Exercise 2

Part A:

- The unobservable set of variables given by the example in the book is $Rain_t$
- The observable set of variables given by the example in the book is $Umbrealla_t$
- [0.7 0.3] Represents the dynamic model $P(X_t|X_{t-1})$
  [0.3 0.7]
  [0.9 0.0] (U_t = True) [0.1 0.0] (U_t = False) Represents the observation model $P(E_t|X_t)$
  [0.0 0.2] [0.0 0.8]
- We assume the process is stationary as in the laws of that govern the transition of states remain the same over all time steps. This is because we assume that the probability of rain today given how it rained yesterday remains the same. In addition to this, we assume that rain today only depend on rain yesterday (first order markov process).

Part B:

- Review code from partB.py, as well as common.py which has most of the general implementtations. Result on day two was indeed 0.883, or rather 0.88335704125177805 (the exact output from my program).
- Review code ffrom partB.py, as wel as common.py which has most of the general implementations. Result is a follows:

	Rain:	Not Rain:
Day 0:	0.5	0.5
Day 1:	0.81818181818181812	0.1818181818181818
Day 2:	0.88335704125177805	0.11664295874822191
Day 3:	0.19066793972352525	0.80933206027647475
Day 4:	0.73079400458498212	0.26920599541501788
Day 5:	0.86733888957548477	0.13266111042451526

Part C:

- Review code from partC.py, as well as common.py which has most of the general implementtations. Result for the first two days was indeed <0.883, 0.117>, or rather <0.88335704125177805, 0.11664295874822191> (the exact output from my program).
  Day 0: [[0.0443845671], [0.024222833899999997]]

- Review code from partC.py, as wel as common.py which has most of the general implementations. Result is a follows:

Rain:	Not Rain:
Day 1: 0.06611763	0.04550766999999999
Day 2: 0.090639	0.15025099999999997
Day 3: 0.4593	0.2437
Day 4: 0.69	0.41000000000000003
Day 5: 1	1

- Viterbi can be viewed as a slightly different take on the forward-backward algorithm. One can imagine sequences of probabilities as paths through a graph where the nodes are possible states in each time step.We want to find the most likely path through this space. The likelihood of a path is the product of the transition probabilities along the path as well as the given observations at each state.This means that if you want to find the most likely path to a given state, that is found recursively by finding the most likely state in the previous time step followed by a transition to the state in the current time step.
  We achieve this by altering the forward-backward algorithm by changing the forward message such that it rather maximizes probabilities up to the goal state (finding probabilities for the most likely path). In addition to this, we have to exchange the summation over x_t in the one step prediction with a maximization.
  When the algorithm is done it would have gone forward computing the “maximized” messages leading it find the most likely sequence to each of the final states. The sequence is identified by pointers to the previous state, thus one can find the best sequence by following the pointers backwards from the best goal state.