Counting States

Name: Instructor: Section: Date:

First let's make's make sure that all the libraries that we will be using are installed, you can do this by running the following cell:

In [ ]:
```
!conda install numpy, scipy, seaborn, matplotlib
```

Part 1: Today we are going to learn about the statistical properties of physical systems, and how that affects the energy distribution in them and the probability of macrostates. We will model a system of 2 blocks, in the Block 1, there are 2 atoms, and in Block 2, there is 1 atom. We can model atoms as 3 oscillators (1 oscillator per dimension).

If we add one quantum of energy, it can go into either blocks, with probability 2/3 of going into block 1, and 1/3 going into block 2. Why? What happens if we add 4 quanta?

Fill in the table below with the 5 different ways of arranging the 4 quanta into the blocks. In lectures, we learned that the number of microstates (i.e. number of ways to arrange) q_i quanta into N_i oscillators can be calculated using the formula:

$$\Omega_i = \frac{(q_i + N_i - 1)\,!}{q_i\,!\,(N_i - 1)\,!}$$

q_1	q_2	Ω_1	Ω_2	Ω_{total}	**Probability**
0					
1					
2					
3					
4					
Total	—	—	—		1

Now we're gonna try to recreate the above table using Python, and see if we got everything correctly.

Import some libraries: numpy and scipy are the essential numerical and scientific computing libraries in Python. Seaborn and Matplotlib are plotting libraries.

In [1]:
```
import numpy as np
from scipy.special import factorial
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Let's first define a function that we can use later, which takes the total number of quanta of energy, q, and a number, N, as arguments and returns the number of ways q can be distributed among N oscillator states, using the formula for Ω above.

In [2]:
```
def Omega_fun(q,N):
    Omega = np.zeros(len(q))
    Omega = factorial(q+N-1)/(factorial(q)*factorial(N-1))
    return Omega
```

Define number of oscillators in each block and number of energy quanta

In [3]:
```
N1=6
N2=3
qTotal=4
q1=np.linspace(0,qTotal,qTotal+1)
q2=np.zeros(len(q1))
q2 = qTotal - q1
```

Use the function above to calculate Ω_1 , Ω_2 and Ω_{total}

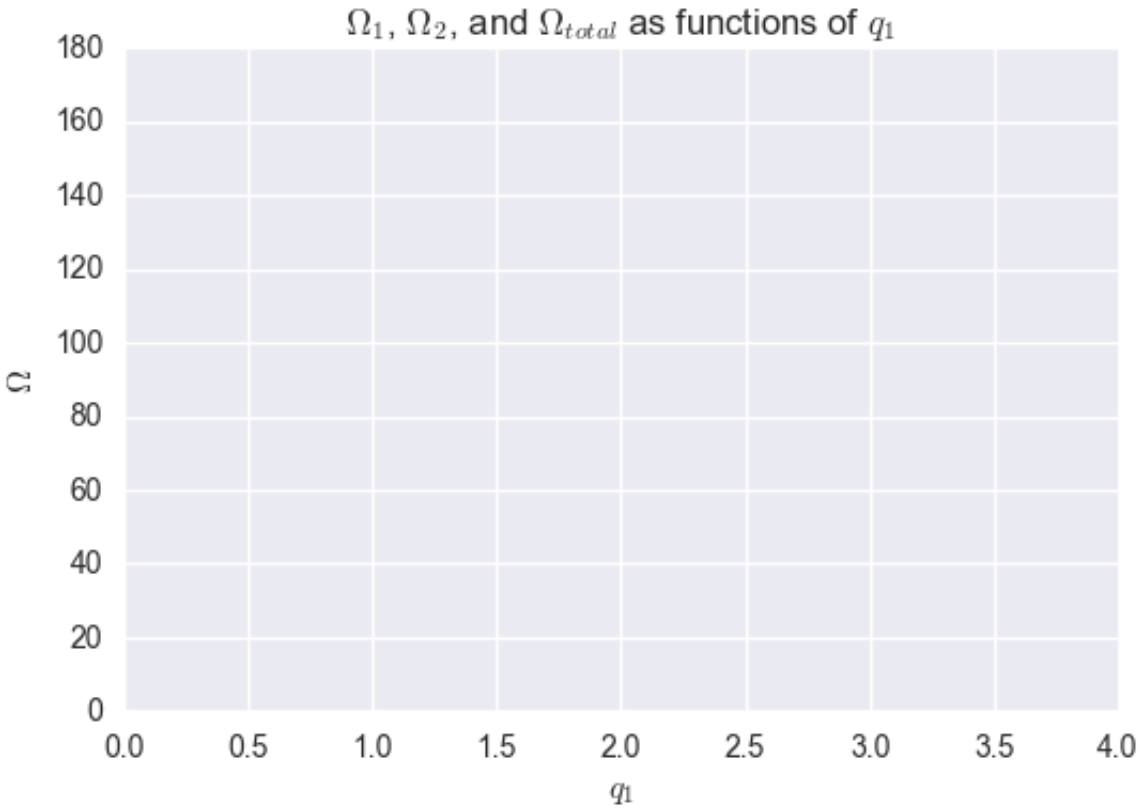
In [4]:
```
Omega1=Omega_fun(q1,N1)
Omega2=Omega_fun(q2,N2)
OmegaTotal=Omega1*Omega2
```

Plot Ω_1 , Ω_2 and Ω_{total} as **funtions of q_1**

In [5]:
```
sns.set_style("darkgrid")
plt.plot(q1, Omega1, sns.xkcd_rgb["pale red"],   lw=2,label=r'$\Ome
ga_1$')
plt.plot(q1, Omega2, sns.xkcd_rgb["denim blue"], lw=2,label=r'$\Ome
ga_2$')
plt.plot(q1, OmegaTotal, sns.xkcd_rgb["muted green"], lw=2,label=r'
$\Omega_{total}$')
plt.xlabel(r'$q_1$')
plt.ylabel(r'$\Omega$')
plt.legend(loc="upper left")
plt.title(r'$\Omega_1$, $\Omega_2$, and $\Omega_{total}$ as functio
ns of $q_1$')

