Qubits and operators

Link to IBM Quantum platform

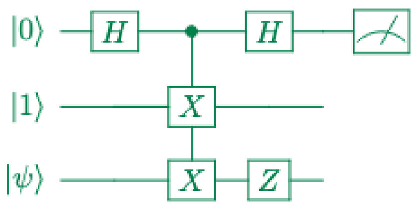
What we want to study today is quantum dynamics from a quantum computation perspective? The question here is if I want to simulate quantum mechanics in a logical manner to perform a computation task -- how do we go about it?

So, our quantum states become our logical inputs and our evolution operators become our gates, and we connect them using a circuit.

Here is a Wikipedia link to logic gates

In [62]:
```
from IPython.display import Image
Image("Circuit.png")
```

Out[62]:



A qubit

A single qubit quantum state can be written as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where α and β are complex numbers. In a measurement the probability of the bit being in $|0\rangle$ is $|\alpha|^2$ and $|1\rangle$ is $|\beta|^2$.

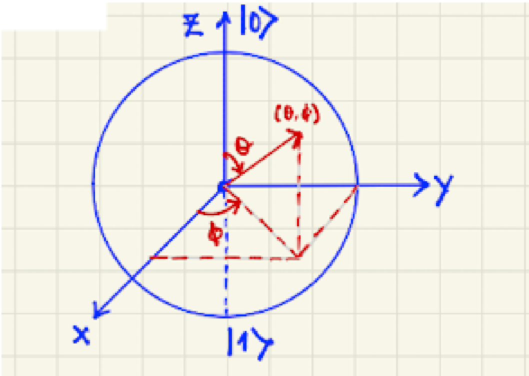
As a vector this is $|\psi\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix}$, where $|0\rangle = \begin{pmatrix}1 \\ 0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0 \\ 1\end{pmatrix}$.

As such, a qubit is a vector in a two-dimensional Hilbert space descibed by an orthonormal basis $|0\rangle$ and $|1\rangle$, and with an unit norm i.e, $\langle\psi|\psi\rangle = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}\begin{pmatrix}\alpha \\ \beta\end{pmatrix} = 1$.

A convenient representation is $|\psi\rangle = \cos(\theta/2)\,|0\rangle + \sin(\theta/2)e^{i\phi}\,|1\rangle = \begin{pmatrix}\cos(\frac{\theta}{2}) \\ e^{i\varphi}\sin(\frac{\theta}{2})\end{pmatrix}$, where $0 \le \phi < 2\pi$, and $0 \le \theta \le \pi$. The qubit can be viewed as a unit line on the Bloch sphere.

In [63]:
```
Image("Bloch.png")
```

Out[63]:



A single qubit gate

Quantum gates/operations are usually represented as unitary matrices. For instance, a gate which acts on a qubit is represented by a 2×2 unitary matrix. The qubit evolution under a gate/unitary can be viewed as rotations in a Bloch sphere.

In quantum mechanics, the final state after a unitary matrix acts on an initial state is given by: $|\psi_f\rangle = U|\psi_{in}\rangle$, where, $U^\dagger U = UU^\dagger = I$.

X gate

The X gate is nothing but the Pauli σ_x operator we have seen before. It represents a π rotation around the x axis of the Bloch sphere.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

What does it do? $X|0\rangle \rightarrow |1\rangle$ and $X|1\rangle \rightarrow |0\rangle$. What is this called classically?

What is happening here is simple:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

In [64]:

```
import numpy as np
import matplotlib.pyplot as plt
from math import pi
# import qutip as qu
```

In [65]:

```
state_0 = np.array([1,0])
state_1 = np.array([0,1])
print (state_0,'\n')
print (state_1,'\n')

X = np.array([[0,1],[1,0]])

print (X)
X@state_0
```

```
[1 0]

[0 1]

[[0 1]
 [1 0]]
```

Out[65]:

```
array([0, 1])
```

The Hadamard gate

The Hadamard gate is given by:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

What does H do? $X|0\rangle \rightarrow |+\rangle$ and $X|1\rangle \rightarrow |-\rangle$, where $|\pm\rangle = |0\rangle \pm |1\rangle$.

