In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
```

In [ ]:
```
import tensorflow as tf

def eval_plot(file):
    return get_section_results(file, 'Eval_AverageReturn')

def get_section_results(file, s):
    eval_returns = []
    for e in tf.train.summary_iterator(file):
        for v in e.summary.value:
            if v.tag == s:
                eval_returns.append(v.simple_value)
    return eval_returns
```

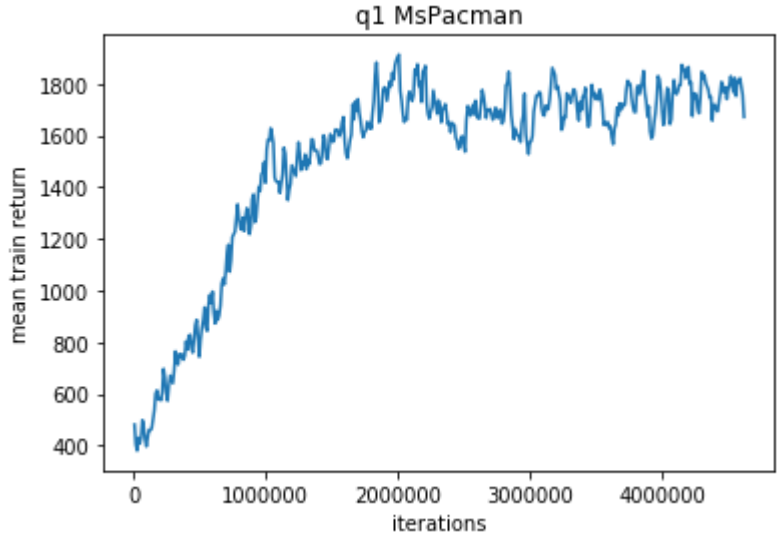
In [3]:
```
def for_section (question):
    experiments = os.listdir('data')
    rv = {}
    for exp in experiments:
        if exp[:len(question)] == question:
            exp_fn = 'data/' + os.path.join (exp, os.listdir('data/' + e
            print (exp)
            env_steps = get_section_results (exp_fn, 'Train_EnvstepsSoFa
            eval_avg_return = get_section_results (exp_fn, 'Train_Averag
            eval_std_return = get_section_results (exp_fn, 'Eval_Average
            rv[exp] = (env_steps, eval_avg_return, eval_std_return)
    return rv

def plot_section (exp_dict, title): # plot
    plt.title(title)
    plt.xlabel('iterations')
    plt.ylabel('mean eval return')
    for key in exp_dict.keys():
        plt.plot (exp_dict[key][1], label=key)
    plt.legend()
    plt.show()

def plot_single (exp, title):
    plt.title(title)
    plt.xlabel('iterations')
    plt.ylabel('mean eval return')
    plt.plot (exp[:1])
    plt.show()
```

In [4]:
```
rv = for_section('hw3_q1')
```

```
hw3_q1_MsPacman-v0_17-10-2020_01-04-13
```

q1

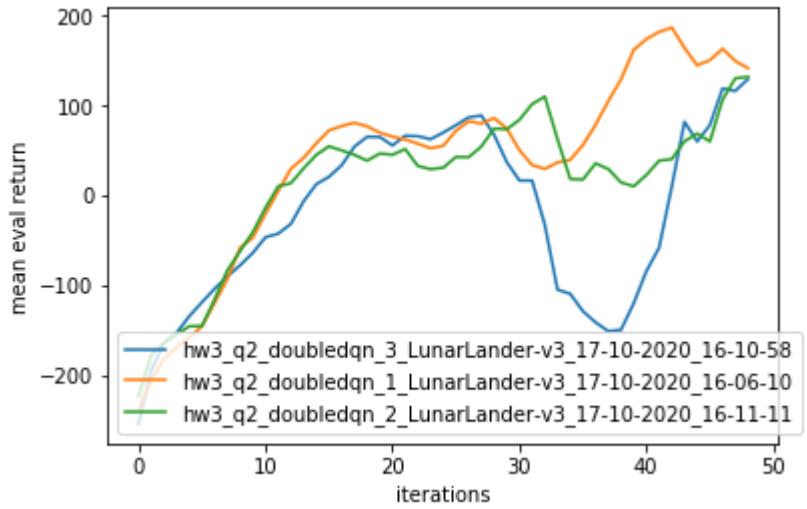
In [5]:
```
plt.title('q1 MsPacman')
plt.xlabel('iterations')
plt.ylabel('mean train return')
plt.plot (rv['hw3_q1_MsPacman-v0_17-10-2020_01-04-13'][0][1:], rv['hw3_q
plt.show()
```



In [15]:
```
double = for_section('hw3_q2_doubledqn')
dqn = for_section('hw3_q2_dqn')
```

