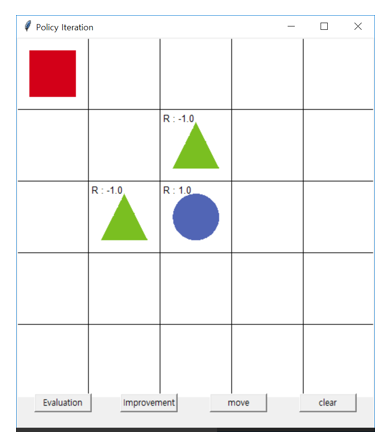
HW3

3. Visit and check the following URL. https://github.com/rlcode/reinforcement-learning (for short, https://goo.gl/qe44V5)

1

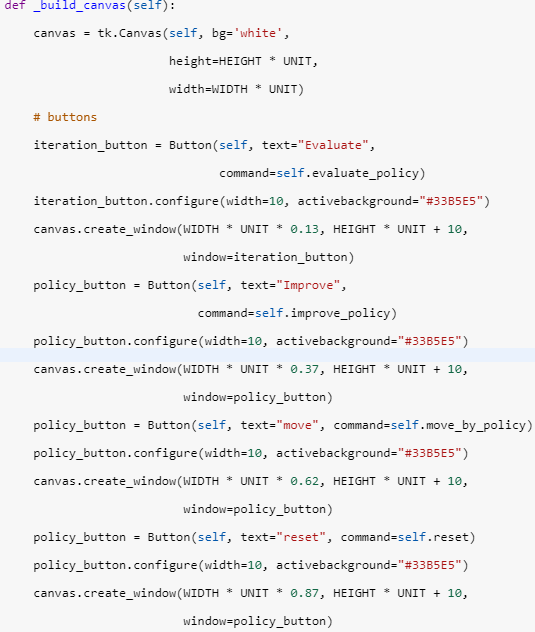
(a) Run the policy iteration code in the following URL. https://github.com/rlcode/reinforcement-learning/tree/master/1-grid-world/1-policy-iteration (for short, https://goo.gl/74r2Z8)

i. Analyze the programs (environment.py and policy\_iteration.py) and write/submit a report. Also, capture two screenshots of the program execution and include them in the report.



In the game, the red rectangle will move step by step. If it comes cross triangle, then reward will -1, if it gets to the circle, then reward will +1.

**environment.py**



This function gives the definition of canvas.



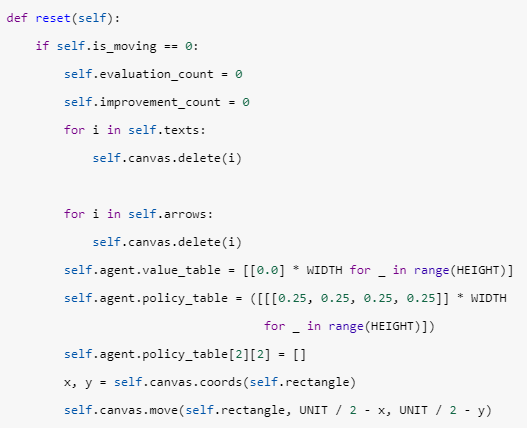
The apartment of creating grids draw grids on canvas by using create\_line function.Every line is drawed from (col,0) to (col, HEIGHT\*UNIT) and from (0, row) to (HEIGHT\*UNIT,row).

The second apartment will add the image of triangle, circle on vanvas.

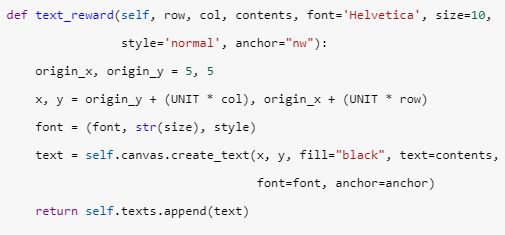
The function of pack(), help to package data.



