Hints for Breakout Task

```
# breakout.py
...
env = breakout_environment(5, 8, 3, 1, 2)     # don't use different parameters (but, you may want to try
                                              # simpler setups for testing and debugging your code)
x = tf.placeholder(tf.float32, shape=[None, env.ny, env.nx, env.nf])
...
for episode in range(n_episodes):
    s = env.reset()
    for t in range(max_steps):
        if np.random.rand() < epsilon:
            a = np.random.randint(env.na)       # random action
        else:
            q = sess.run(y_hat, feed_dict={x: np.reshape(s, [1, env.ny, env.nx, env.nf])})
            a = np.random.choice(np.where(q[0]==np.max(q))[0])
        sn, r, terminal, _, _, _, _, _, _, _, _ = env.run(a - 1)     # action to take is -1, 0, 1
        ...
```

Hints for Breakout Task

```
# Implement both target network Q̂ and Q network.
#   You can use sess.run(tf.assign(a,b)) to copy from tensorflow variable b to another tf variable a
# Don't skip frames. You can simply call env.run(a-1) once in each time step, then it will return
#   two consecutive frames without frame skipping, e.g., frames #1 and #2 are returned in 'sn' when
#   calling 'sn, r, terminal, _, _, _, _, _, _, _, _ = env.run(a - 1)' once and then frames #2 and #3 are
#   returned in 'sn' when calling it in the next time step.
# About 150 lines is enough to implement breakout.py. Use max_steps = 200.
# Since the problem is simple (input is only 8 pixels by 5 pixels by 2 frames), after training for
# 1 minute your agent may already achieve the maximum score of 15 (but may take many steps, say 150).
# Don't use the same hyperparameters as in DQN paper. For example, instead of 1,000,000 frames,
#   pick something like 1,000 (or 10,000) frames for the size of the replay memory.
# Don't train for hours just because you are not getting desired results.
# If you are not getting a high enough score (say > 10) after training for 1 minute, you may be
#   doing something wrong already and it is highly likely to be a waste of time to train with more
#   episodes or try many different combinations of hyperparameters.
# Try to find bugs by analyzing values of variables and by using simple test inputs.
```