Lab 4: Blackjack (Value - Iteration)

In [2]:
```
import gym
import pandas as pd
from collections import defaultdict
env=gym.make('Blackjack-v1',render_mode='human')
```

In [3]:
```
print(env.observation_space)
```

```
Tuple(Discrete(32), Discrete(11), Discrete(2))
```

In [2]:
```
def policy(state):
    if(state[0]>19):
        return 0
    else:
        return 1
```

In [9]:
```
state=env.reset()
print(state)
```

```
((20, 5, True), {})
```

In [10]:
```
print(policy(state))
```

```
0
```

In [3]:
```
n1=100
def generate_episode(policy):
    episode=[]
    tempstate=env.reset()
    state=tempstate[0]
    for i in range(n1):
        action=policy(state)
        next_state,reward,done,info,x=env.step(action)
        episode.append((state,action,reward))
        if done:
            break
        state=next_state
    return episode
```

In [4]:
```
print(generate_episode(policy))
```

```
[((19, 3, False), 1, -1.0)]
```

In [5]:
```
total_return=defaultdict(float)
num_of_time_state_visited=defaultdict(int)
```

In [6]:
```
total_iterations=500
for i in range(total_iterations):
    episode=generate_episode(policy)
    states,actions,rewards=zip(*episode)
    for j,state in enumerate(states):
        R=(sum(rewards[j:]))
        total_return[state]=total_return[state]+R
        num_of_time_state_visited[state]=num_of_time_state_visited[state]+1
```

In [7]:
```
total_return=pd.DataFrame(total_return.items(),columns=['state','total_return'])
num_of_time_state_visited=pd.DataFrame(num_of_time_state_visited.items(),columns=['st
df=pd.merge(total_return,num_of_time_state_visited,on="state")
df.head()
```

Out[7]:

	state	total_return	N
0	(18, 6, False)	-2.0	6
1	(20, 2, True)	0.0	2
2	(11, 10, False)	2.0	15
3	(17, 10, False)	-16.0	24
4	(11, 1, False)	-3.0	5

In [8]:
```
df['value']=df['total_return']/df['N']
df.head(10)
```

Out[8]:

	state	total_return	N	value
0	(18, 6, False)	-2.0	6	-0.333333
1	(20, 2, True)	0.0	2	0.000000
2	(11, 10, False)	2.0	15	0.133333
3	(17, 10, False)	-16.0	24	-0.666667
4	(11, 1, False)	-3.0	5	-0.600000
5	(21, 1, False)	0.0	4	0.000000
6	(19, 6, False)	-1.0	4	-0.250000
7	(20, 6, False)	7.0	8	0.875000
8	(20, 7, False)	14.0	16	0.875000
9	(10, 7, False)	-1.0	4	-0.250000

In [9]:
```
df.to_csv('RLtrained.csv',index=False)
```

In [14]:
```
df[df['state']==(10,7,False)]['value'].values
```

Out[14]:
```
array([-0.25])
```