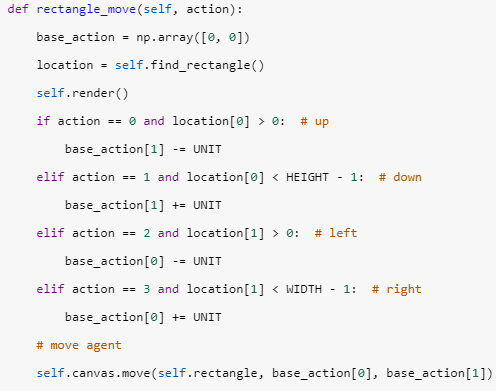
This part will load the images of arrows (up, right, down and left) and images of rectangle, triangle and circle from the saved location.



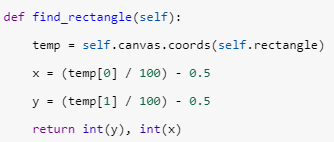
Reset function will initializes the game .



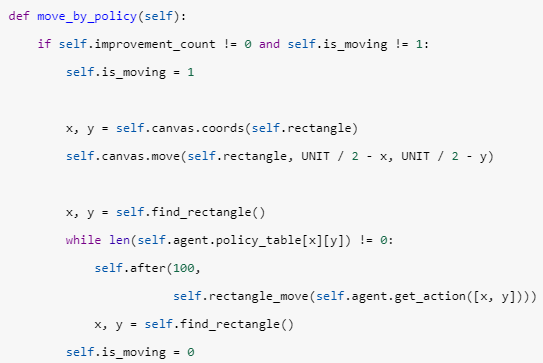
The function gives contents to texts.



The value of action (0,1,2,3) give the sense of “up, down, left, right”.For example” action == 0 and location[0] > 0” means if action is up and current location is not on the boundary.” Then “base\_action[1] -= UNIT” means that the rectangle moves up, the step is one UNIT,UNIT is equal to pixels.

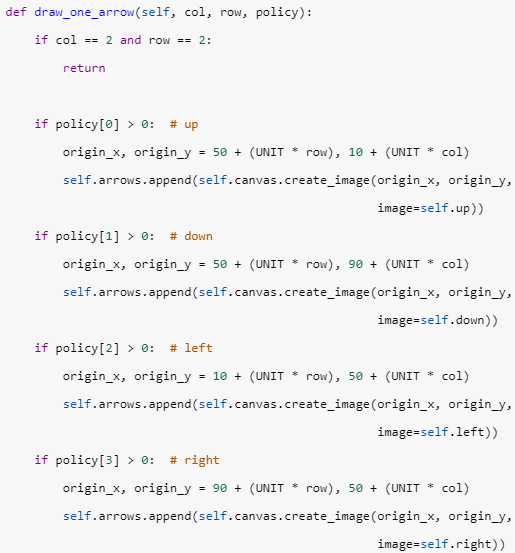


This function will give the location on the rectangle in int.

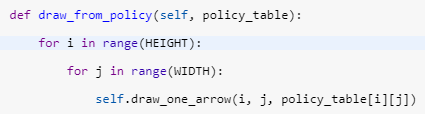


If ”improvement\_count=0” means “shouldn’t move” , ”improvement\_count=1” means “should move”.” is\_moving = 1” means the rectangle is moving, either “isn’s moving”. Coords() function gets the coordinate of rectangle. Move() function gives the coordinate (UNIT / 2 - x, UNIT / 2 - y) the rectangle should move to. Then the rectangle will move by following agent.get\_action.

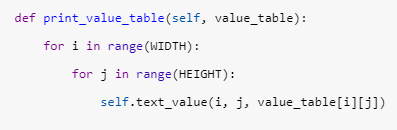
When the moving is over, giving 0 to is\_moving.



Using create\_image() function to draw arrows(up, down, left, right).



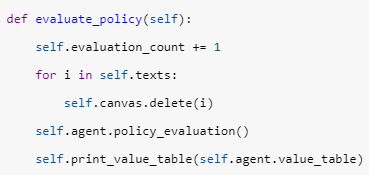
This function requires an arrow to be drawn in a specific direction for each step.