```
hw3_q2_doubledqn_3_LunarLander-v3_17-10-2020_16-10-58
hw3_q2_doubledqn_1_LunarLander-v3_17-10-2020_16-06-10
hw3_q2_doubledqn_2_LunarLander-v3_17-10-2020_16-11-11
hw3_q2_dqn_3_LunarLander-v3_17-10-2020_16-10-47
hw3_q2_dqn_2_LunarLander-v3_17-10-2020_16-10-30
hw3_q2_dqn_1_LunarLander-v3_17-10-2020_16-06-46
```

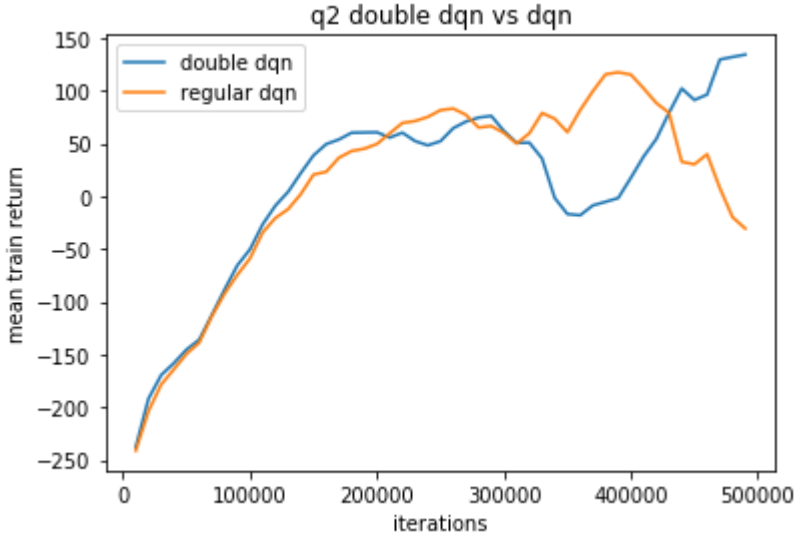
In [16]:
```
plot_section (double, '')
```



In [17]:
```
def avg(dic):
    arr = []
    for _, values in dic.items():
        arr.append (values[1])
    arr = np.asarray(arr).T
    return np.mean(arr, axis=1)
```

q2

In [19]:
```
nums = double['hw3_q2_doubledqn_3_LunarLander-v3_17-10-2020_16-10-58'][0
plt.plot(nums[1:], avg(double), label='double dqn')
plt.plot(nums[1:], avg(dqn), label='regular dqn')
plt.legend()
plt.title('q2 double dqn vs dqn')
plt.xlabel('iterations')
plt.ylabel('mean train return')
plt.show()
```



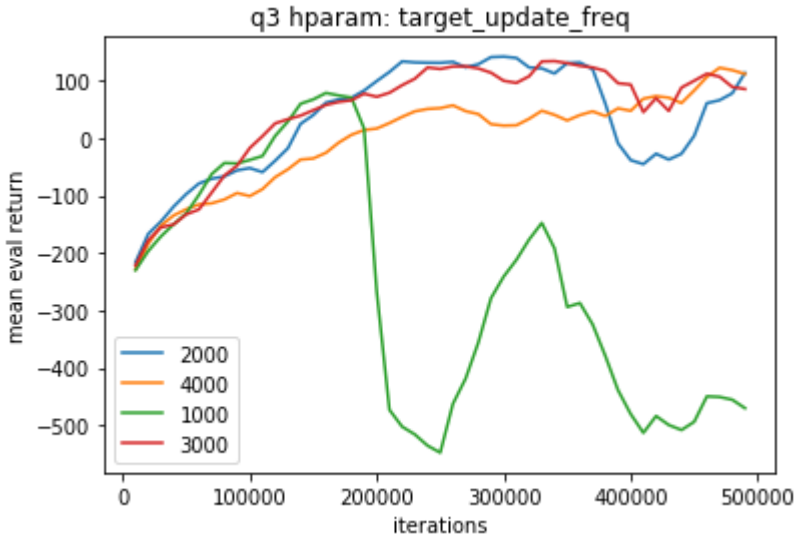
We can see that the regular dqn reaches its peak performance then falls off because of an inaccurate q_value maximization. The double dqn works much better in this regard and steadily increases. The only reason we see the dip in the graph above is because seed 3 has a very large dip that brings down the average

In [45]:
```
rv = for_section('hw3_q3')
q1 = for_section('hw3_q1')

