

Quantum Circuits

```
#Define Simulator
S_simulator = Aer.backends (name= 'statevector_simulator')[0]
M_simulator = Aer.backends (name= 'qasm_simulator')[0]
#Define Register (1 qubit)
qreg_q = QuantumRegister (1,'q')
creg_c = ClassicalRegister (1,'c')
#Define quantum circuit
qc = QuantumCircuit( qreg_q ,creg_c)
#Add Quantum gates to Circuit
qc.h(qreg_q[0])
# Add Measure
qc.measure(qreg_q, creg_c)
#Run circuit on Simulator
job = execute(qc,M_simulator)
result = job.result()
result.get_counts(qc)
```

Quantum Gates

Names	Example	Notes
I, Identity	<code>qc.id(0)</code> or <code>qc.i(0)</code>	Applies I gate to qubit 0.
H, Hadamard	<code>qc.h(0)</code>	Applies H gate to qubit 0.
X	<code>qc.x(0)</code>	Applies X gate to qubit 0.
Y	<code>qc.y([0,1,2])</code>	Applies Y gates to qubits 0, 1, and 2.
Z	<code>qc.z(0)</code>	Applies Z gate to qubit 0. Equivalent to P gate with π phase rotation.
P, Phase	<code>qc.p(math.pi/2,0)</code>	Applies P gate with $\pi/2$ phase rotation to qubit 0.
S	<code>qc.s(0)</code>	Applies S gate to qubit 0. Equivalent to P gate with $\pi/2$ phase rotation.
S†	<code>qc.sdg(0)</code>	Applies S† gate to qubit 0. Equivalent to P gate with $3\pi/2$ phase rotation.
SX	<code>qc.sx(0)</code>	Applies SX (square root of X) gate to qubit 0. Equivalent to RX gate with $\pi/2$ rotation.
T	<code>qc.t(0)</code>	Applies T gate to qubit 0. Equivalent to P gate with $\pi/4$ phase rotation.
T†	<code>qc.tdg(0)</code>	Applies T† gate to qubit 0. Equivalent to P gate with $7\pi/4$ phase rotation.
RX	<code>qc.rx(math.pi/4,0)</code>	Applies RX gate with $\pi/4$ rotation to qubit 0.
RY	<code>qc.ry(math.pi/8,0)</code>	Applies RY gate with $\pi/8$ rotation to qubit 0.
RZ	<code>qc.rz(math.pi/2,0)</code>	Applies RZ gate with $\pi/2$ rotation to qubit 0.
U	<code>qc.u(math.pi/2, 0,math.pi,5)</code>	Applies rotation with 3 Euler angles to qubit 5.

Quantum Teleportation

```
from qiskit import *
from qiskit.tools.jupyter import *
```



By Dinesh1102

Not published yet.

Last updated 24th December, 2022.

Page 1 of 6.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Quantum Teleportation (cont)

```

from qiskit.visualization import *
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display, Math, Latex
%matplotlib inline
sim = Aer.get_backend('aer_simulator')
qr = QuantumRegister(3)
crz = ClassicalRegister(1)
crx = ClassicalRegister(2) # we will need separate registers for using 'c_if' later.
qc = QuantumCircuit(qr, crz, crx)
qc.x(0)
qc.h(0) # 'psi' can't be known to us as we are creating it here. Let us take '-' state as our 'psi'.
This is done by operating X and H gate on the q0 i.e., H.X|0>

# We will verify later if the
'-' is been teleported.
qc.draw('mpl')
qc.h(1)
qc.cx(1,2) # creating a bell state
qc.barrier() # Use barrier to separate steps, everything till this barrier is just initialization.
qc.draw('mpl')
qc.cx(0,1) # '0' and '1' are with Alice and '2' is with Bob.
# psi_1 prepared.
qc.barrier() # Use barrier to separate steps
qc.draw('mpl')
qc.h(0)
# psi_2 prepared.
qc.barrier()
qc.draw('mpl')
qc.measure(0,0)
qc.measure(1,1)
qc.barrier()
qc.draw('mpl')
qc.x(2).c_if(crx,1) # 'c_if' compares a classical register with a value (either 0 or 1) and performs
the
qc.z(2).c_if(crz,1) # operation if they are equal.
qc.draw('mpl')
qc.h(2)
qc.measure(2, crx[1])
qc.draw('mpl')
qobj = assemble(qc)

```



By Dinesh1102

Not published yet.

Last updated 24th December, 2022.

