In-class exercise for tutorial 010

In this exercise we will practice making datasets. We will on the one hand simulating data, on the other hand not quite as we will be making some asumptions and simplifaction on the data generation process that will not make the data.

Nestor lives right under the ATX airport, along the routes of airliners from major companies (United, Delt, etc).

Because of that Nestor hears noise. The noise is generated by the airplanes landing and taking off. There is a lot of noise that is generated by these airplanes. One day Nestor decides to simulate the noise generated by the airplanes and reaching his years. [Nestor is worried about his hearing loss (https://www.cdc.gov/niosh/topics/aircrew/noise.html)](https://www.cdc.gov/niosh/topics/aircrew/noise.html).

The problem to simulate

Nestor is interested in calculating how much noise he is being exposed to. So he goes and looks up the timetable during the 3 hours window in the morning, when he is at home, and during the 7 hours window in the evening, when he is at hom, and before midnight when the airport shuts down and no more flights land or takeoff.

Nestor lives about 2 miles away from the airport. At that distance each jet generates about 75-80 dB (Decibels). We will say 75dB. In the morning there are about 45 airplanes that land/takeoff. Each airplane can be heard consistently for about 7 minutes (we will assume a flat top distribution of the dBs, no ramp up, no decay, a simple flat distribution).

In the 3 hours window of the morning, Airplanes depart and land every 3 minutes, so their noise overlap for about 4 minutes. The dB of an airplane is corrupted by noise due to the city and nature around around, the cars, trains, trucks (all add some noise, randomly) and the position of the moving cloud in the sky, the wind and humidity (all diminish the noise randomly). So the noise is never 75dB but it stays on average around 75 dB while being corrupted by noise.

In the 7 hours window of the evening, Airplanes depart and land about every 4 minutes, so their noise overlaps for about 3 minutes.

Nestor will assume a linear summation of the airplane noise in a single day. This is not the best way especially when dealing with dB, but this is a simple exercise and we can break some fundamental physics rules to make things easier for us. We will want to simulate the situation.

How many airplanes depart/land in the morning widow of a single day. How many in the evening window?

In [2]:

```python
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-poster') # picking our preferred plotting style
```

In [3]:

```
#descriptive names and values
noysLevel = 75 #in decibles
noysVar = 2 # guess about variability

noisePerPlane = 7 #length of noise in minutes
plnPerMinMorn = 3 #in minutes
plnPerMinEve = 4 #in minutes

morningInMin = 180 #3 hours * 60 minutes
eveningInMin = 420 #7 hours * 60 minutes

numPlanesMorn = 60
numPlanesEve = 105

totalPlanes = numPlanesMorn+numPlanesEve

print(numPlanesMorn, "planes in the morning and ", numPlanesEve, "planes in t
```

```
60 planes in the morning and  105 planes in the evening
```

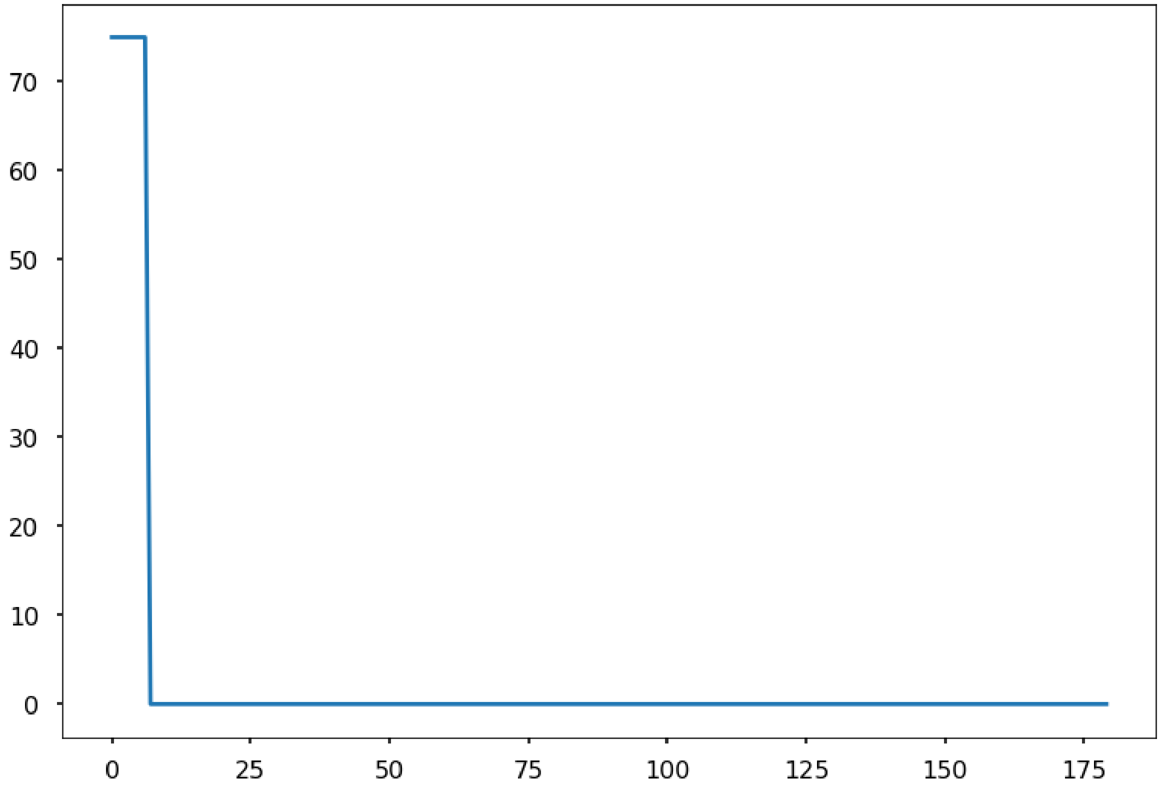
Show the noise profile of one airplane in the morning (pure for the moment no corruption but other exsternal noise)