target_update_freq ={
    'hparam1':1000,
    'hparam2':2000,
    'hparam3':4000,
    'default':3000
}
```

```
hw3_q3_hparam2_LunarLander-v3_18-10-2020_18-41-55
hw3_q3_hparam3_LunarLander-v3_18-10-2020_18-42-56
hw3_q3_hparam1_LunarLander-v3_18-10-2020_18-41-34
hw3_q1_MsPacman-v0_17-10-2020_01-04-13
hw3_q1_LunarLander-v3_18-10-2020_18-42-38
```

q3

In [46]:
```
plt.title('q3 hparam: target_update_freq')
plt.xlabel('iterations')
plt.ylabel('mean eval return')
for key, val in rv.items():
    plt.plot (val[0][1:],val[1], label = target_update_freq[key[7:14]])
p = q1['hw3_q1_LunarLander-v3_18-10-2020_18-42-38']
plt.plot (p[0][1:],p[1], label = 3000)
plt.legend()
plt.show()
```



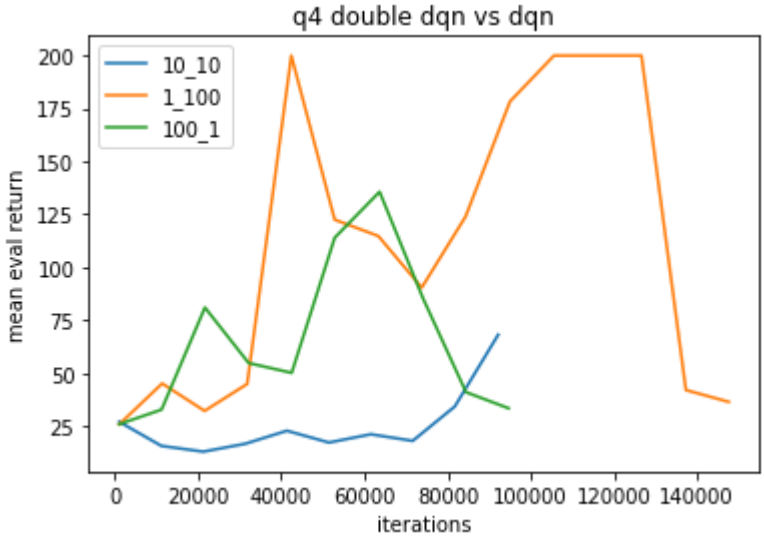
I experimented the the hyperparameter target_update_freq to see how changing how much the target was moving would effect the training process. It seemed 100 was much too low and caused the target to not be reached before it changed, which made for a target that never hit. The higher the update frequency, the slower the eval value converged because there was more time between iterations before finding a new target.

In [56]:
```
rv = for_section('hw3_ q4')
```