Alternately,

We know that: $H\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $H\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

What is the classical analogue here?

In [66]:
```
H = (1/np.sqrt(2))*np.array([[1,1],[1,-1]])  # Hadamard gate
print (H)
```

```
[[ 0.70710678  0.70710678]
 [ 0.70710678 -0.70710678]]
```

In [67]:
```
print(state_1)
print(state_0)
print (H@state_1)
print (H@state_0)
```

```
[0 1]
[1 0]
[ 0.70710678 -0.70710678]
[0.70710678 0.70710678]
```

Other important gates:

The Y gate:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

The Z gate:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The phase or S gate:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}.$$

The $\pi/8$ or T gate:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

In [68]:
```
Y = np.array([[0,-1j],[1j,0]])
Z = np.array([[1,0],[0,-1]])
S = np.array([[1,0],[0,-1j]])
T = np.array([[1,0],[0,np.exp(1j*pi/4)]])

print (T)
```

```
[[1.        +0.j          0.        +0.j          ]
 [0.        +0.j          0.70710678+0.70710678j]]
```

General single qubit gates

Mathematically, the general form can be expressed as:

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

or alternatively,

$$U = e^{i\varphi/2} \begin{pmatrix} e^{i\alpha}\cos\theta & e^{i\beta}\sin\theta \\ -e^{-i\beta}\sin\theta & e^{-i\alpha}\cos\theta \end{pmatrix},$$

where $\varphi, \alpha, \beta, \theta$ can take any values. This is the most general form of a single qubit unitary. The σ_x or X operator is given by:

$$U(\theta = \pi/2, \alpha = 0, \beta = \pi/2, \varphi = -\pi) = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The σ_y or Y operator is given by:

$$U(\theta = \pi/2, \alpha = 0, \beta = 0, \varphi = -\pi) = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Another useful representation, is based on the rotations matrices:

$$R_x(\theta) = e^{-i\theta X/2} = \cos(\theta/2)I - i\sin(\theta/2)X = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix},$$

$$R_y(\theta) = e^{-i\theta Y/2} = \cos(\theta/2)I - i\sin(\theta/2)Y = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix},$$

$$R_z(\theta) = e^{-i\theta Z/2} = \cos(\theta/2)I - i\sin(\theta/2)Z = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix},$$

where $R_x(\theta)$, $R_y(\theta)$ and $R_y(\theta)$ are rotations around the X, Y and Z axes.

By introducing $\alpha = \psi + \delta$ and $\beta = \psi - \delta$, has the following factorization:

$$U = e^{i\varphi/2} \begin{pmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} e^{i\delta} & 0 \\ 0 & e^{-i\delta} \end{pmatrix}.$$

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta).$$

In [69]:
```
# Define the generic U operator

def U(θ,α,β,φ):
    return np.around(np.exp(1j*φ/2)*np.array([[np.exp(1j*α)*np.cos(θ),np.exp(1j*β)*
                                               [-1*np.exp(-1j*β)*np.sin(θ),np.exp(-1j*α)*np.cos(θ)]

print ("X =",U(pi/2,0,pi/2,-pi),'\n')  # X from the general U(θ,ϕ,λ)

print ("Y =",U(pi/2,0,0,-pi),'\n') # X from the general U(θ,ϕ,λ)
```

```
X = [[0.-0.j 1.+0.j]
 [1.+0.j 0.-0.j]]

Y = [[ 0.-0.j  0.-1.j]
 [-0.+1.j  0.-0.j]]
```

Two-qubit quantum states and operators

A two qubit quantum state can be written as the superposition of tensor-products of two single qubit states:

$$|\psi_{AB}\rangle = \mathcal{N}\sum_i \begin{pmatrix} \alpha_i^A \\ \beta_i^A \end{pmatrix} \otimes \begin{pmatrix} \alpha_i^B \\ \beta_i^B \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix},$$

where $\mathcal{N}$ is a normalization constant. The basis state here is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

Similarly, two qubit operators/gates can be written as 4×4 matrices.

