:
plot_error_map(backend)
plot_bloch_vector([x, y, z])

(11) Fidelity
state_fidelity() for statevectors and densitymatrixs
average_gate_fidelity() and process_fidelity() for gates, unitarys, operators etc.
Global phase make no difference to all fidelity.

(12) Monitors (qiskit.tools)
job_monitor(job)
backend_monitor(backend)
backend.configuration() or .properties()
backend_overview(): all IBMQ backend info.

(13) Version check
qiskit.__version__ → for terra only
qiskit.__qiskit_version__ → for all qiskit sub packages