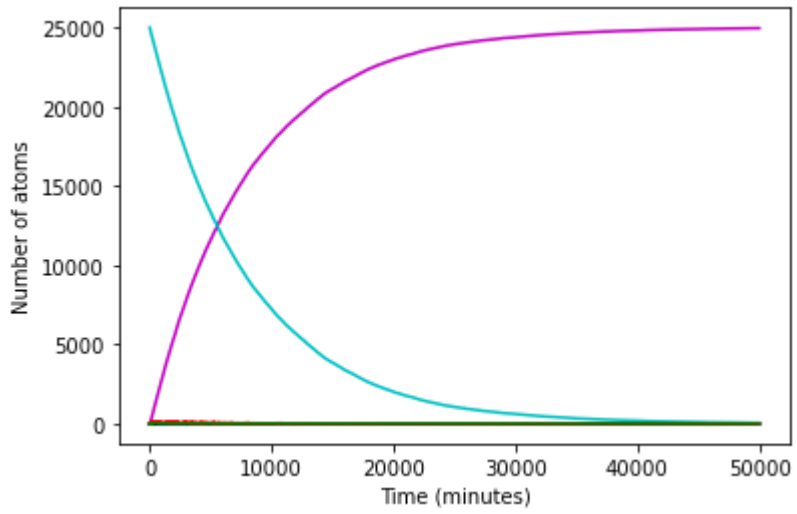
In [1]:

```
import numpy as np
from numpy import arange
from pylab import plot,xlabel,ylabel,show
import math
# Constants
N222Rn=25000 #change to 25000 starting atom
N218Po=0
N214Pb=0
N214Bi=0
N207Tl=0
N207Pb=0

h=1  #ONE MINUTE AS THE TIME STEP  simulate the decay of the atoms by dividing ti
#used this way on prev. problem and it works
pRn = 1 - (2**(-h/(5500.8)))   # Probability of decay in one step using half life
pPo= 1-(2**(-h/3.1))
pPb=1-(2**(-h/26.8))
pBi=1-(2**(-h/19.9))
pTl=1-(2**(-h/1.3))
tmax = 3000000/60  #time in minutes aka 50,000 minutes
#tmax=1500000/60
#(5.0 times 10^6)
# Lists of plot points
tpoints = arange(0.0,tmax,h) # make time array
Popts = [] # empty array to store # of Po atoms at each time step
Pb214pts = [] # empty array to store # of Pb atoms at each time step
Bi214pts=[] # empty array to store # of Bi atoms at each time step
Tl207pts=[] # empty array to store # of Tl atoms at each time step
Rn222pts=[] #starting atoms
Pb207pts=[] #ending atoms
alpha=4
r=9
beta=3
z=8
totalpha=0
totr=0
totbeta=0
totz=0
totenergy=0
# Main loop

alphaenergy=[]
betaenergy=[]
renergy=[]
zenergy=[]
for t in tpoints:    # for time 0-50000 minutes
    Popts.append(N218Po) # append starting # of Tl atoms
    Pb214pts.append(N214Pb) # append starting # of Pb atoms
    Bi214pts.append(N214Bi)
    Tl207pts.append(N207Tl)
    Rn222pts.append(N222Rn)
    Pb207pts.append(N207Pb)

    decay=0
    for m in range(N207Tl):
        if np.random.random()< pTl:
```

```
        decay+=1
        totbeta+=beta
N207Tl-=decay
N207Pb+=decay

decay=0
decayTl=0
decayPb=0
for l in range(N214Bi):
    if np.random.random() < pBi:
        decay+=1
        if np.random.random()<=0.997:
            decayTl+=1
            totalpha+=alpha
        else:
            decayPb+=1
            totr+=r
N214Bi-=decay
N207Tl+=decayTl
N207Pb+=decayPb

decay=0
for k in range(N214Pb):
    if np.random.random()< pPb:
        decay +=1
        totz+=z
N214Pb-=decay
N214Bi+=decay

decay=0
decayBi=0
decayPb2=0
for j in range(N218Po):
    if np.random.random()< pPo:
        decay+=1
        if np.random.random() <=0.9998:
            decayBi+= 1
            totbeta+=beta
        else:
            decayPb2+=1
            totalpha+=alpha
N218Po-=decay
N214Pb +=decayPb2
N214Bi +=decayBi

# Calculate the number of atoms that decay
decay=0
for i in range(N222Rn): # determine number of atoms that decay
    if np.random.random() < pRn:
        decay += 1
        totr+=r
N222Rn -= decay # subtract number of decayed Parent atoms
N218Po += decay # add number of decayed Daughter atoms


#we will be doing a multi step logic
# make sure you have the same amount of atoms that you started with
```

```
# Make the graph
plot(tpoints,Pb214pts, c='b') # plot Pb214 vs. time
plot(tpoints,Pb207pts, c='m') # plot Pb vs. time
plot(tpoints,Popts, c='k')
plot(tpoints,Rn222pts, c='c') #plot Rn vs. time
plot(tpoints,Bi214pts, c='r')
plot(tpoints, Tl207pts, c='g')
xlabel("Time (minutes)")
ylabel("Number of atoms")
show()
alphaenergy.append(totalpha)
betaenergy.append(totbeta)
renergy.append(totr)
zenergy.append(totz)
print(totalpha, totbeta, totz, totr)
totenergy=totalpha+totbeta+totz+totr
print(totenergy)
```




```
99552 149505 48 225324
474429
```