In [4]:

```
#First we make a vector of the noise level over time
#This method makes room for only one of the plane data
plane1Noys = np.zeros([morningInMin, 1])
plane1Noys[0:noisePerPlane,:] = noysLevel
plt.plot(plane1Noys)
```

Out[4]:

```
[<matplotlib.lines.Line2D at 0x1a8ee5ab580>]
```



In [5]:

```
#First we make a vector of the noise level over time
#This method makes room for all the plane data
time = np.zeros((morningInMin, numPlanesMorn))
time.shape
```

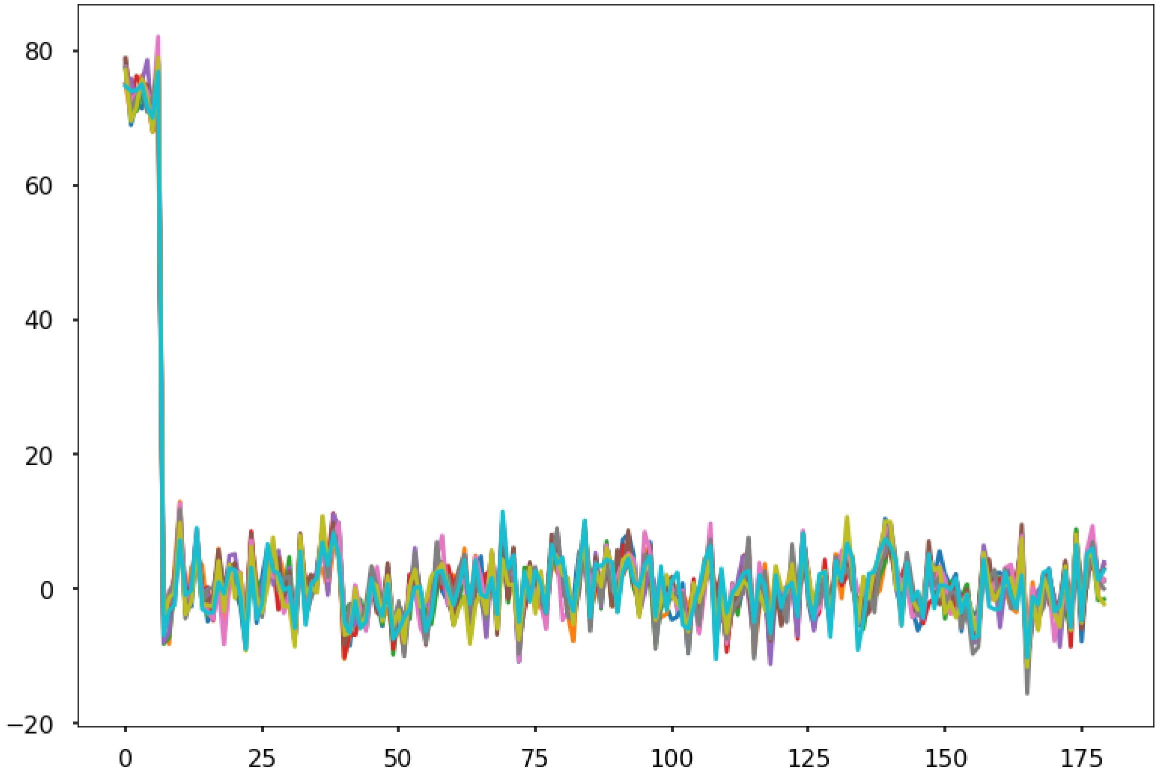
```
Out[5]: (180, 60)
```

Type *Markdown* and LaTeX: α^2

Show the noise profile of the same airplane corrupted by some small noise.

In [32]:

```
#environmental noise
envNoys = noysVar*np.random.randn(morningInMin, 1)
plane1Noys = plane1Noys + envNoys
plt.plot(plane1Noys);
```



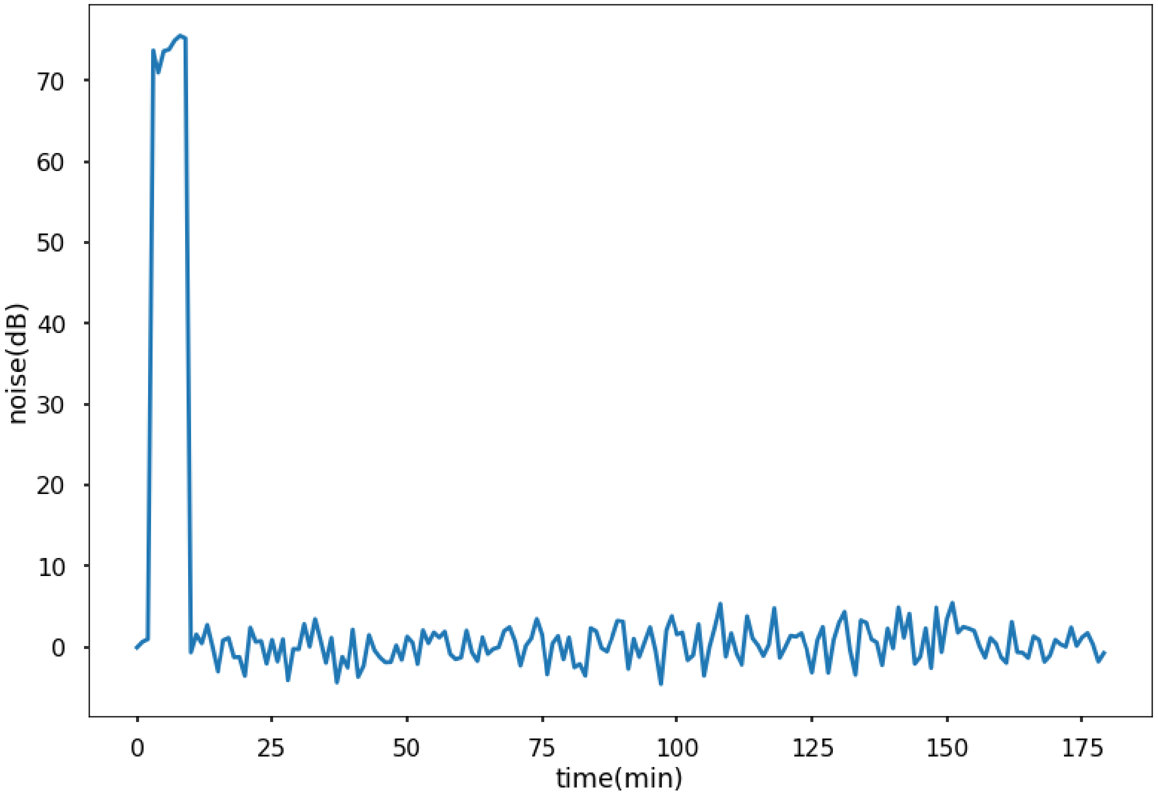
Now add a second airplane, corrupted by noise but departing/landing at a different time. Make a plot of the two airplanes together.