print ('SUMMARY')
print ('N1 = %d , N2 = %d' %(N1,N2))
print ('Total number of energy quanta ',qTotal)
print ('Total Number of States :',np.sum(OmegaTotal))
# for i in range(len(q1)):                                    # uncom
ment to display all values
#    print (q1[i],q2[i],Omega1[i],Omega2[i],OmegaTotal[i]) # uncomm
ent to display all values
```

```
SUMMARY
N1 =    , N2 =
Total number of energy quanta:
Total Number of States:
```



Suppose every quantum of energy represents $0.2eV$ of energy:

a. What is the total amount of energy in the system?

b. If we pull the two blocks apart, what is the most probable amount of energy in eV in each of the two blocks?

c. What is the probability that the 1-atom block (block 2) is in its ground state? (This means we find the system with zero quanta in its ground state.)

Part 2: Now let's increase the number of oscillators and quanta to $N_1 = 20, N_2 = 30$ and $q_{total} = 100$, and recalculate Ω's, and plot everything again.

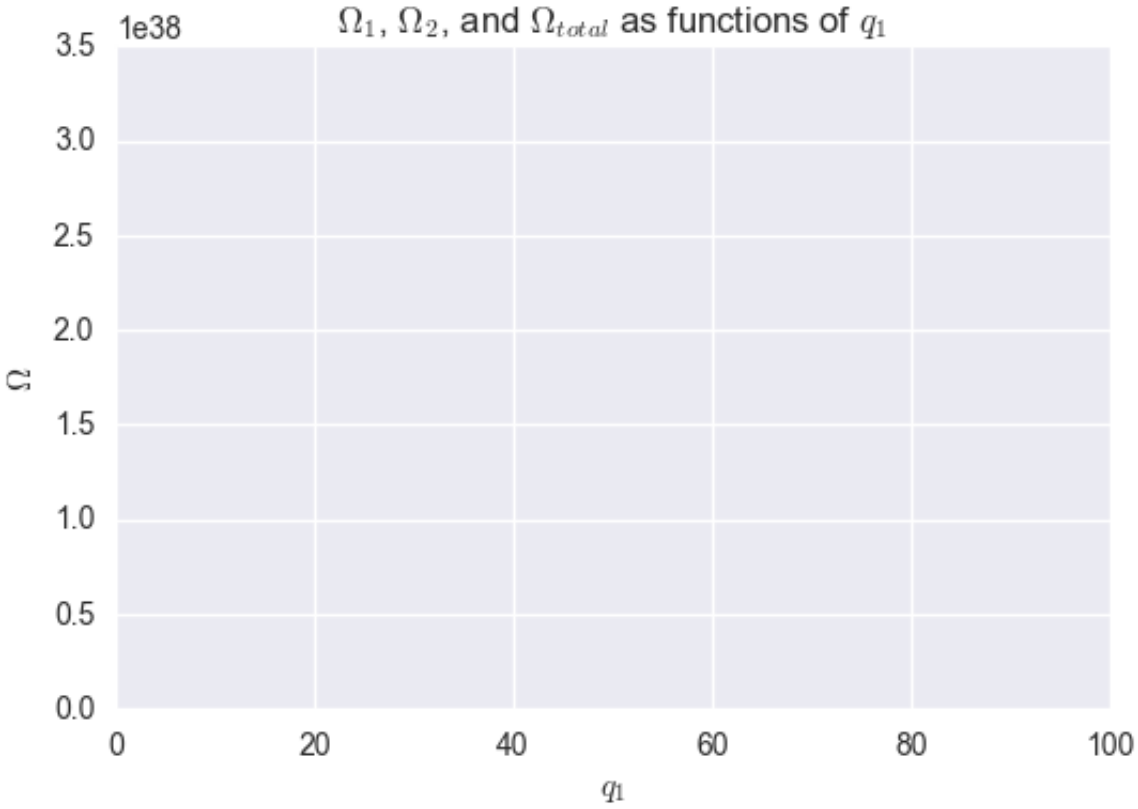
In [6]:
```
N1=
N2=
qTotal=
q1=np.linspace(0,qTotal,qTotal+1)
q2=np.zeros(len(q1))
q2 = qTotal - q1
```

In [7]:
```
Omega1=Omega_fun(q1,N1)
Omega2=Omega_fun(q2,N2)
OmegaTotal=Omega1*Omega2
```

In [8]:
```
sns.set_style("darkgrid")
plt.plot(q1, Omega1, sns.xkcd_rgb["pale red"],   lw=2,label=r'$\Ome
ga_1$')
plt.plot(q1, Omega2, sns.xkcd_rgb["denim blue"], lw=2,label=r'$\Ome
ga_2$')
plt.plot(q1, OmegaTotal, sns.xkcd_rgb["muted green"], lw=2,label=r'
$\Omega_{total}$')
plt.xlabel(r'$q_1$')
plt.ylabel(r'$\Omega$')
plt.legend(loc="upper left")
plt.title(r'$\Omega_1$, $\Omega_2$, and $\Omega_{total}$ as functio
ns of $q_1$')