In [2]:
```
#works! this is for part 3 of final project.
for a in range(0,9):
    if( a >=1):
        N222Rn=25000 #change to 25000 starting atom
        N218Po=0
        N214Pb=0
        N214Bi=0
        N207Tl=0
        N207Pb=0
        totalpha=0
        totbeta=0
        totr=0
        totz=0
    for t in tpoints:    # for time 0-10000 minutes
        Popts.append(N218Po) # append starting # of Tl atoms
        Pb214pts.append(N214Pb) # append starting # of Pb atoms
        Bi214pts.append(N214Bi)
        Tl207pts.append(N207Tl)
        Rn222pts.append(N222Rn)
        Pb207pts.append(N207Pb)

        decay=0
        for m in range(N207Tl):
            if np.random.random()< pTl:
                decay+=1
                totbeta+=beta
        N207Tl-=decay
        N207Pb+=decay

        decay=0
        decayTl=0
        decayPb=0
        for l in range(N214Bi):
            if np.random.random() < pBi:
                decay+=1
                if np.random.random()<=0.997:
                    decayTl+=1
                    totalpha+=alpha
                else:
                    decayPb+=1
                    totr+=r
        N214Bi-=decay
        N207Tl+=decayTl
        N207Pb+=decayPb

        decay=0
        for k in range(N214Pb):
            if np.random.random()< pPb:
                decay +=1
                totz+=z
        N214Pb-=decay
        N214Bi+=decay

        decay=0
        decayBi=0
        decayPb2=0
```

```
        for j in range(N218Po):
            if np.random.random()< pPo:
                decay+=1
                if np.random.random() <=0.9998:
                    decayBi+= 1
                    totalpha+=alpha
                else:
                    decayPb2+=1
                    totbeta+=beta
        N218Po-=decay
        N214Pb +=decayPb2
        N214Bi +=decayBi

    # Calculate the number of atoms that decay
        decay=0
        for i in range(N222Rn): # determine number of atoms that decay
            if np.random.random() < pRn:
                decay += 1
                totr+=r
        N222Rn -= decay # subtract number of decayed Parent atoms
        N218Po += decay # add number of decayed Daughter atoms
    alphaenergy.append(totalpha)
    betaenergy.append(totbeta)
    renergy.append(totr)
    zenergy.append(totz)

#print(totalpha, totbeta, totr, totz)
print(alphaenergy)
print(betaenergy)
print(renergy)
print(zenergy)
```

```
[99552, 99880, 199272, 199272, 199216, 199388, 199364, 199256, 199196, 199268]
[149505, 149628, 74637, 74619, 74616, 74709, 74661, 74622, 74577, 74637]
[225324, 225693, 225261, 225288, 225288, 225225, 225324, 225144, 225414, 22524
3]
[48, 48, 48, 24, 64, 64, 24, 16, 40, 48]
```