In [33]:

```
#noisePerPlane = 7 #length of noise in minutes
#plnPerMinMorn = 3 #in minutes
#plnPerMinEve = 4 #in minutes
#morningInMin = 180 #3 hours * 60 minutes
#eveningInMin = 420 #7 hours * 60 minutes

start = plnPerMinMorn # second plane noise should start 3 minutes in
stop = start + noisePerPlane # stop 7 minutes later

plane2Noys = np.zeros([morningInMin, 1])
plane2Noys[start:stop, :] = noysLevel

envNoys = noysVar*np.random.randn(morningInMin, 1)
plane2Noys = plane2Noys + envNoys

#hstack takes 2 arrays and tries to stack horizontal - have to have the same
planeNoys = np.hstack((plane1Noys, plane2Noys))

plt.plot(plane2Noys);
plt.xlabel('time(min)')
plt.ylabel('noise(dB)')
```

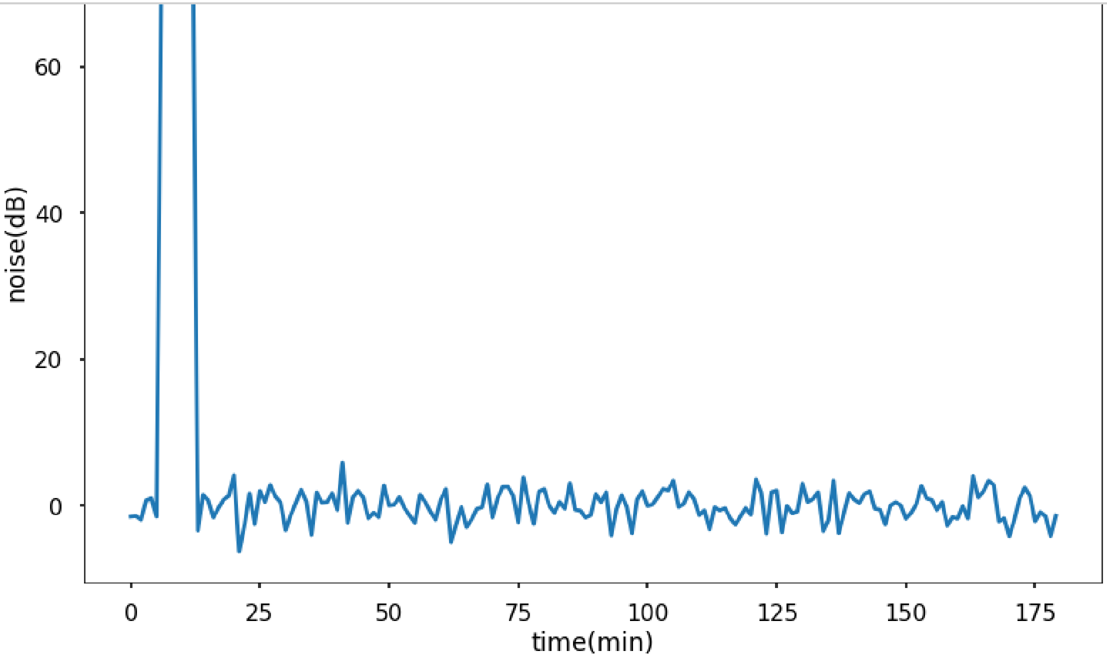
Out[33]:

```
Text(0, 0.5, 'noise(dB)')
```