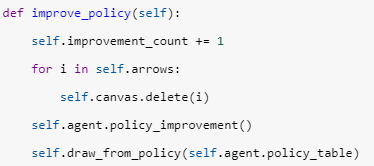
This function will put the value of each step in value\_table.

![C:\Users\IAI User\AppData\Roaming\Tencent\Users\570117572\QQ\WinTemp\RichOle\}8$$VGV6W~WI%[Z](J3@WOO.png](data:image/png;base64,)

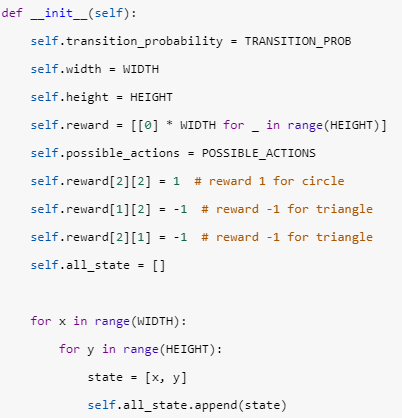
Drawing a rectangle in certain location after 0.1 second.



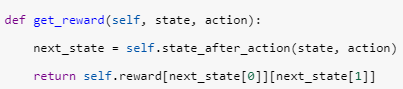
For this function, in first step, evaluation\_count will plus 1,and each text will be deleted, and then the program of agent.policy\_evaluation will be run, in the end, the result of value for agent will be printed.



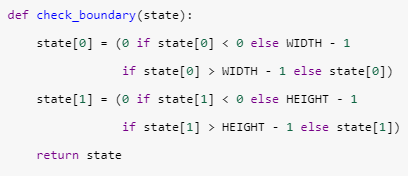
First two steps are similar to the previous function. After that , agent.policy\_improvement() will be run, and the arrow for next action will be given.



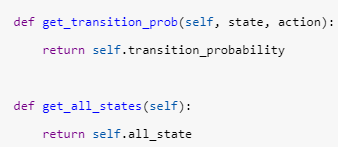
Class Env has some definition for environment. In init() definition, all state will be put in all\_state[].



By the current state and action, we will know the next state. And we can also get the reward by recognizing if its circle , triangle or nothing. If its circle ,the reward will be 1. If its triangle reward will be -1.



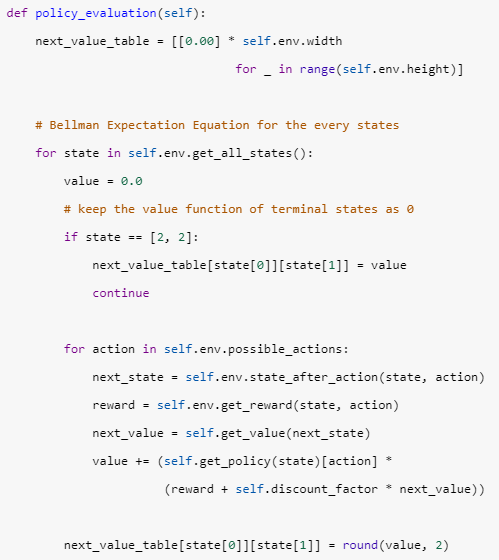
This function will help to find whether current step is on the boundary. If it has reached to the boundary, state will be 0.



Function will return transition\_probability 1 and the vector all\_state.

**policy\_iteration.py**

The program has two import parts: policy evaluation and policy improvement.



