In this assignment, you are asked to actually implement the EM algorithm derived in class. Your implementation will be using Python. Recall that the basic setup was that we imagine that there are two coins in a bag. Repeatedly, we pick one out and flip it 10 times, then put it back in. We derived an algorithm to look at all of the sequences of 10 flips, and figure out the probability that each coin comes up heads.

Start with the following code:

```
import numpy as np
import scipy.stats

# one coin has a probability of coming up heads of 0.2, the other 0.6
coinProbs = np.zeros (2)
coinProbs[0] = 0.2
coinProbs[1] = 0.6

# reach in and pull out a coin numTimes times
numTimes = 100

# flip it numFlips times when you do
numFlips = 10

# flips will have the number of heads we observed in 10 flips for each coin
flips = np.zeros (numTimes)
for coin in range(numTimes):
        which = np.random.binomial (1, 0.5, 1);
        flips[coin] = np.random.binomial (numFlips, coinProbs[which], 1);

# initialize the EM algorithm
coinProbs[0] = 0.79
coinProbs[1] = 0.51

# run the EM algorithm
for iters in range (20):
```

Using this code as a start, write some Python code that runs 20 iterations of the EM algorithm that we derived. At the end of each iteration, print out the current probabilities of the two coins.

I'm going to ask you to run your code twice. In the first run, you'll set numTimes to 100, and numFlips to 10. In the second, reduce numFlips to 2.

Here's what I get for the first run:

```
[ 0.70187489   0.34086585]
[ 0.65524938   0.27485199]
[ 0.63212797   0.24290806]
[ 0.62024191   0.2275406 ]
[ 0.61411395   0.22006691]
[ 0.6109907    0.21638431]
[ 0.60941259   0.21455518]
[ 0.60861893   0.21364295]
[ 0.6082207    0.21318709]
[ 0.6080211    0.21295906]
[ 0.60792112   0.21284495]
[ 0.60787105   0.21278783]
[ 0.60784598   0.21275924]
[ 0.60783343   0.21274492]
[ 0.60782714   0.21273776]
[ 0.60782399   0.21273417]
[ 0.60782242   0.21273237]
[ 0.60782163   0.21273147]
[ 0.60782123   0.21273102]
[ 0.60782104   0.2127308 ]
```

Note that your results might be different, since the data are randomly generated. And for the second:

```
[ 0.5594007    0.26368322]
[ 0.51265313   0.23194679]
[ 0.50398287   0.22817581]
[ 0.50172385   0.22853859]
[ 0.50059052   0.22939705]
[ 0.49974148   0.23021383]
[ 0.49903154   0.23092592]
[ 0.49842427   0.23153913]
[ 0.49790197   0.23206693]
[ 0.49745175   0.2325218 ]
[ 0.49706308   0.23291436]
[ 0.49672714   0.23325357]
[ 0.49643647   0.23354699]
[ 0.49618474   0.23380104]
[ 0.49596657   0.23402119]
[ 0.49577736   0.23421209]
[ 0.49561316   0.23437773]
[ 0.4954706    0.23452153]
[ 0.49534677   0.23464641]
[ 0.49523918   0.23475492]
```

Remarkably, the EM algorithm does a pretty job, even with just two flips of each coin!!

One thing that might help you is scipy.stats.binom.pmf (numHeads, numTrials, probOfHeads). This allows you to compute the binomial probability of seeing the specified number of heads in the specified number of trials.

When you are done, turn in your code as well as your results (like my results above). You can produce a pdf document, a text document, or simply entry your text results directly into Canvas.