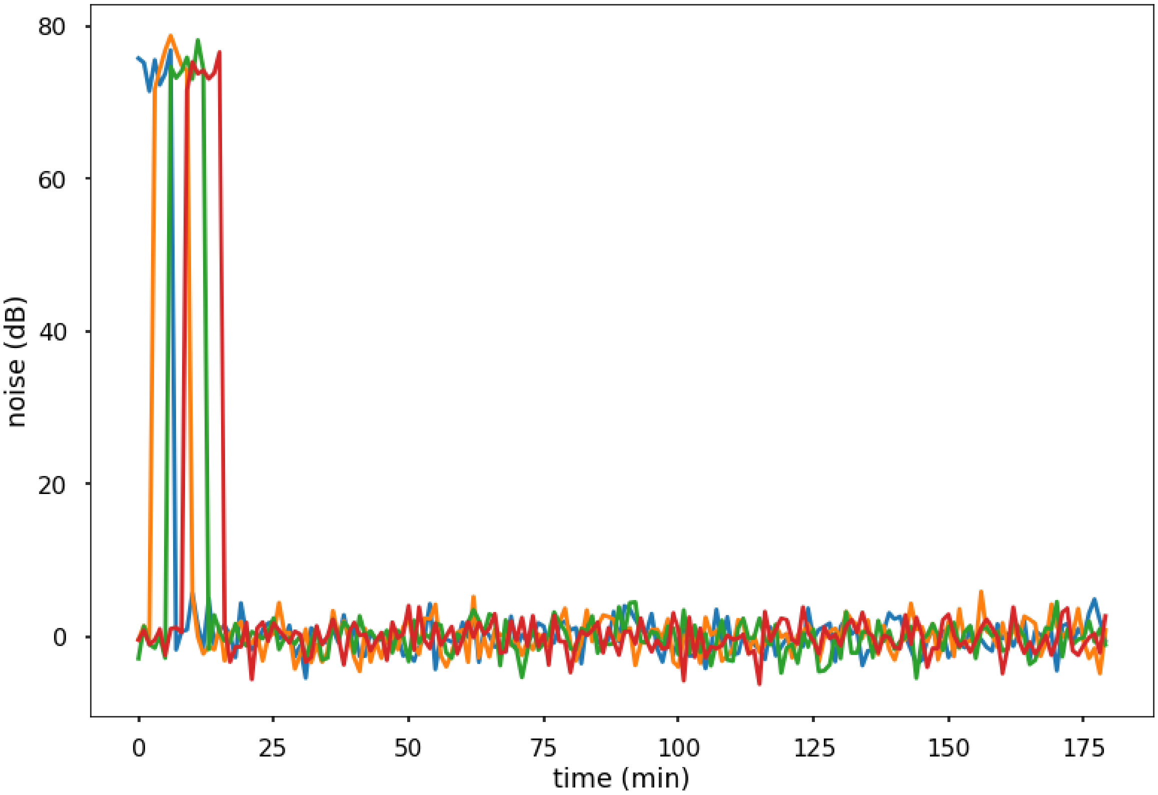
In [34]:

```
#third plane
planeIndex = 2
start = planeIndex*plnPerMinMorn
stop = start + noisePerPlane

plane3Noys = np.zeros([morningInMin, 1])
plane3Noys[start:stop, :] = noysLevel
envNoys = noysVar*np.random.randn(morningInMin, 1)
plane3Noys = plane3Noys + envNoys

planeNoys = np.hstack((plane2Noys, plane3Noys)) # add new plane as column to

plt.plot(plane3Noys);
plt.xlabel('time(min)')
plt.ylabel('noise(dB)')
```



In [9]:

```
# Make the fourth plane using a genertic "nextPlaneNoys" array

# figure out start and stop
planeIndex = 3  # THIS IS THE ONLY THING WE SHOULD NEED TO CHANGE NOW
start = planeIndex*plnPerMinMorn  # noise should start 6 min in for 3rd plane
stop = start + noisePerPlane      # and always stops 7 minutes later

# make the noise profile
nextPlaneNoys = np.zeros([morningInMin, 1])         # make our array
nextPlaneNoys[start:stop, :] = noysLevel        # add the plane noise
envNoys = noysVar*np.random.randn(morningInMin, 1)  # delicious randomness th
nextPlaneNoys = nextPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
planeNoys = np.hstack((planeNoys,  nextPlaneNoys)) # add new plane as column

# and plot
plt.plot(planeNoys); # and plot (use the terminal semicolon so you get ONLY t
plt.xlabel('time (min)');
plt.ylabel('noise (dB)');
```



In [10]:

```
#### NEXT PLANE ####
planeIndex = 4
start = planeIndex*plnPerMinMorn
stop = start + noisePerPlane

#make the noise profile
nextPlaneNoys = np.zeros([morningInMin, 1])
nextPlaneNoys[start:stop,:] = noysLevel
envNoys = noysVar*np.random.randn(morningInMin, 1)
nextPlaneNoys = nextPlaneNoys + envNoys

planeNoys = np.hstack((planeNoys, nextPlaneNoys))

#### NEXT PLANE ####
planeIndex = 5
start = planeIndex*plnPerMinMorn
stop = start + noisePerPlane

#make the noise profile
nextPlaneNoys = np.zeros([morningInMin, 1])
nextPlaneNoys[start:stop,:] = noysLevel
envNoys = noysVar*np.random.randn(morningInMin, 1)
nextPlaneNoys = nextPlaneNoys + envNoys

planeNoys = np.hstack((planeNoys, nextPlaneNoys))

#### NEXT PLANE ####
planeIndex = 6
start = planeIndex*plnPerMinMorn
stop = start + noisePerPlane

#make the noise profile
nextPlaneNoys = np.zeros([morningInMin, 1])
nextPlaneNoys[start:stop,:] = noysLevel
envNoys = noysVar*np.random.randn(morningInMin, 1)
nextPlaneNoys = nextPlaneNoys + envNoys

planeNoys = np.hstack((planeNoys, nextPlaneNoys))

#### NEXT PLANE ####
planeIndex = 7
start = planeIndex*plnPerMinMorn
stop = start + noisePerPlane

#make the noise profile
nextPlaneNoys = np.zeros([morningInMin, 1])
nextPlaneNoys[start:stop,:] = noysLevel
envNoys = noysVar*np.random.randn(morningInMin, 1)
nextPlaneNoys = nextPlaneNoys + envNoys

planeNoys = np.hstack((planeNoys, nextPlaneNoys))

#### NEXT PLANE ####
planeIndex = 8
start = planeIndex*plnPerMinMorn
stop = start + noisePerPlane
```