CNOT gate

The CNOT gate is a two operator gate, with a control on one qubit and X gate on the other. It's operation is given by:

$U_{CNOT}(|0\rangle \otimes |0\rangle) \rightarrow |0\rangle \otimes |0\rangle$ and $U_{CNOT}(|1\rangle \otimes |0\rangle) \rightarrow |1\rangle \otimes |1\rangle$.

It can be written as:

$$U_{CNOT} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}}_{|0\rangle\langle 0|} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}}_{|1\rangle\langle 1|} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

All quantum gates can be created using the single qubit, U, and the CNOT gate, U_{CNOT}.

In [70]:

```
proj_0 = np.outer(state_0,state_0) # top half of the identity matrix
proj_1 = np.outer(state_1,state_1) # bottom half of the identity matrix

U_cnot = np.kron(proj_0,np.eye(2))+np.kron(proj_1,X)
print (U_cnot)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]]
```

In [71]:

```
print (U_cnot@(np.kron(state_1,state_1)),'\n')

print (np.kron(state_1,state_0))
```

```
[0. 0. 1. 0.]

[0 0 1 0]
```

CNOT and entanglement

Also, CNOT acting on these gates can create entanglement:

The maximally entangled state can be written as:

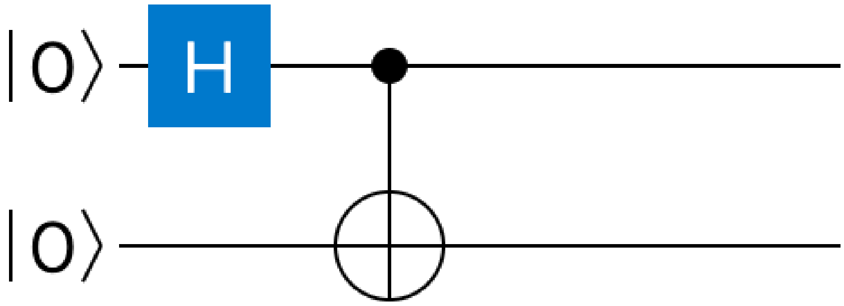
$$|\psi_{AB}\rangle = \frac{1}{\sqrt{2}}\left[\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right] = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix},$$

Applying the CNOT with Hadamard:

$U_{CNOT}\left(H\left|1\right\rangle\otimes\left|1\right\rangle\right)=\frac{1}{\sqrt{2}}(|01\rangle-|10\rangle)$

In [72]:
```
Image("CNOT_ent.png")
```

Out[72]:



In [73]:
```
U_cnot@(np.kron(H@state_1,state_1)) # Entangled singlet state
```

Out[73]:
```
array([ 0.        ,  0.70710678, -0.70710678,  0.        ])
```

Controlled-U gate

Similaro to CNOT gate you can create any controlled single-qubit gate, with a control on one qubit and U gate on the other. It's operation is given by:

$U_c(|0\rangle\otimes|0\rangle)\rightarrow|0\rangle\otimes I|0\rangle$ and $U_c(|1\rangle\otimes|0\rangle)\rightarrow|1\rangle\otimes U|0\rangle$.

Try, what this looks for the controlled Hadamard.

In [74]:
```
U_H = np.kron(proj_0,np.eye(2))+np.kron(proj_1,H)
print (U_H)
```

```
[[ 1.          0.          0.          0.        ]
 [ 0.          1.          0.          0.        ]
 [ 0.          0.          0.70710678  0.70710678]
 [ 0.          0.          0.70710678 -0.70710678]]
```

Measurements

Consider the single qubit state:

$$\left|\psi\right\rangle=\alpha\left|0\right\rangle+\beta\left|1\right\rangle$$

What does measurement mean here? It is the probability of the state being in $|0\rangle$ or $|1\rangle$?

This is given by:

Prob$(|0\rangle)=\langle 0|(|\psi\rangle\langle\psi|)|0\rangle$ = Tr$(\prod_{|0\rangle\langle 0|}|\psi\rangle\langle\psi|)$, and similarly

Prob$(|1\rangle)=\langle 1|(|\psi\rangle\langle\psi|)|1\rangle$ = Tr$(\prod_{|1\rangle\langle 1|}|\psi\rangle\langle\psi|)$.

In [75]:
```
state_0 = np.array([1,0])
state_1 = np.array([0,1])

psi = np.sqrt(1/3)*state_0 + np.sqrt(2/3)*state_1

print (psi)
```

```
[0.57735027 0.81649658]
```

In [76]:
```
prob0 = np.dot(state_0,np.outer(psi,psi)@state_0)
print (prob0)

prob1 = np.dot(state_1,np.outer(psi,psi)@state_1)
print (prob1)
```

```
0.3333333333333333
0.6666666666666666
```

In [77]:
```
proj_0 = np.outer(state_0,state_0) # top half of the identity matrix
proj_1 = np.outer(state_1,state_1) # bottom half of the identity matrix

postmeas0 = proj_0@np.outer(psi,psi)@proj_0
print (postmeas0,'\n')

postmeas1 = proj_1@np.outer(psi,psi)@proj_1
print (postmeas1)
```

```
[[0.33333333 0.        ]
 [0.         0.        ]]

[[0.         0.        ]
 [0.         0.66666667]]
```

Consider the two qubit entangled state: $\psi = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix}$, or the state $\psi = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Measurements can be done on both the qubits:

Prob$(|ij\rangle) = \langle ij|\psi\rangle\langle\psi|ij\rangle$, with the post measurement state being given by $\prod_{|ij\rangle\langle ij|} |\psi\rangle\langle\psi| \prod_{|ij\rangle\langle ij|}$

In [78]:
```
psi = U_cnot@(np.kron(H@state_1,state_1)) # Entangled singlet state
psi = (1/2)*np.ones(4) # Entangled singlet state

proj_0 = np.outer(state_0,state_0) # top half of the identity matrix
proj_1 = np.outer(state_1,state_1) # bottom half of the identity matrix


meas0 = np.kron(proj_0,np.eye(2))@np.outer(psi,psi)@np.kron(proj_0,np.eye(2))
meas1 = np.kron(proj_1,np.eye(2))@np.outer(psi,psi)@np.kron(proj_1,np.eye(2))

print (meas0,'\n')
print (meas1)
```

```
[[0.25 0.25 0.   0.  ]
 [0.25 0.25 0.   0.  ]
 [0.   0.   0.   0.  ]
 [0.   0.   0.   0.  ]]

[[0.   0.   0.   0.  ]
 [0.   0.   0.   0.  ]
 [0.   0.   0.25 0.25]
 [0.   0.   0.25 0.25]]
```

Tasks for today:

1. Find the output for the following operations:

- $XX|0\rangle$

- $HXH|0\rangle$
- $XY|+\rangle$

2. Define functions that create the 3 rotation matrices $R_x(\theta), R_y(\theta)$ and $R_z(\theta)$?

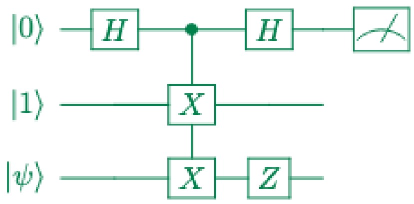
3. Define a function that creates an arbitrary single qubit gate using the 4 angles: $U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta)$.

   What are the values for a Hadamard and X gate?

4. Define a function that creates the three qubit circuit below by using the gates defined earlier, for an arbitrary state $|\psi\rangle$:

In [79]:
```
Image("Circuit.png")
```

Out[79]:



In [ ]:
```
```