Topic	**Q-MATRIX**	
Class Description	**The student will be introduced to the concept of Q-matrix and the use of the Bellman Equation to update Q-matrix.**	
Class	**PRO C125**	
Class time	**45 mins**	
Goal	● To create a Q-Matrix ● Use of the Bellman equation ● To decide action using Q-matrix	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher-Led Activity 1** **Student-Led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **05 mins**
WARM-UP SESSION - 10 mins		
Teacher Starts Slideshow **Slide 1 to 4** Refer to speaker notes and follow the instructions on each slide.		
Teacher Action	**Student Action**	

<table>
<tr><td>Hey &lt;student's name>. How are you? It’s great to see you! Are you excited to learn something new today?

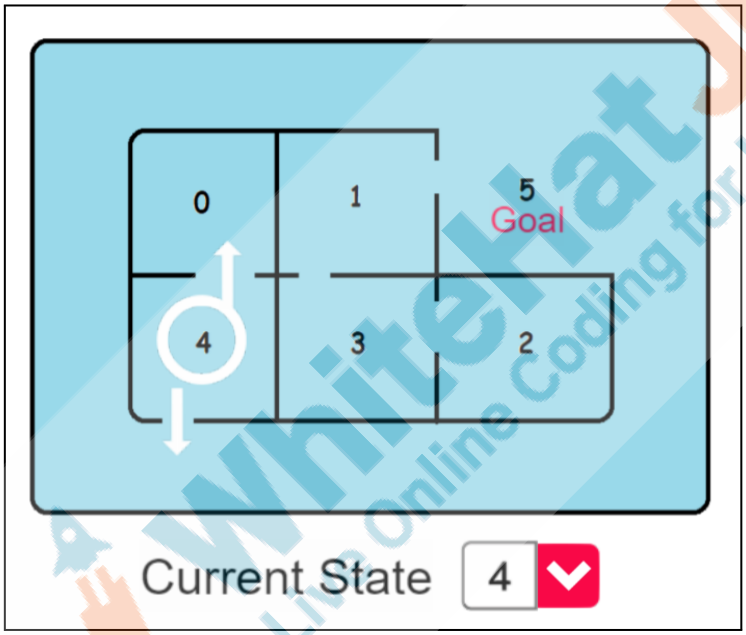
<b>Following are the WARM-UP session deliverables:</b>
● Greet the student.
● Revision of previous class activities.
● Quizzes.</td><td><b>ESR</b>: Hi, thanks!
Yes, I am excited about it!

Click on the slide show tab and present the slides</td></tr>
<tr><td colspan="2"><b>WARM-UP QUIZ</b>
Click on In-Class Quiz</td></tr>
<tr><td colspan="2"><b>Continue WARM-UP Session</b>
<b>Slide 5 to 12</b></td></tr>
<tr><td colspan="2"><b>Activity Details</b>

<b>Following are the session deliverables:</b>
● Appreciate the student.
● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in the students.</td></tr>
<tr><td colspan="2"><b>Teacher Ends Slideshow</b></td></tr>
<tr><td colspan="2"><b>TEACHER-LED ACTIVITY - 10 mins</b></td></tr>
<tr><td colspan="2"><b>Teacher Initiates Screen Share</b></td></tr>
<tr><td colspan="2"><b><u>ACTIVITY</u></b>

● <b>Introduction to Q-Matrix</b>
● <b>Introduction to Bellman equation</b></td></tr>
<tr><td><b>Teacher Action</b></td><td><b>Student Action</b></td></tr>
<tr><td>In the last class we had seen that if the current state is</td><td></td></tr>
</table>

selected as 4, its possible actions were state 0 or 5. The next state was chosen randomly among any of these states. Ideally, Which state should be chosen according to you? ***Note:*** *Open Teacher Activity 1 to show the available actions for state 4 to the student.*	**ESR:** It should choose 5 as it is the Goal.



Now, our aim is to make the agent learn from the action taken and decide the action accordingly. So, to find the way out, today we are going to learn about the Q-learning method. **Q-learning:** **Q-learning** is a reinforcement learning algorithm. Given the current state, it **helps to find the next best action** to be taken by the agent. **Q** stands for **Quality** in Q-learning. Quality represents how useful action is in gaining a reward.	

Best Action:

Now to find the best action, Q-learning algorithm repeated over multiple times.

While running multiple times, agents will take multiple actions, then how do we find the best action among them?

This is done by keeping track of actions in a table which is called **Q-Table** or **Q-Matrix**.