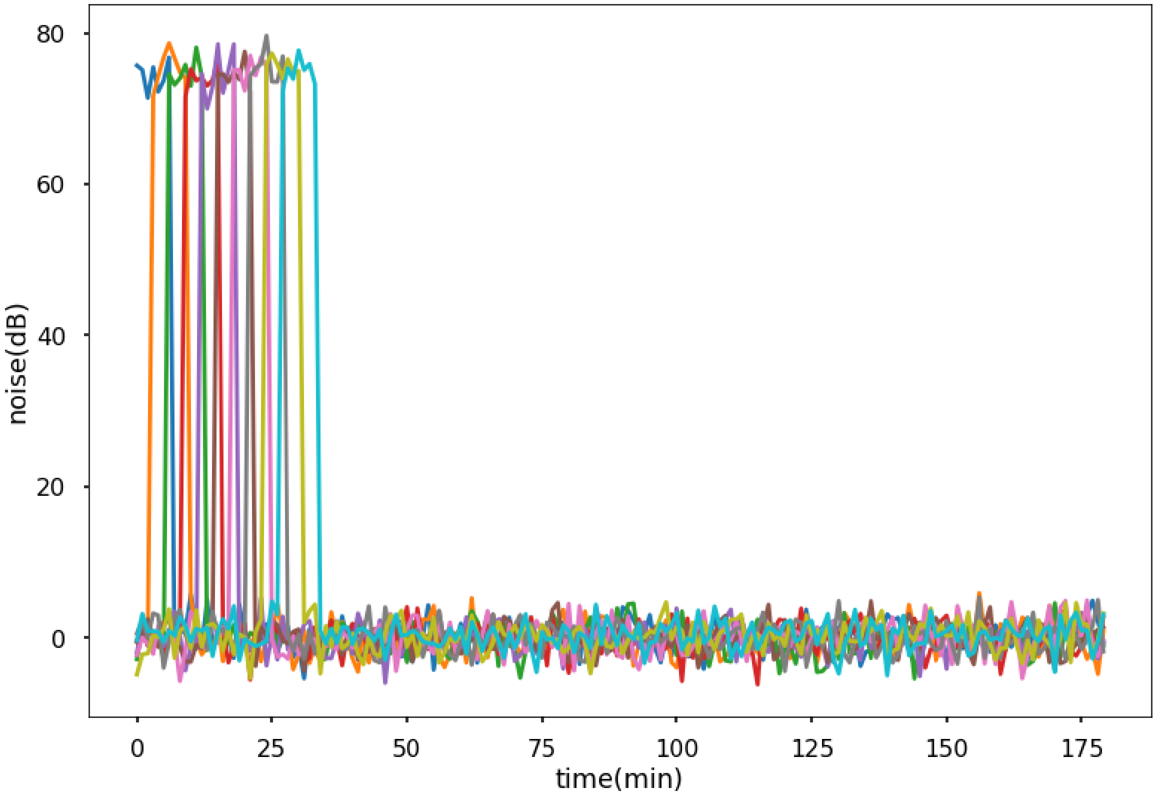
```
#make the noise profile
nextPlaneNoys = np.zeros([morningInMin, 1])
nextPlaneNoys[start:stop,:] = noysLevel
envNoys = noysVar*np.random.randn(morningInMin, 1)
nextPlaneNoys = nextPlaneNoys + envNoys

planeNoys = np.hstack((planeNoys, nextPlaneNoys))

#### NEXT PLANE ####
planeIndex = 9
start = planeIndex*plnPerMinMorn
stop = start + noisePerPlane

#make the noise profile
nextPlaneNoys = np.zeros([morningInMin, 1])
nextPlaneNoys[start:stop,:] = noysLevel
envNoys = noysVar*np.random.randn(morningInMin, 1)
nextPlaneNoys = nextPlaneNoys + envNoys

planeNoys = np.hstack((planeNoys, nextPlaneNoys))
```

In [11]:

```
plt.plot(planeNoys);
plt.xlabel('time(min)')
plt.ylabel('noise(dB)')
```

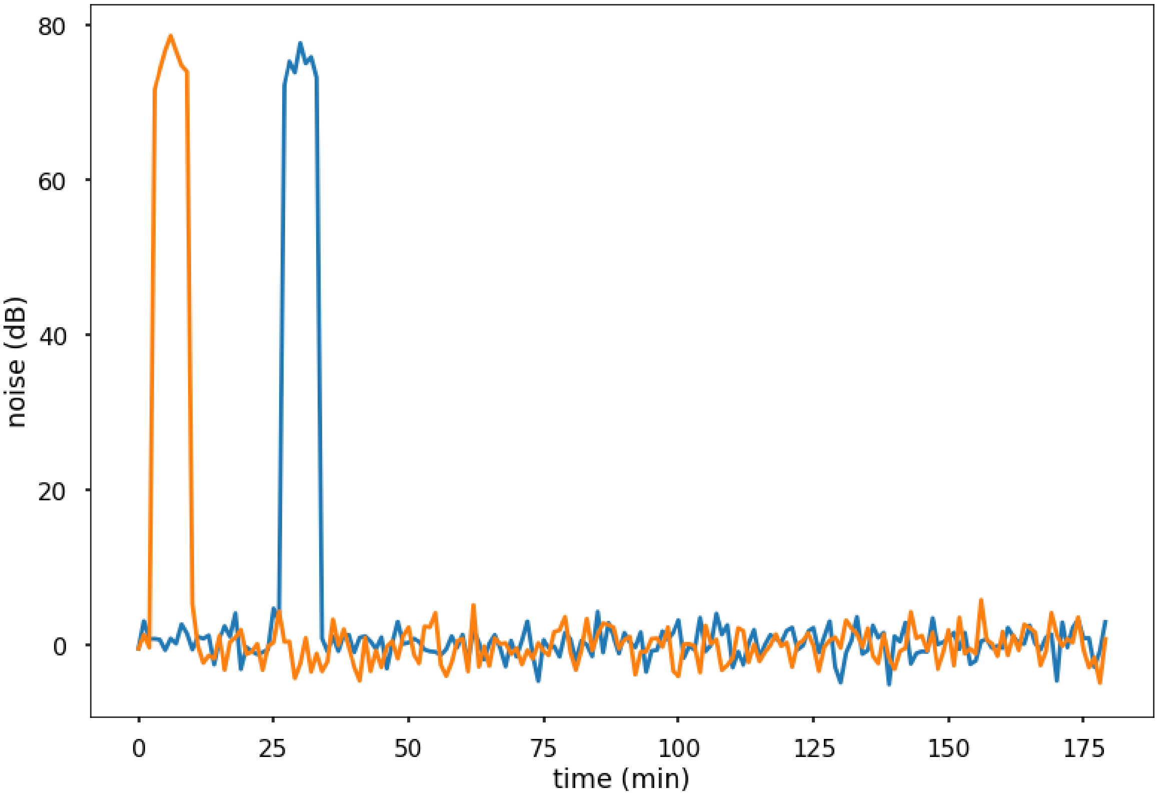
Out[11]:

```
Text(0, 0.5, 'noise(dB)')
```



In [12]:

```
plt.plot(planeNoys[:,-1]);
plt.plot(planeNoys[:,1]);
plt.xlabel('time (min)');
plt.ylabel('noise (dB)');
```

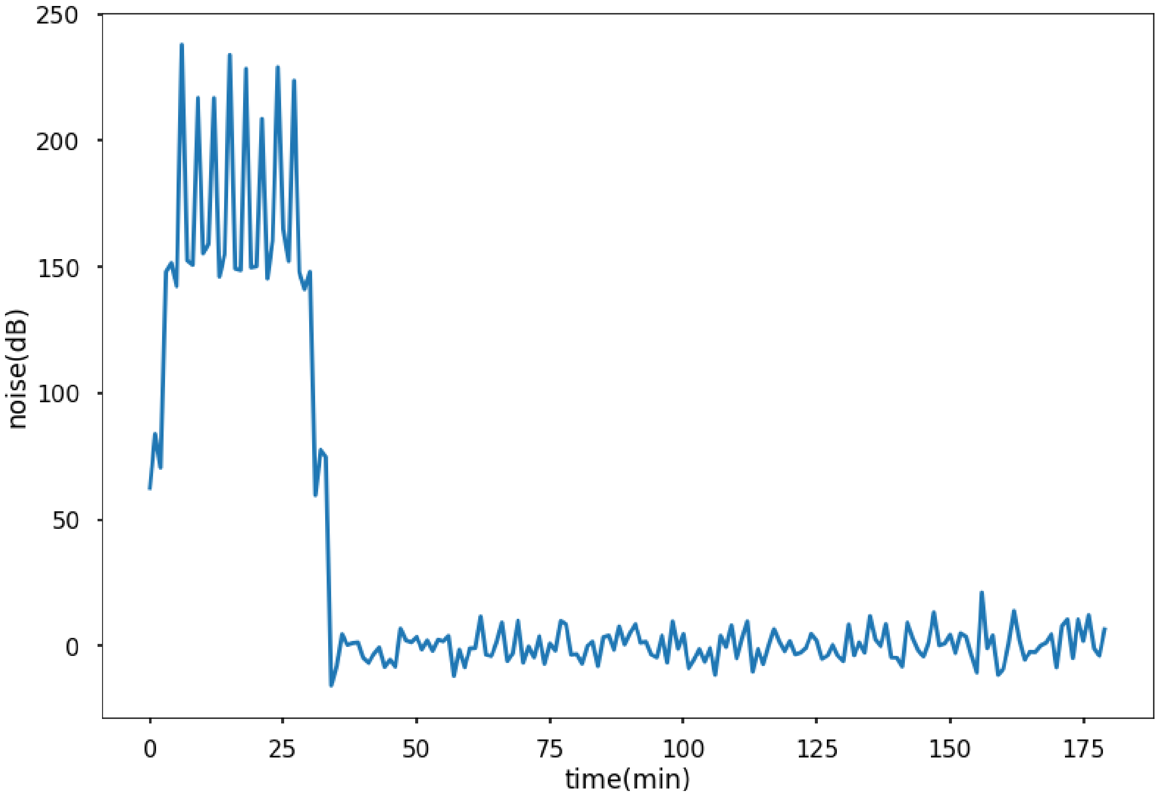


In [13]:

```
# plot the sum of all the noise
plt.plot(np.sum(planeNoys, 1));
plt.xlabel('time(min)')
plt.ylabel('noise(dB)')
```