In [10]:
```
# use the list of alpha, beta, r, and z energy to calculate the total and indivic

asum=0
sadevarr=[]
sadevseries=0
s=0
for w in range(0, len(alphaenergy)):
    asum+=alphaenergy[w]
avgalpha=asum/len(alphaenergy)
print("Average of alpha energy", avgalpha, "MeV")
for num in range(0, len(alphaenergy)):
    sadevseries=(alphaenergy[num]-avgalpha)**2
    sadevarr.append(sadevseries)
    s+=((1/(len(alphaenergy)-1))*(sadevarr[num]))
sdev=math.sqrt(s)
print("The average of alpha energy is ", avgalpha,"Mev and the standard deviatior
#This is correct!
print(asum)
```

```
Average of alpha energy 179366.4 MeV
The average of alpha energy is  179366.4 Mev and the standard deviation of alph
a energy is  41979.55858960141
1793664
```

In [4]:
```
bsum=0
sbdevarr=[]
sbdevseries=0
s=0
for x in range(0, len(betaenergy)):
    bsum+=betaenergy[x]
avgbeta=bsum/(len(betaenergy))
print("Average of beta energy", avgbeta, "MeV")
for n in range(0, len(betaenergy)):
    sbdevseries=(betaenergy[n]-avgbeta)**2
    sbdevarr.append(sbdevseries)
    s+=((1/(len(betaenergy)-1))*(sbdevarr[n]))
sbdev=math.sqrt(s)
print("The average of beta energy is ", avgbeta,"Mev and the standard deviation c
```

```
Average of beta energy 89621.1 MeV
The average of beta energy is  89621.1 Mev and the standard deviation of beta e
nergy is  31594.031304979108
```

In [7]:
```
rsum=0
sdevarr=[]
sdevseries=0
s=0
end=len(renergy)
for y in range(0, end):
    rsum+=renergy[y]
avgr=rsum/len(renergy)

for num in range(0, end):
    sdevseries=(renergy[num]-avgr)**2
    sdevarr.append(sdevseries)
    s+=((1/(end-1))*(sdevarr[num]))
srdev=math.sqrt(s)
print("The average of r energy is ", avgr,"Mev and the standard deviation of r er
```

```
The average of r energy is  225320.4 Mev and the standard deviation of r energy
is  148.8677265225744
```

In [8]:
```
zsum=0
szdevarr=[]
szdevseries=0
s=0
for z in range(0,len(zenergy)):
    zsum+=zenergy[z]
avgz=zsum/len(zenergy)
#print("Average of z energy", avgz, "MeV")
for n in range(0, len(zenergy)):
    szdevseries=(zenergy[n]-avgz)**2
    szdevarr.append(szdevseries)
    s+=((1/(len(zenergy)-1))*(szdevarr[n]))
szdev=math.sqrt(s)
print("The average of z energy is ", avgz,"Mev and the standard deviation of z er
#correct dev!
```

```
The average of z energy is  42.4 Mev and the standard deviation of z energy is
16.460052652811694
```

In [9]:
```
#part 4
length=0
shield= avgalpha+(3* sdev)
print(shield)
shieldpts= arange(0.0,shield,10000)
for s in shieldpts:
    length+=1

print(length)
#This is Correct
```

```
305305.07576880424
31
```

In [ ]:
```
# average and standard deviations of the total and individual energies produced w
#199300 74649 32 225189
#499170

#199296 74763 64 225423
#499546
```