```
hw3_ q4_10_10_CartPole-v0_18-10-2020_17-09-03
hw3_ q4_1_100_CartPole-v0_18-10-2020_20-05-27
hw3_ q4_100_1_CartPole-v0_18-10-2020_17-08-31
```

q4

In [57]:
```
plt.title('q4 double dqn vs dqn')
plt.xlabel('iterations')
plt.ylabel('mean eval return')
for key, val in rv.items():
    plt.plot (val[0],val[2], label = key[8:13])

plt.legend()
plt.show()
```



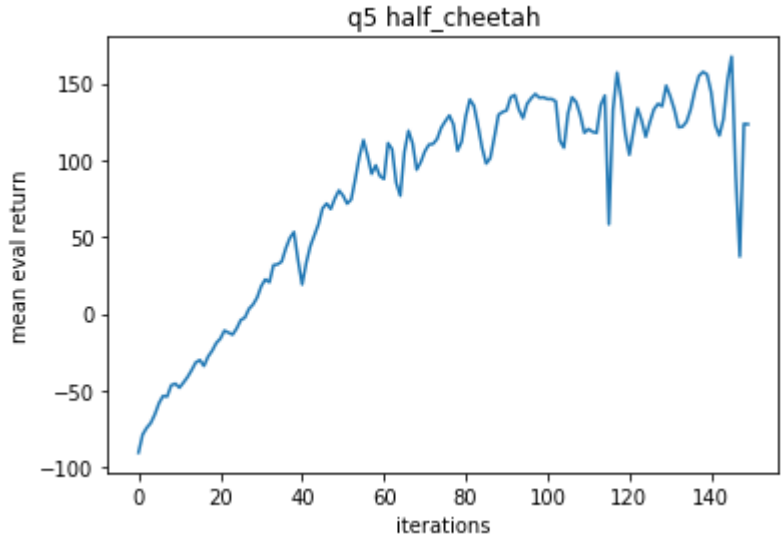
-ntu 1 -ngsptu 100 performed the best. I had to let the 1_100 run a little longer so that I could get that 200 mean eval return

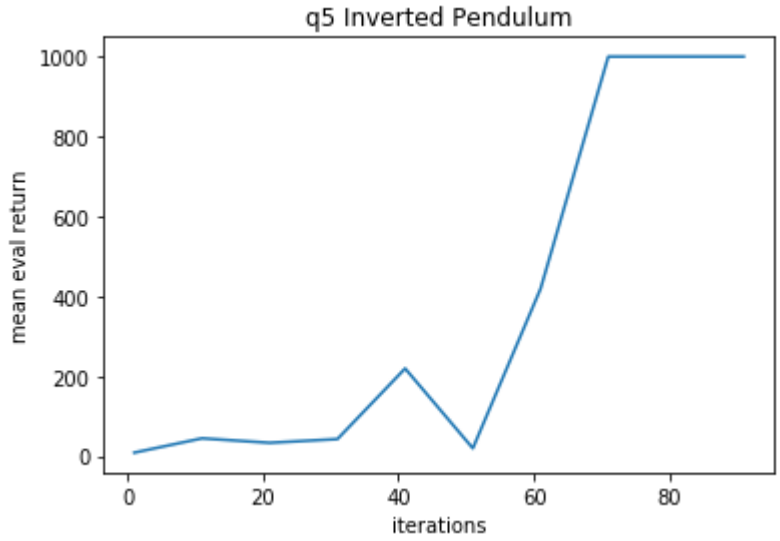
In [30]:
```
rv = for_section('hw3_ q5')
```

```
hw3_ q5_1_100_HalfCheetah-v2_18-10-2020_17-19-20
hw3_ q5_1_100_InvertedPendulum-v2_18-10-2020_17-19-35
```

q5

In [35]:
```
plt.title('q5 half_cheetah')
plt.xlabel('iterations')
plt.ylabel('mean eval return')
plt.plot (rv['hw3_ q5_1_100_HalfCheetah-v2_18-10-2020_17-19-20'][1])
plt.show()

plt.title('q5 Inverted Pendulum')
plt.xlabel('iterations')
plt.ylabel('mean eval return')
plt.plot (np.arange(1,101,10),rv['hw3_ q5_1_100_InvertedPendulum-v2_18-1
plt.show()

# plt.legend()
plt.show()
```





In [ ]: