ck your progress on the go! Install App

Back

Congrats Daniel! This project has been marked as completed.

Project Rating

★★★☆☆

Teacher's Comment

"Good"

-

Was this helpful?

Community Link

Publish to Community

Edit Your Project

Last Submitted

Previous Submissions

3rd Mar 2024 Open Link

Start Project

Submit Your Project

Learn how to submit your project

Paste your project URL

Submit Project

Class Summary

This project is based on your last class PRO-C125

View Class Summary

PRO-C125: ARCHERY PLAYER 2 Completed

In Class 125, We Learned How To Create Q-Matrix To Maximize The Reward And Take Appropriate Action. In This Project, We Are Going To Help The Agent To Take Action Using Q-Matrix.

Goal of the Project:

In class 125, we learned how to create Q-matrix to maximize the reward and take appropriate action.

In this project, we are going to help the agent to take action using Q-matrix.

Story:

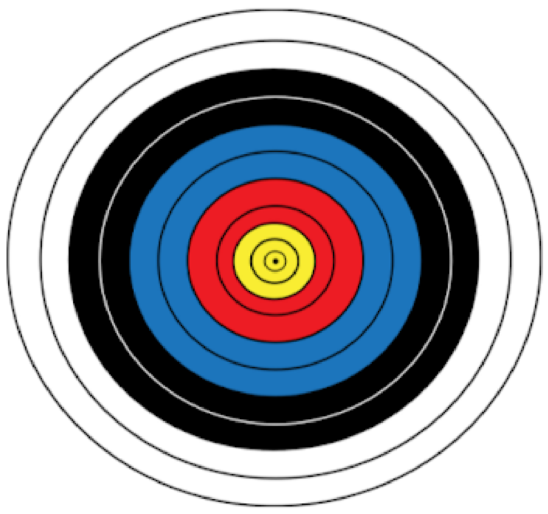
At the Chicago School of Artificial Intelligence, an archery competition is held for Robots. **Archery-Target** is a game in which the players shoot sharp-pointed arrows at a round target having 10 rings. Joseph has to train his robot to hit the Bull's eye in the very first attempt. He needs your help so that he can use the RL method to train the robot to play archery.

Can you help him to do so?

Project Template Output

RL Problem to Solve

Hit the center of the target with maximum reward



Number of **State**: ?

Number of **Actions**: ?

Expected Output

```
action,reward=take_action(rewards)

Shot at:   3
Reward for the shot :   7
```

Ask a doubt to your tea HELP

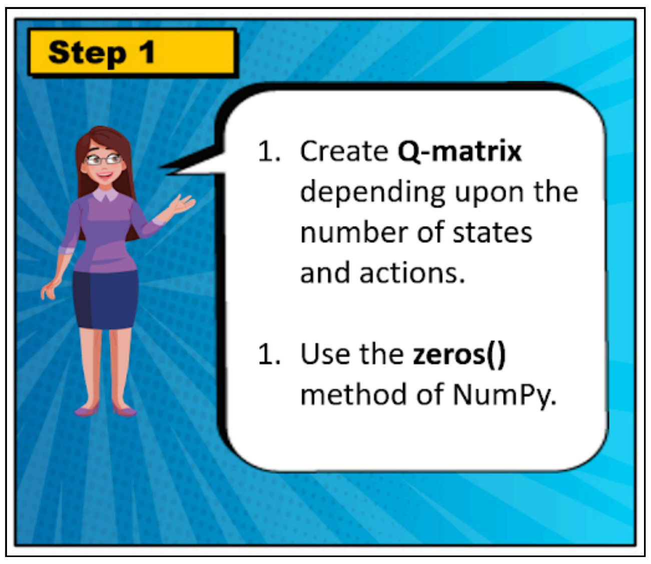
***This is just for your reference. We expect you to apply your own creativity to the project.**

Note: This project is the continuation of **Project 123**. You can continue with the same project code. Previously we created the **Reward matrix** and defined the **shoot()** function to choose random actions.

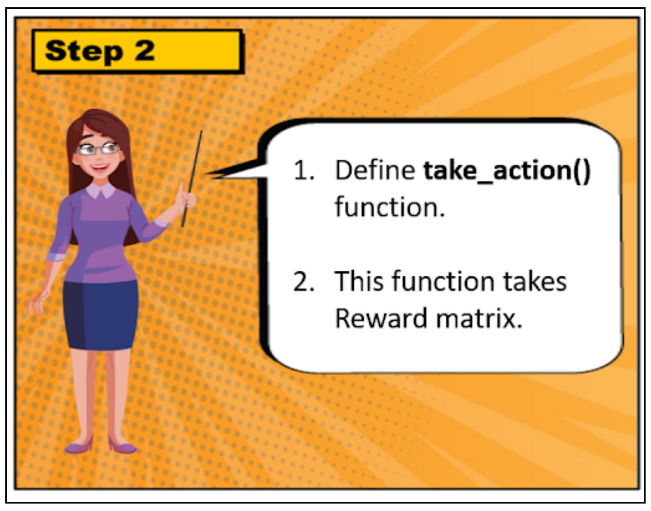
Getting Started:

1. Open the boilerplate link or continue with project 123.

Specific Tasks to complete the Project:

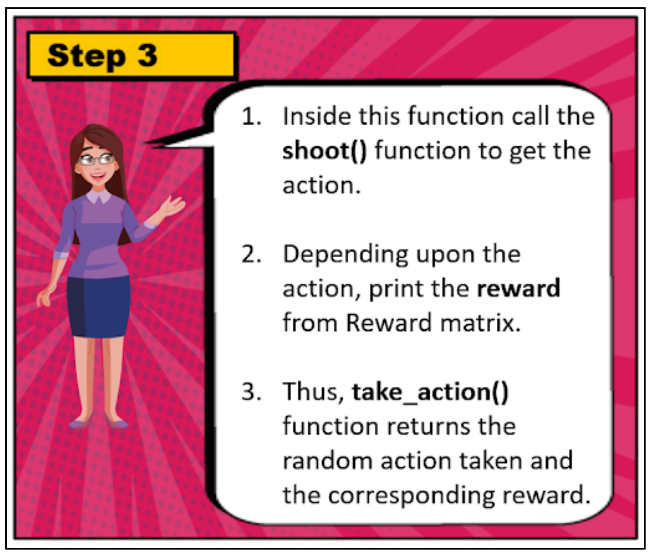

Step 1
1. Create **Q-matrix** depending upon the number of states and actions.
1. Use the **zeros()** method of NumPy.

```
#Create Q-matrix
```


Step 2
1. Define **take_action()** function.
2. This function takes Reward matrix.

```
#Define take_action

def take_action(reward_matrix):
```


Step 3
1. Inside this function call the **shoot()** function to get the action.
2. Depending upon the action, print the **reward** from Reward matrix.
3. Thus, **take_action()** function returns the random action taken and the corresponding reward.

HELP

```
#Define take_action

def take_action(reward_matrix):

     #Call the shoot() function to get the action

     #Print the action

     #Get the corresponding reward using Reward matrix

     #Print reward

     return action, reward
```

Step 4
Call the take_action() function to print the action and reward.

```
#Call take_action function
```

Submitting the Project:

1. **SAVE** all the changes made to the project.
2. Click on **"Run"** once to check if it is working.
3. Open GitHub and create a repository named **Project125**.
4. Click **Share**.

5. Click Change and choose the **'anyone with the link'** option.

Hints:

1. In step 1, Q-matrix will be a **1D array** as only one state multiple actions are present.
2. In step 3, in the **take_action()** function, use action as the index to find the corresponding reward in the Reward matrix.