# Question 3

*Note: documentations and comments are omitted in this word document as including them takes up too much space, but are included in the notebooks provided*

## Part a)

What policy iteration is

## Part b) Policy Iteration algorithm

Text

Description automatically generated with medium confidence

Figure Policy iteration pseudo code, also included in question3.ipynb

**Implementation**



The policy evaluation corresponds to the first and second block in the pseudo code, and the implementation follows exactly line-by-line from the pseudo code. Note that to iterate through every state, we have to iterate through the entire grid which is row by columns by headings large. Where the first channel is the row, the second channel is the column, and the last channel is the heading (directions). Hence, we use a nested for-loop to get access to all the states. The policy evaluation function will return a value grid that approximates the state value of every state given the current policy.

The implementation of the helper function get\_state\_value is as follows:



Where the environment object’s internal functions are used to calculate the next state reward and whether or not a terminal state has reached. Note that there’s a small bug in the file simple\_grid\_env.py, this is described in detail in question3.ipynb. The presented implementation assumes that the mistake is fixed.

The policy improvement step corresponds to the third block in the pseudo code, and is implemented as follows: 

Similarly, a nested loop is used to access each state, and the code follows line-by-line from the pseudo code. This function returns an improved policy and a Boolean that describes whether or not the policy grid has converted to a stable state.

Lastly, a wrapper iteration function to include both the policy evaluation function and the policy improvement function is as follows:

