```
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
```

```
<ipython-input-21-067bc48fc9ca>:16: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated,
  flips[coin] = np.random.binomial(numFlips, coinProbs[which], 1);
```

```
# EM algorithm
# Profs solution
for iters in range(20):
    # Expectation step
    p = np.zeros((numTimes, 2))
    for ind in range(numTimes):
      lik_probs1 = scipy.stats.binom.pmf(flips[ind], numFlips, coinProbs[0]);
      #print(lik_probs1)
      lik_probs2 = scipy.stats.binom.pmf(flips[ind], numFlips, coinProbs[1]);
      p[ind, 0] = lik_probs1 / (lik_probs1 + lik_probs2)
      p[ind, 1] = lik_probs2 / (lik_probs1 + lik_probs2)
    # Maximization step
    for coin in range(2):
      coinProbs[coin] = np.sum(p[:,coin] * flips) / np.sum(p[:, coin] * numFlips)

    print(f"Iteration {iters + 1}: [{coinProbs[0]}, {coinProbs[1]}]")
```

```
Iteration 1: [0.7171611047215931, 0.3480038355906213]
Iteration 2: [0.6746065098982565, 0.29419864176905836]
Iteration 3: [0.6501879752764033, 0.27143569001643364]
Iteration 4: [0.6370979018370088, 0.2607035588163036]
Iteration 5: [0.6302854540755475, 0.25540057463571664]
Iteration 6: [0.6267817831889948, 0.2527289151696926]
Iteration 7: [0.6249881398810178, 0.2513722262861108]
Iteration 8: [0.6240715146630693, 0.25068107878664725]
Iteration 9: [0.6236033776812191, 0.2503285267306966]
Iteration 10: [0.6233643441245721, 0.25014859655249594]
Iteration 11: [0.6232423008968215, 0.25005674667760425]
Iteration 12: [0.6231799906765804, 0.25000985536817305]
Iteration 13: [0.6231481778182321, 0.24998591543545]
Iteration 14: [0.6231319355787968, 0.2499736929130757]
Iteration 15: [0.6231236430044664, 0.24996745266205778]
Iteration 16: [0.6231194091776356, 0.24996426666918564]
Iteration 17: [0.6231172475694099, 0.2499626400411405]
Iteration 18: [0.6231161439457956, 0.2499618095556389]
Iteration 19: [0.6231155804832599, 0.24996138554571487]
Iteration 20: [0.6231152928036116, 0.24996116906457272]
```

```
# EM Algo: Iter 2
numFlips = 2
# initialize the EM algorithm
coinProbs[0] = 0.79
coinProbs[1] = 0.51

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);
```

```
for iters in range(20):
    # Expectation step
    p = np.zeros((numTimes, 2))
    for ind in range(numTimes):
      lik_probs1 = scipy.stats.binom.pmf(flips[ind], numFlips, coinProbs[0]);
      lik_probs2 = scipy.stats.binom.pmf(flips[ind], numFlips, coinProbs[1]);
      p[ind, 0] = lik_probs1 / (lik_probs1 + lik_probs2)
      p[ind, 1] = lik_probs2 / (lik_probs1 + lik_probs2)
    # Maximization step
    for coin in range(2):
      coinProbs[coin] = np.sum(p[:,coin] * flips) / np.sum(p[:, coin] * numFlips)

    print(f"Iteration {iters + 1}: [{coinProbs[0]}, {coinProbs[1]}]")
```

```
<ipython-input-31-7e0dfb786a29>:10: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated,
  flips[coin] = np.random.binomial(numFlips, coinProbs[which], 1);
Iteration 1: [0.7601897719598969, 0.45660774240161867]
Iteration 2: [0.7579690270744084, 0.4448728081075841]
Iteration 3: [0.7600194104136742, 0.4403412166850563]
Iteration 4: [0.7624088709929774, 0.437507575599772]
Iteration 5: [0.7645007649533316, 0.43535002735244505]
Iteration 6: [0.766232214117418, 0.43362507706421455]
Iteration 7: [0.767640598060548, 0.4322360258322992]
Iteration 8: [0.7687768442129229, 0.4311198059532618]
Iteration 9: [0.769688703815167, 0.43022603416921235]
Iteration 10: [0.7704176060968607, 0.42951275545389556]
Iteration 11: [0.7709984581584816, 0.4289450915519442]
Iteration 12: [0.7714601922342108, 0.42849431931179965]
Iteration 13: [0.771826516430134, 0.4281370017352559]
Iteration 14: [0.772116691677583, 0.42785416043245206]
Iteration 15: [0.7723462618888643, 0.4276305200689729]
Iteration 16: [0.7725277058818839, 0.4274538439669106]
Iteration 17: [0.7726710007042236, 0.4273143659060825]
Iteration 18: [0.7727840974330835, 0.4272043139298379]
Iteration 19: [0.7728733164557879, 0.4271175171378062]
Iteration 20: [0.772943671839096, 0.42704908456779345]
```