C:\Users\IAI User\AppData\Roaming\Tencent\Users\570117572\QQ\WinTemp\RichOle\}W)J0VOQ85U59)04MH}N]DM.png

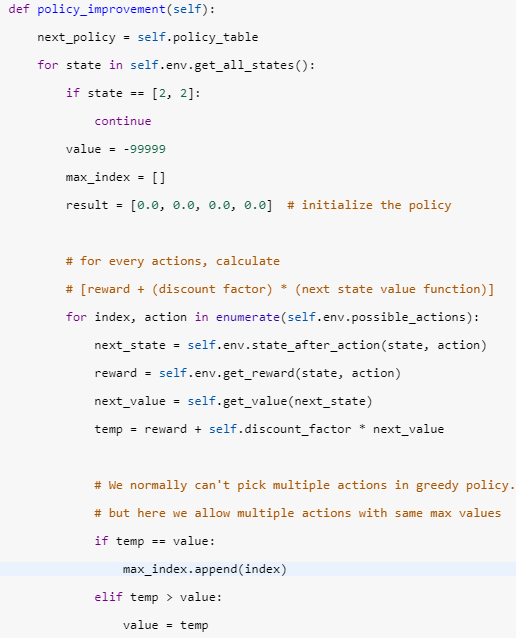
Policy\_evaluation: To evaluate the policy is to check the reward.Firstly, we will clear the next\_value\_table and variable value. Before rectangle starting to go, it will check the current state is [2,2]( where the circle is), if the rectangle has reached the circle, then next\_value\_table will be 0, then the final result will be reached.

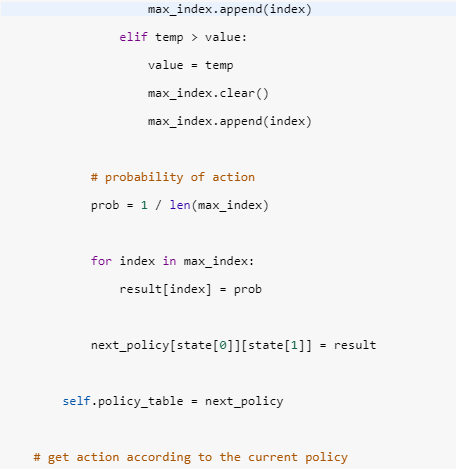
If the rectangle hasn’t reached the circle, we will calculate the formulation:

C:\Users\IAI User\AppData\Roaming\Tencent\Users\570117572\QQ\WinTemp\RichOle\]5M}C)WY[3H5FH}F%~K{_ZW.png

C:\Users\IAI User\AppData\Roaming\Tencent\Users\570117572\QQ\WinTemp\RichOle\{(_0~(NEP1[B0D$_J%~ZRXQ.pngis the policy, C:\Users\IAI User\AppData\Roaming\Tencent\Users\570117572\QQ\WinTemp\RichOle\PC_9DHG[WU$5SE7KIP`F_X3.pngis reward, C:\Users\IAI User\AppData\Roaming\Tencent\Users\570117572\QQ\WinTemp\RichOle\N1R`UT9`J_B8240G3X1LE~L.png is discount\_factor, C:\Users\IAI User\AppData\Roaming\Tencent\Users\570117572\QQ\WinTemp\RichOle\_8LZ1I%1~Y55~(GM[1~X914.pngis next\_value. In the formulation, A will be 4 actions.

After that, the value will be put in current value\_table[state[0]] [state[1]].





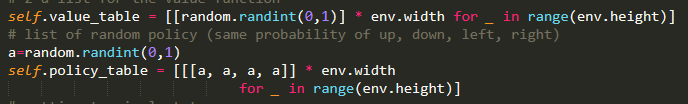
Policy improvement:

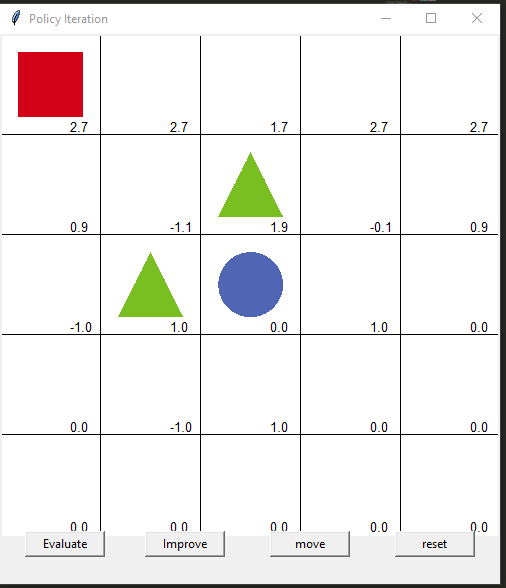
1. Initialize value=-9999.
2. Calculate temp=reware+0.9\*next\_value
3. If “temp” equal to “value”, the current index will be put in max\_index.

If “temp” > “value”, then max\_index will be epmptied, current index will be put in it.

1. For 4 actions, each action will have one value, if len( max\_index)=0, the probability of choosing is 0,if len( max\_index)=1, the probability of choosing is 1, len( max\_index)=2, the probability of choosing is 1/2, len( max\_index)=3, the probability of choosing is 1/3, len( max\_index)=4, the probability of choosing is ¼.

ii. In the program, the initial values of value function are zero, and the initial policy follows uniform distribution. Change initial values to random values, and change the initial policy to be normalized random values. (i.e. random probabilities and their sum should be one) Observe the training process and include the analysis in the report. Also, capture two screenshots of this modiﬁed program’s execution and include them in the report. iii. With the original setting for value function and policy, consider the following experiment. Note that there are already triangles at (3,2) and (2,3). Now add one more triangle with reward=-1 at (3,4). Observe the training process and include the analysis in the report. Also, capture two screenshots of this modiﬁed program’s execution and include them in the report.

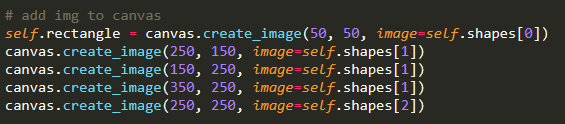


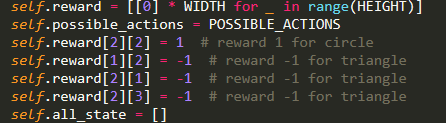


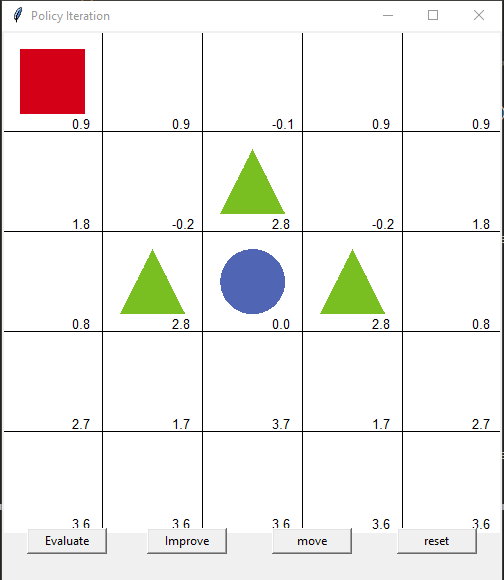
Analyze:

Change initial values to random values, and change the initial policy to be normalized random values. The rout rectangle chooses may be change.

iii. With the original setting for value function and policy, consider the following experiment. Note that there are already triangles at (3,2) and (2,3). Now add one more triangle with reward=-1 at (3,4). Observe the training process and include the analysis in the report. Also, capture two screenshots of this modiﬁed program’s execution and include them in the report.