Page 2 of 6.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Quantum Teleportation (cont)

```
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```

QKD with BB84 protocol

```
from qiskit import *
from qiskit.compiler import transpile, assemble
from qiskit.tools.jupyter import *
from qiskit.visualization import *
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display, Math, Latex
import math as m
%matplotlib inline
from qiskit import *
from qiskit.visualization import plot_histogram
%config InlineBackend.figure_format = 'svg'
qc = QuantumCircuit(6,6) #Create a quantum circuit with 6 qubits and 6 classical bits
##ENCODE BIT STRING
#The random bit sequence Alice needs to encode is: 100100, so the first and fourth qubits are flipped from
|0> -> |1>
qc.x(0) #The first qubit is indexed at 0, following Python being zero-indexed. From now on it'll be
referred to as qubit 0 and so on.
qc.x(3)
qc.barrier()
##ALICE CHOOSES
#Alice randomly chooses to apply an X or an H.
#Note that since the state is already either a |0> or |1>, a Z essentially leaves the qubit state
unchanged. But let's write it anyway, shall we?
qc.h(0) # or qc.z(0) # switch these based on your own choice
qc.z(1) # or qc.h(1)
qc.z(2) # or qc.h(2)
qc.h(3) # or qc.z(3)
qc.z(4) # or qc.h(4)
qc.h(5) # or qc.z(5)
qc.barrier()
##BOB CHOOSES
#Alice sends the qubit sequence to Bob, and Bob randomly chooses measurements
qc.h(0) # or qc.z(0) # switch these based on your own choice
qc.z(1) # or qc.h(1)
qc.h(2) # or qc.z(2)
```



By Dinesh1102

Not published yet.

Last updated 24th December, 2022.

Page 3 of 6.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

QKD with BB84 protocol (cont)

```

qc_ab.h(3) # or qc.z(3)
qc_ab.z(4) # or qc.h(4)
qc_ab.z(5) # or qc.h(5)
qc_ab.barrier()
##PUBLICIZE CHOICES
#Alice and Bob publicize their choices and only retain those for which their choices match. In this case:
qubits 0,1,3,4.
#Note: technically Bob performs the measurement BEFORE publicizing, but we're combining the two here
since no one is actually communicating.
qc_ab.measure(0,0)
qc_ab.measure(1,1)
qc_ab.measure(3,3)
qc_ab.measure(4,4)
#qc_ab.measure(2,2) #come back to uncomment these to see what happens to the results after you've run
this once
#qc_ab.measure(5,5)
qc_ab.draw(output='mpl') #let's see what this circuit looks like!
##EXECUTE
result = execute(qc_ab, Aer.get_backend('qasm_simulator')).result().get_counts() #We're
only making use of the simulator. Refer to [2] to see how you can run this on a real quantum computer.
plot_histogram(result)
#Same situation but now with an eavesdropper (Eve)
qc_aeb = QuantumCircuit(6,6) #Create a quantum circuit with 6 qubits and 6 classical bits
##ENCODE BIT STRING
qc_aeb.x(0)
qc_aeb.x(3)
qc_aeb.barrier()
##ALICE CHOOSES
qc_aeb.h(0)
qc_aeb.z(1)
qc_aeb.z(2)
qc_aeb.h(3)
qc_aeb.z(4)
qc_aeb.h(5)
qc_aeb.barrier()
##EVE CHOOSES
qc_aeb.h(0) #play around with these to see how many states with non-zero probabilities show up at the end
for a fixed set of Alice's and Bob's choices
qc_aeb.z(1)
qc_aeb.h(2)
qc_aeb.h(3)
qc_aeb.z(4)

```



By Dinesh1102

Not published yet.

Last updated 24th December, 2022.

Page 4 of 6.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

QKD with BB84 protocol (cont)

```
qc_aeb.z(5)
qc_aeb.barrier()
##BOB CHOOSES
qc_aeb.h(0)
qc_aeb.z(1)
qc_aeb.h(2)
qc_aeb.h(3)
qc_aeb.z(4)
qc_aeb.z(5)
qc_aeb.barrier()
##PUBLICIZE CHOICES
qc_aeb.measure(0,0)
qc_aeb.measure(1,1)
qc_aeb.measure(3,3)
qc_aeb.measure(4,4)
#qc_aeb.measure(2,2) #come back to uncomment these to see what happens to the results after you've run
this once
#qc_aeb.measure(5,5)
qc_aeb.draw(output='mpl') #let's see what this circuit looks like!
##EXECUTE
result = execute(qc_aeb, Aer.get_backend('qasm_simulator')).result().get_counts()
plot_histogram(result)
```

Import Libraries

```
from qiskit import
QuantumRegister,
ClassicalRegister,
QuantumCircuit, Aer, execute
from qiskit.visualization
import visualize_transition
import numpy as np
import math as m
```



By Dinesh1102

cheatography.com/dinesh1102/

Not published yet.

Last updated 24th December, 2022.

Page 6 of 6.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

