

SE4050

Deep Learning

**4th Year, 1st Semester**

Lab 08

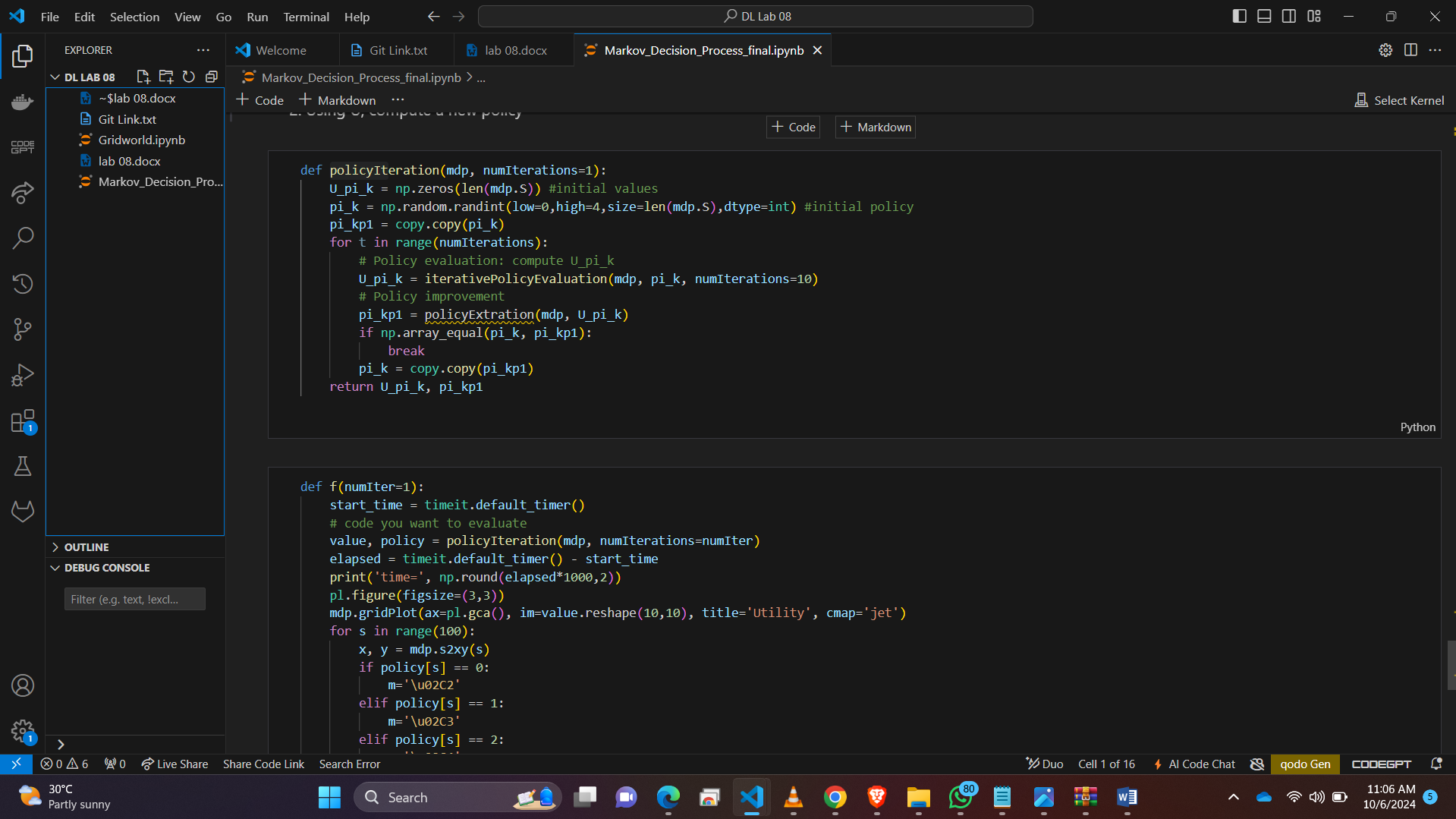