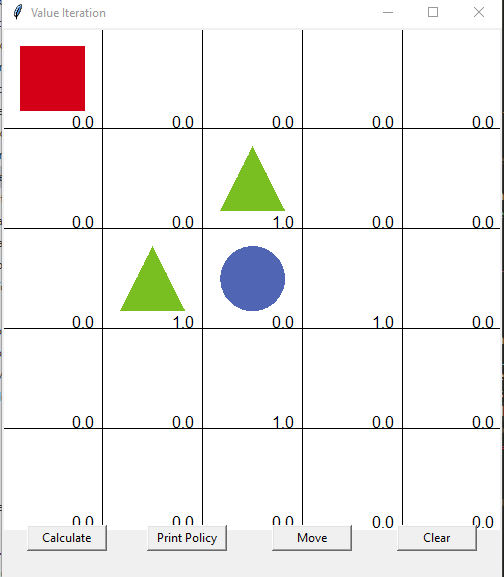
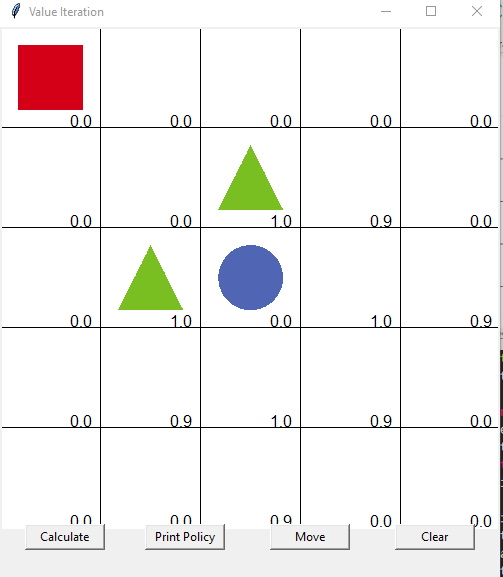


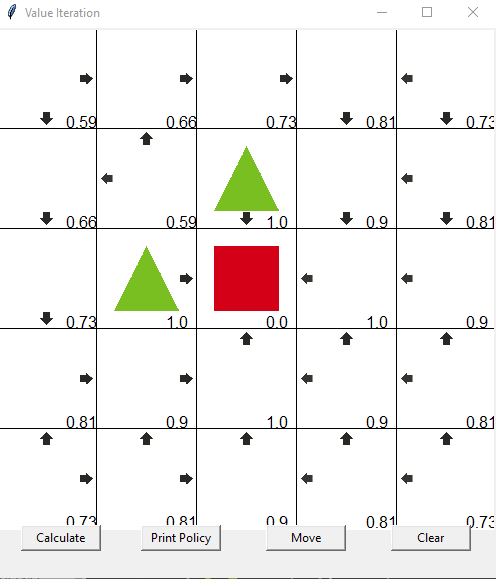


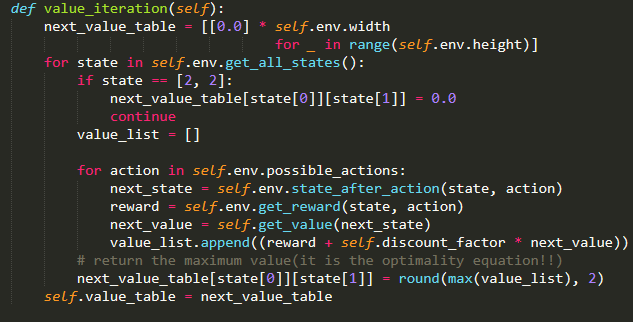
When added another triangle and the process is trained enough, the rectangle will follow the route： (1,1)-(2,1)-(3,1)-(4,1)-(4,2)-(4,3)-(3,3). Because this route will be more near.

(b) Run the value iteration code in the following URL. https://github.com/rlcode/reinforcement-learning/tree/master/1-grid-world/2-value-iteration (for short, https://goo.gl/8Ff8gZ)

i. Analyze the programs (environment.py and value\_iteration.py) and write/submit a report. Also, capture two screenshots of the program execution and include them in the report.



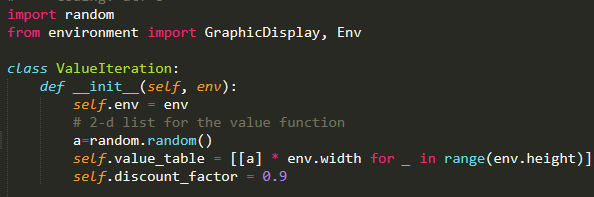
The difference between a and b is this part. In this part value is update by formulation:

reward + self.discount\_factor \* next\_value.

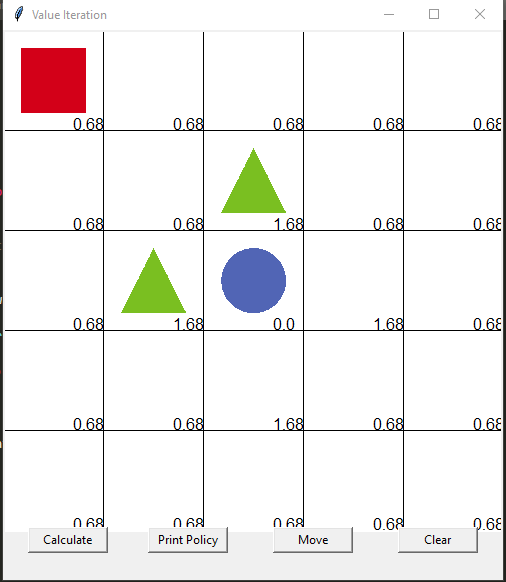
Four pictures above are first step, second step, third step and last step. From the formulation and pictures we can see that the values of grids around circle are 1, values of grids around 1 are 0.9(discount) and values of grids around 0.9 are 0.81, and so on.

How to move: The rectangle will chose one grid around it with biger value, and so on.

ii. In the program, the initial values of value function are zero. Change initial values to random values. Observe the training process and include the results in the report. Also, capture two screenshots of this modiﬁed program’s execution and include them in the report.



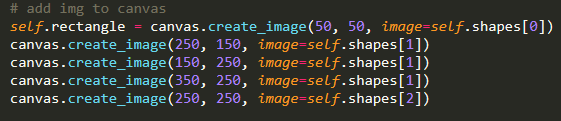
The first calculation:

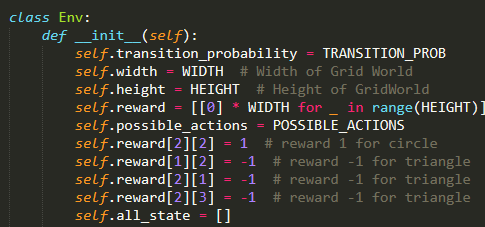


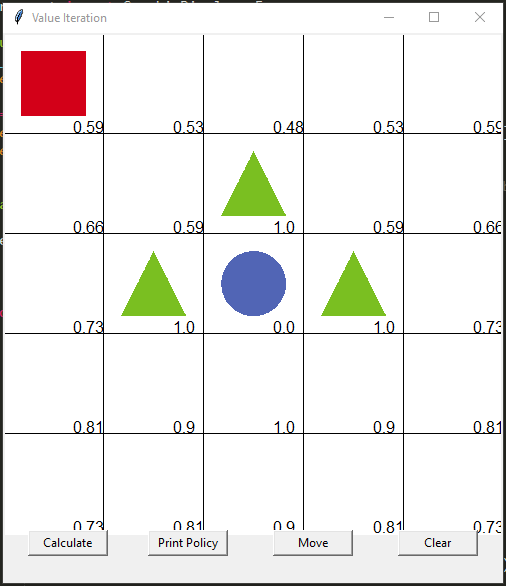
Though value initialization has been changed, but the path rectangle follows will not change.

The reason is that the values of grids around circle will be changed to 1

iii. With the original setting for value function, consider the following experiment. Note that there are already triangles at (3,2) and (2,3). Now add one more triangle with reward=-1 at (3,4). Observe the training process and include the analysis in the report. Also, capture two screenshots of this modiﬁed program’s execution and include them in the report.







When added another triangle and the process is trained enough, the rectangle will follow the route： (1,1)-(2,1)-(3,1)-(4,1)-(4,2)-(4,3)-(3,3). Because this route will be more near.