print ('SUMMARY')
print ('N1 = %d , N2 = %d' %(N1,N2))
print ('Total number of energy quanta ',qTotal)
print ('Total Number of States :',np.sum(OmegaTotal))
# for i in range(len(q1)):                                  # uncom
ment to display all values
#    print (q1[i],q2[i],Omega1[i],Omega2[i],OmegaTotal[i]) # uncomm
ent to display all values
```

```
SUMMARY
N1 =    , N2 =
Total number of energy quanta:
Total Number of States:
```



You will notice that it is now very hard to see the blue and red curves — the reason for that is because the green curve's peak is several orders of magnitude higher than that of either the blue or the red. It becomes harder and harder to see them when we increase the number of Ω (by increasing N and q). To see the red and blue curves, try commenting out the line that plots the green curve and run the above cell again. Is the trend from the red and blue curves consistent with what you expect from your table?

a. What is the most likely amount of quanta, and corresponding amount of energy, for each of block 1 and block 2? Remember that Ω above is plotted as a function of q_1 or equivalently, a function of decreasing q_2 from 100 to 0.

b. What is the probability of this particular state? (Hint: you can uncomment the last line in the cell above to print exact values)

c. If we are able to make 10 observations per second, how long do we have to observe to detect this maximum probability state? (Hint: If the probability was 1, then we need 1 observation to detect this state, and thus would have to wait 0.1s, if the probability was 0.1, we would need on average 10 observations to detect such a state, and thus would have to observe for 1s...)

Part 3: Let's try changing q and N one last time into $N_1 = 70, N_2 = 30$ and $q_{total} = 100$ and recalculate and plot:

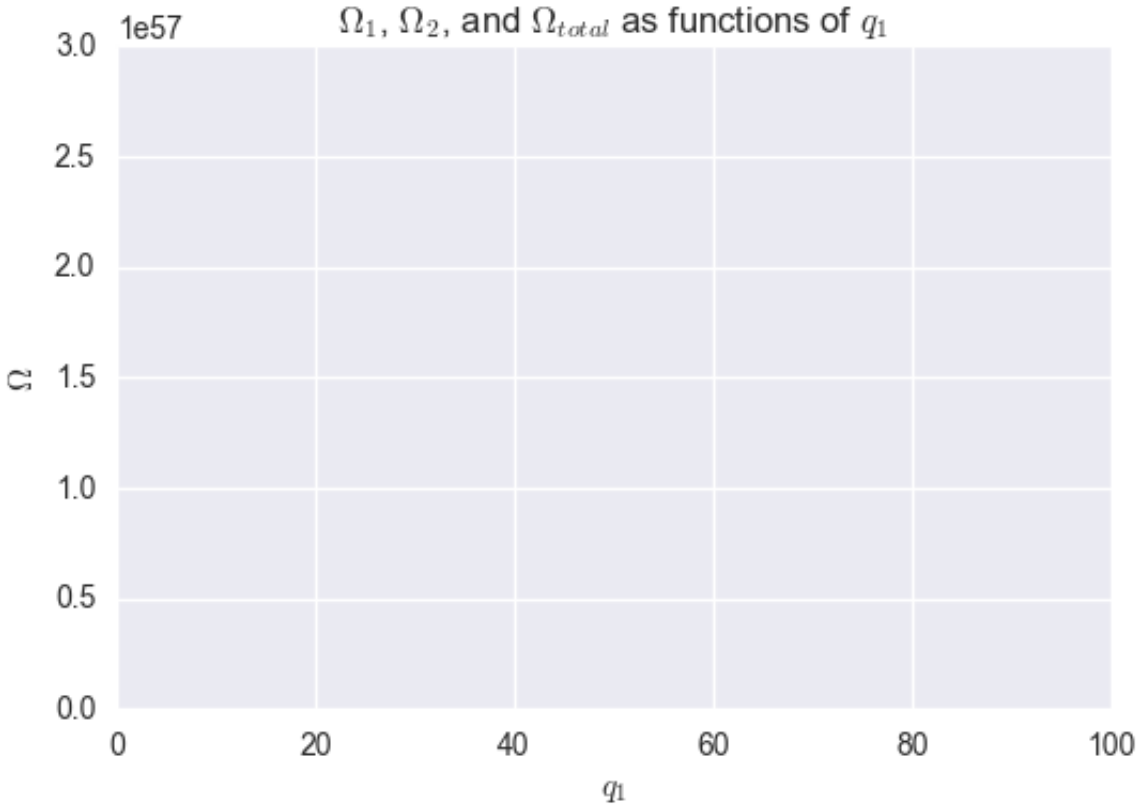
In [9]:
```
N1=
N2=
qTotal=
q1=np.linspace(0,qTotal,qTotal+1)
q2=np.zeros(len(q1))
q2 = qTotal - q1
```

In [10]:
```
Omega1=Omega_fun(q1,N1)
Omega2=Omega_fun(q2,N2)
OmegaTotal=Omega1*Omega2
```

In [11]:
```
sns.set_style("darkgrid") # this line just adds some prettiness :-)
plt.plot(q1, Omega1, sns.xkcd_rgb["pale red"],   lw=2,label=r'$\Ome
ga_1$')
plt.plot(q1, Omega2, sns.xkcd_rgb["denim blue"], lw=2,label=r'$\Ome