Out[13]:

```
Text(0, 0.5, 'noise(dB)')
```



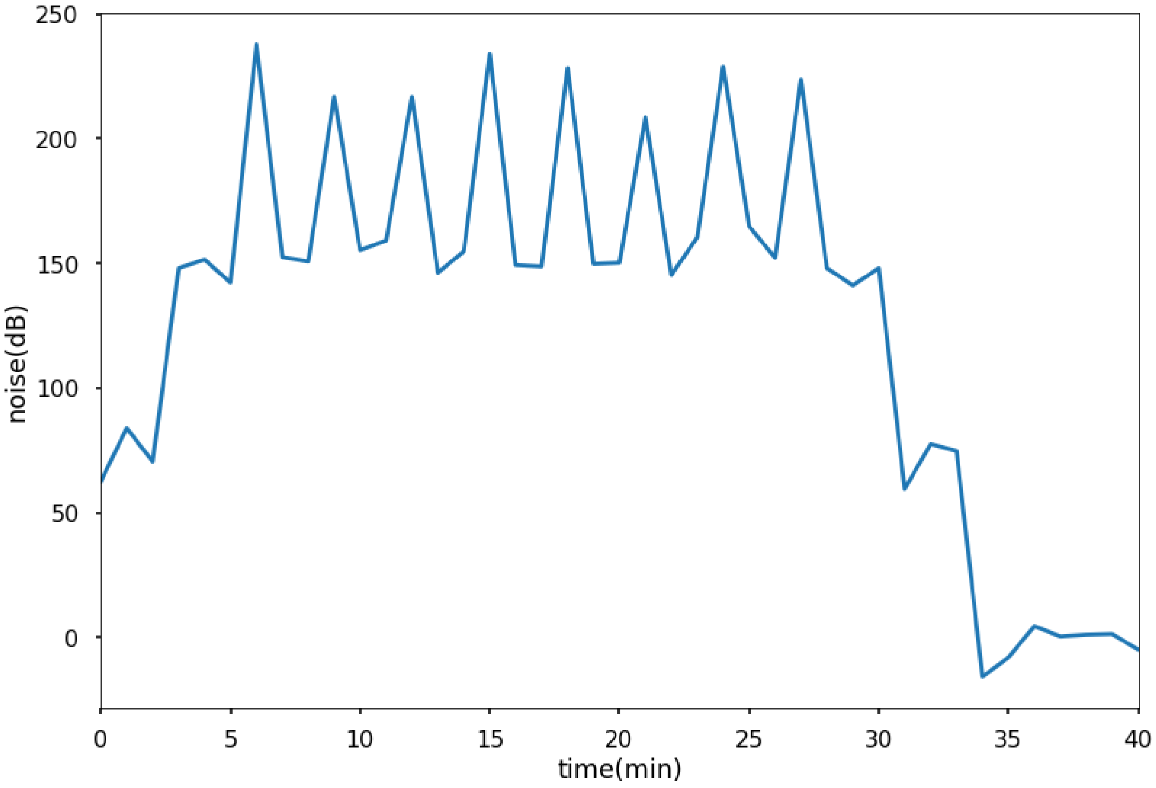
Finally, simulate all the the airplanes that you have estimated to land/depart in the morning. Plot them on the same figure.

In [14]:

```
plt.plot(np.sum(planeNoys,1));
plt.xlabel('time(min)')
plt.ylabel('noise(dB)')
plt.xlim([0,40])
```

Out[14]:

```
(0.0, 40.0)
```



So here's our mission: See if you can write some similar code in which you can generate 10 planes of noise data in a single 180x10 numpy array, where only one number needs to change across planes. Then compute the total noise over time and make sure you get something like the above.

In [37]:

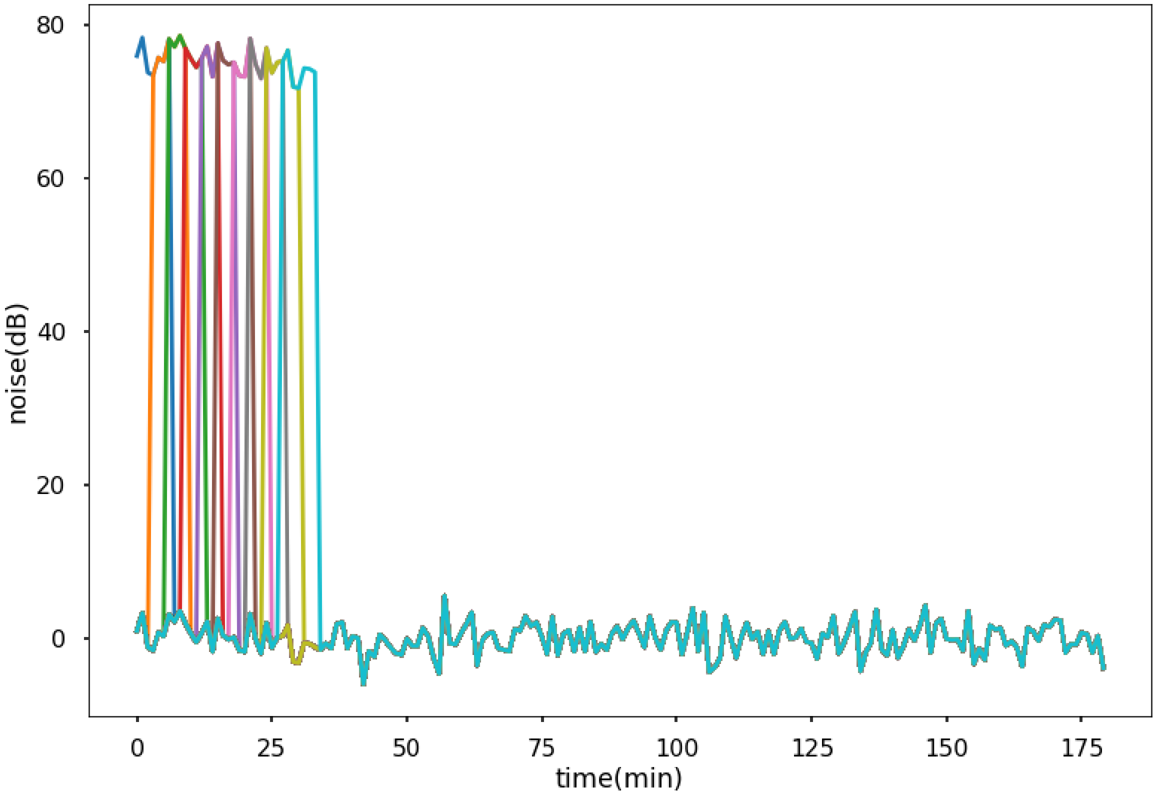
```
#This method makes room for all the plane data (10 planes in this case)
planeNoys = np.zeros([morningInMin, 10])
envNoys = noysVar*np.random.randn(morningInMin, 1)
planeNoys = planeNoys + envNoys

############ NEXT PLANE ##############
planeIndex = 0
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 1
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 2
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 3
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness
```

```
# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 4
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 5
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 6
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 7
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness
```

```
# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 8
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 9
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))
```

```
In [39]:  plt.plot(planeNoys);
          plt.xlabel('time(min)')
          plt.ylabel('noise(dB)')
```

```
Out[39]: Text(0, 0.5, 'noise(dB)')
```