Registration Number: IT21347962

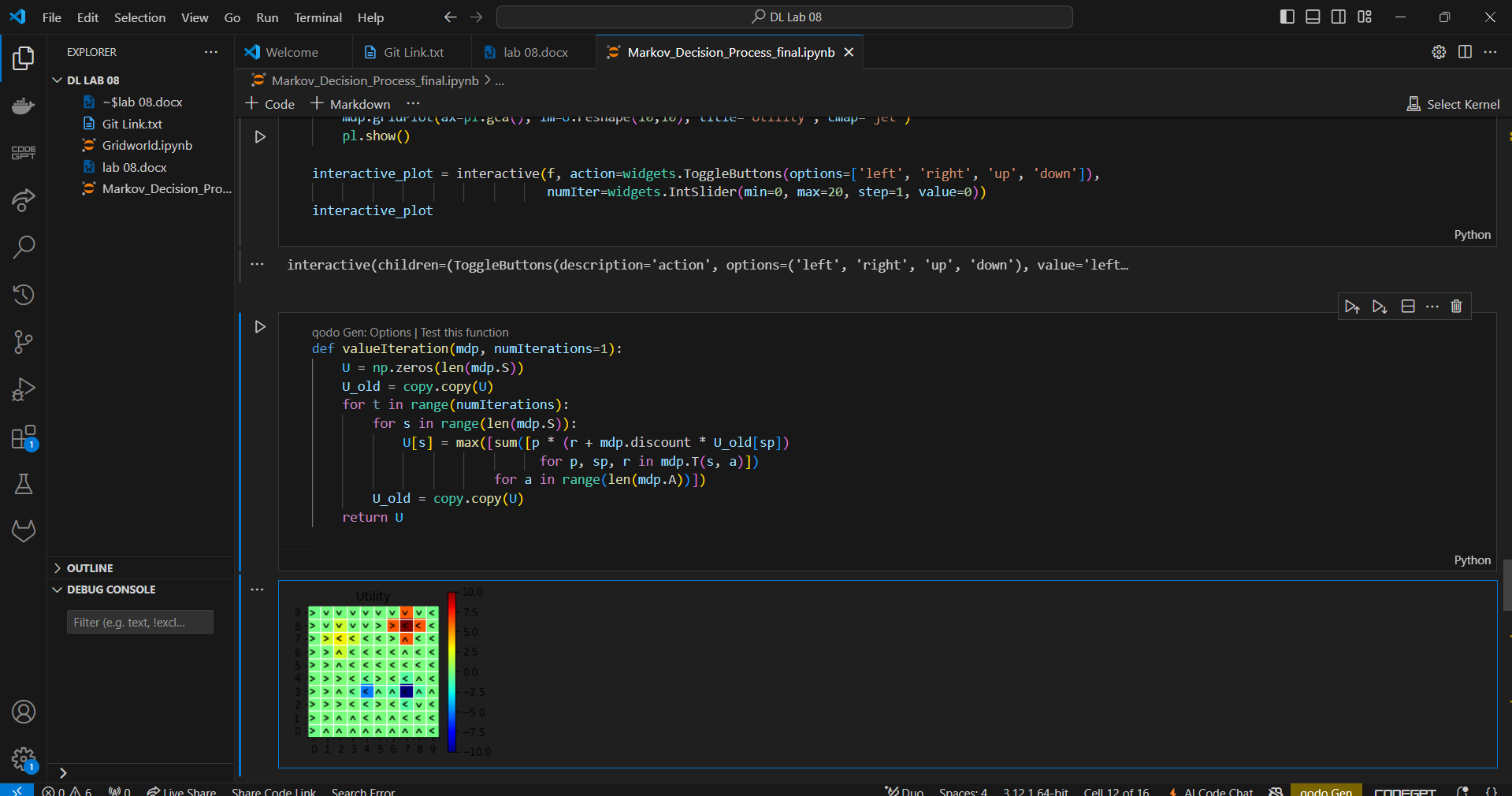
Name: Siribaddana K.

10/03/2024

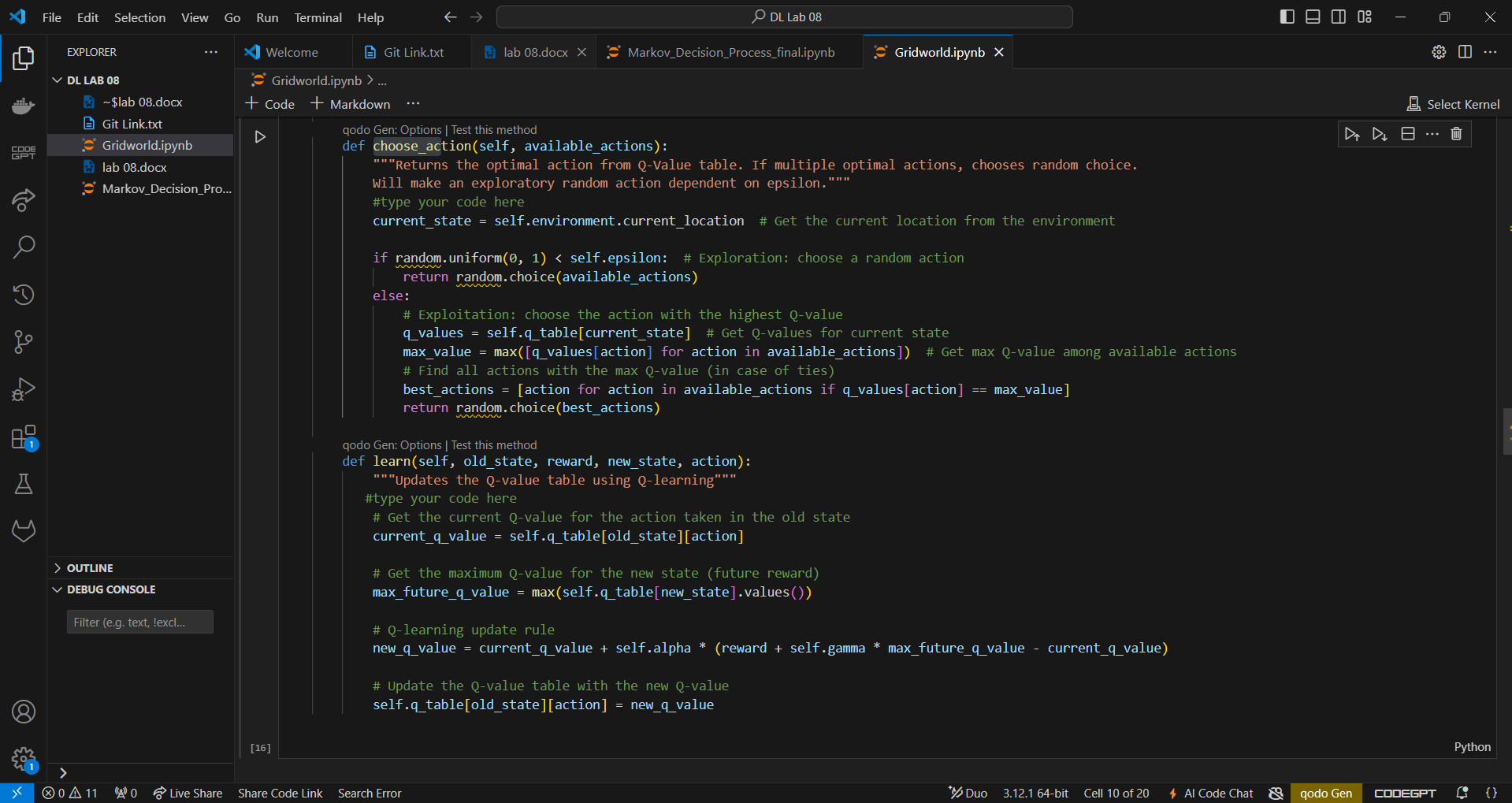
**Question 1**

**Task 3**Updated part in the Markov Decision Process file

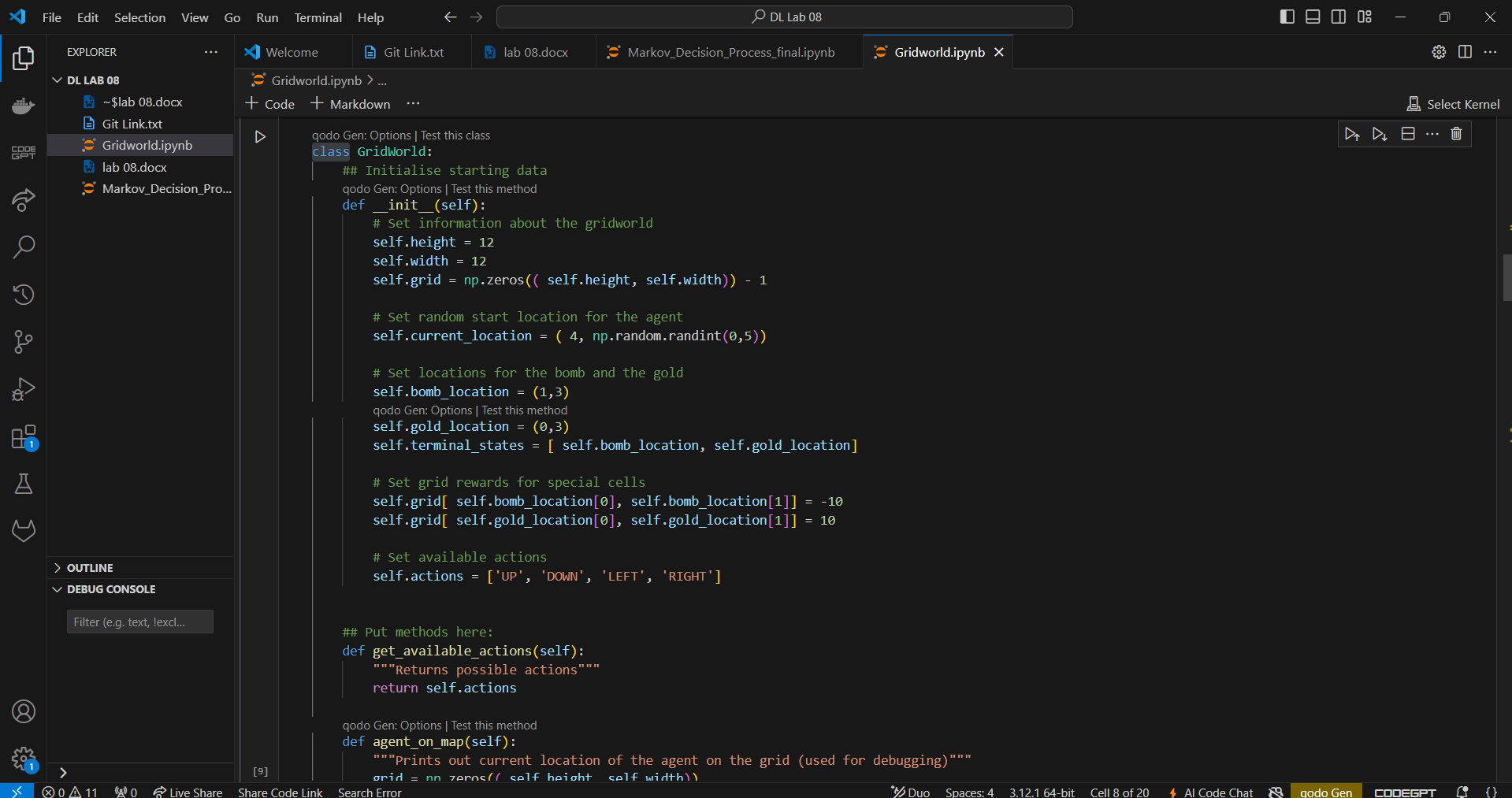




Updpated part in the GridWorld (QLearning) notebook



Updated part in the GridWorld (QLearning) notebook Grid Size



**Question 2**

Value Iteration was implemented to compute the optimal policy by updating the utility values for each state. The following code was added:

def valueIteration(mdp, numIterations=1):  
 U = np.zeros(len(mdp.S))  
 U\_old = copy.copy(U)  
 for t in range(numIterations):  
 for s in range(len(mdp.S)):  
 U[s] = max([sum([p \* (r + mdp.discount \* U\_old[sp])   
 for p, sp, r in mdp.T(s, a)])  
 for a in range(len(mdp.A))])  
 U\_old = copy.copy(U)  
 return U

**Question 3**

The policy extraction method was implemented to derive the policy from the utility values. The following code was added:

def policyExtration(mdp, U):  
 policy = np.zeros(len(mdp.S))  
 for s in range(len(mdp.S)):  
 policy[s] = np.argmax([sum([p \* (r + mdp.discount \* U[sp])   
 for p, sp, r in mdp.T(s, a)])   
 for a in range(len(mdp.A))])  
 return policy

**Question 4**

Policy Iteration was implemented to iteratively evaluate and improve the policy based on the utility values. The following code was added:

def policyIteration(mdp, numIterations=1):  
 U\_pi\_k = np.zeros(len(mdp.S)) #initial values  
 pi\_k = np.random.randint(low=0,high=4,size=len(mdp.S),dtype=int) #initial policy  
 pi\_kp1 = copy.copy(pi\_k)  
 for t in range(numIterations):  
 U\_pi\_k = iterativePolicyEvaluation(mdp, pi\_k, numIterations=10)  
 pi\_kp1 = policyExtration(mdp, U\_pi\_k)  
 if np.array\_equal(pi\_k, pi\_kp1):  
 break  
 pi\_k = copy.copy(pi\_kp1)  
 return U\_pi\_k, pi\_kp1