ga_2$')
plt.plot(q1, OmegaTotal, sns.xkcd_rgb["muted green"], lw=2,label=r'
$\Omega_{total}$')
plt.xlabel(r'$q_1$')
plt.ylabel(r'$\Omega$')
plt.legend(loc="upper left")
plt.title(r'$\Omega_1$, $\Omega_2$, and $\Omega_{total}$ as functio
ns of $q_1$')

print ('SUMMARY')
print ('N1 = %d , N2 = %d' %(N1,N2))
print ('Total number of energy quanta ',qTotal)
print ('Total Number of States :',np.sum(OmegaTotal))
# for i in range(len(q1)):                                    # uncomm
ent to display all values
#    print (q1[i],q2[i],Omega1[i],Omega2[i],OmegaTotal[i]) # uncomm
ent to display all values
```

```
SUMMARY
N1 =    , N2 =
Total number of energy quanta:
Total Number of States:
```



a. The two changes that we made are: doubling the number of oscillators, and redistributing them in a different proportion. What are the effects of those two changes that you can see from the plots? How can you interpret them physically?

Solution

1. Doubling the number of oscillators made the curve narrower — physical interpretation: we are more certain about the highest probability state in part 3 than in part 2, i.e. approaching the classical limit.
2. Putting more oscillators in the first box shifter the peak to the right — physical interpretation: higher probability of having more quanta of energy in the first box than in the second.

b. Calculate the probability of the maximum state for the configuration in **Part 2** to occur in this new configuration.

c. If we are able to make 10 observations per second, how long do we have to observe to detect this maximum probability state?

d. What would happen if we 10^{23} oscillators (Hint #1: In our classical world, we usually have numbers closer to 10^{23} than to a hundred. Think about what this classical limit would imply on the probability distribution. Hint #2: Do not try inputting 10^{23} oscillators in the code above — this would result in Python returning **infinite** (also known in programming as "nan" which stands for "not a number" values for the number of microstate)

Written by Husni Almoubayyed (http://www.husni.space) based on [] for CMU's 33-121 F16-F17 classes. Support and updates available at https://github.com/hsnee/Jupyterize33121