Brain challange: we have used different "environmental" noise for each plane over the same 180 minutes? Is this reasonable? Or might there be environmental noise that should be common to all of the planes' noise profiles?

The environmental noise is shared with all noises, in our case, the plane noise; therefore, the environmental noise should be shared with each plane rather than each plane having its own environmental noise

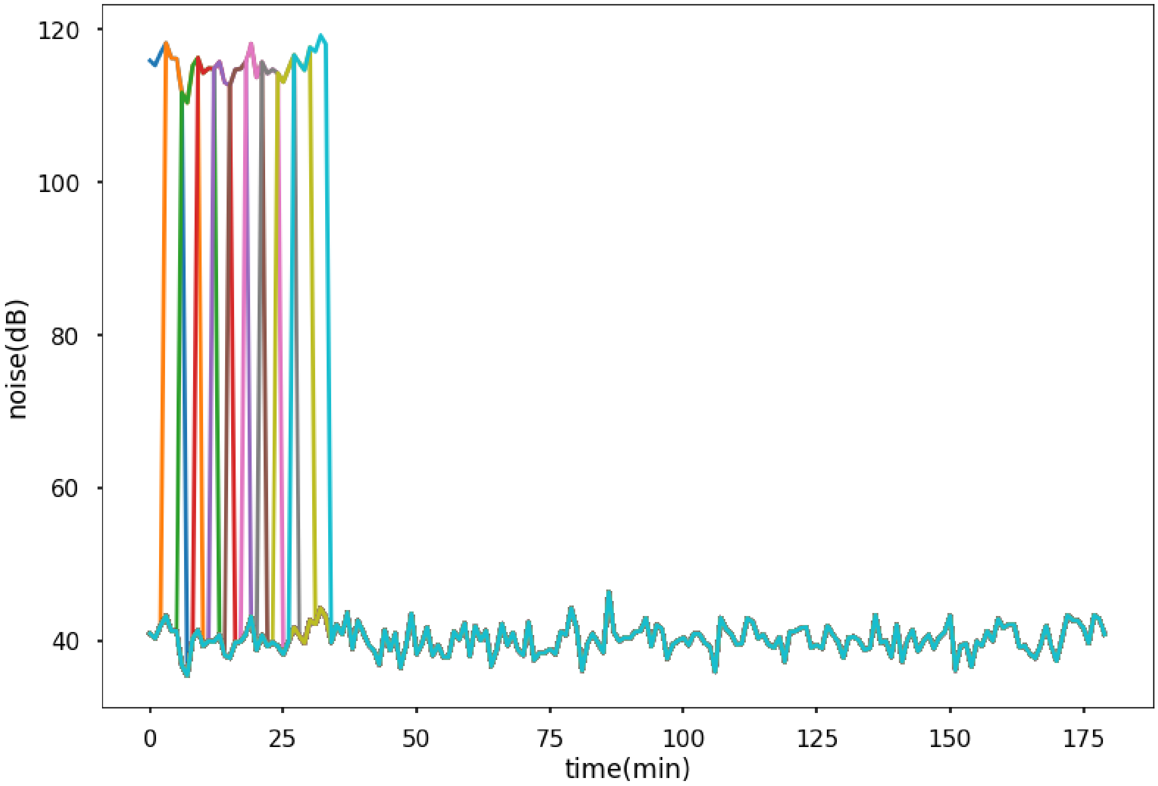
In [40]:

```
#This method makes room for all the plane data (10 planes in this case)
planeNoys = np.zeros([morningInMin, 10])
envNoys = noysVar*np.random.randn(morningInMin, 1) +40
planeNoys = planeNoys + envNoys

############ NEXT PLANE ##############
planeIndex = 0
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 1
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 2
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 3
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness
```

```
# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 4
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 5
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 6
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 7
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness
```

```
# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 8
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))

############ NEXT PLANE ##############
planeIndex = 9
start = planeIndex*plnPerMinMorn  # noise should start 3 min in for each plan
stop = start + noisePerPlane    # and always stops 7 minutes later

#noise profile - 1 array with 10 planes
myPlaneNoys = np.zeros([morningInMin, 10])     # make our array
myPlaneNoys[start:stop,planeIndex] = noysLevel     # add the plane noise
nextPlaneNoys = myPlaneNoys + envNoys         # add life's randomness

# append it to our "all planes" array
#nextPlaneNoys
planeNoys = np.hstack((planeNoys, nextPlaneNoys))
```

```
In [41]:  plt.plot(planeNoys);
          plt.xlabel('time(min)')
          plt.ylabel('noise(dB)')

Out[41]:  Text(0, 0.5, 'noise(dB)')
```



```
In [ ]:
```