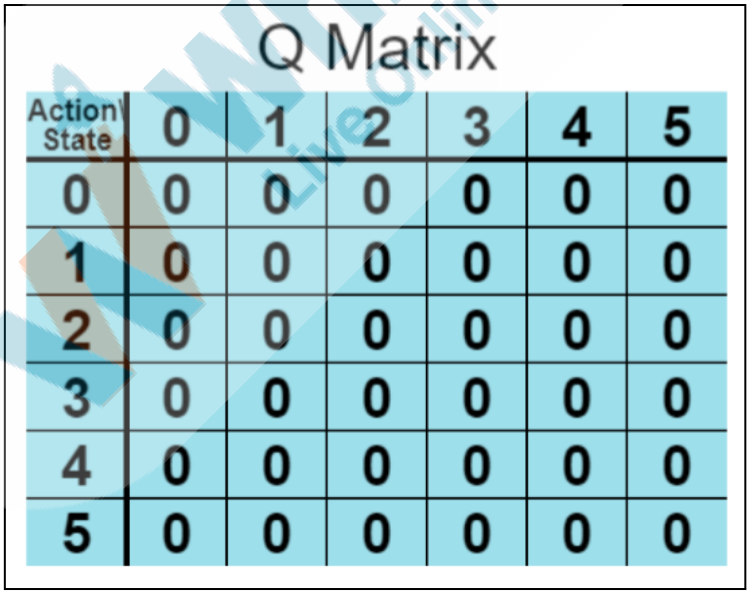
It is also in the form of **states as rows** and **actions as columns**.

We have 6 states (0-5) and 6 actions. Thus the matrix will be a 6x6 matrix.

Initially, all the elements of the Q-matrix are zeroes.

Note: *Open Teacher Activity 2 to show the Q-matrix to the student.*

ESR: Varied.



Q Matrix

Action\State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

<u>The Q-Matrix is updated according to the rewards given for each action.</u>

The Q-Matrix is updated for different rewards, and once the Q-Matrix completes the action with maximum reward is chosen.

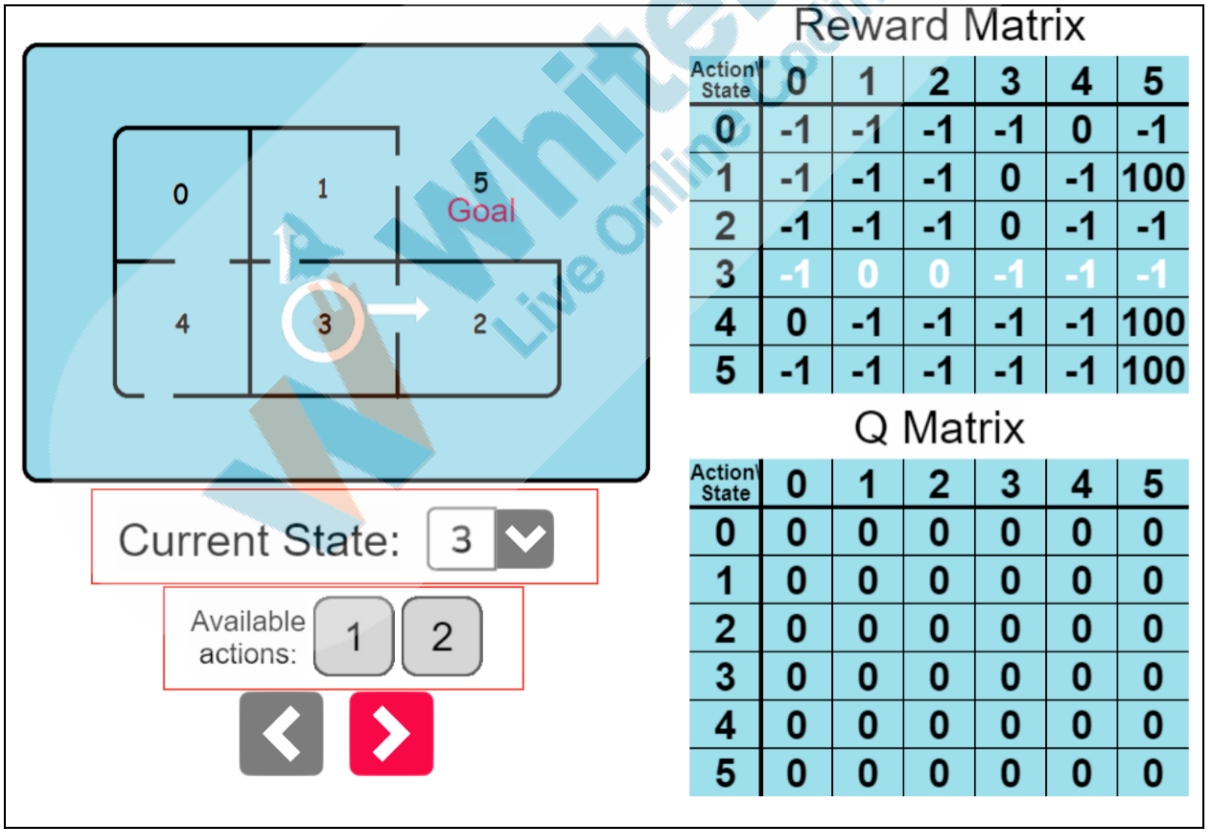
Let’s first learn this by example using the following activity link.

Note: *Open [Teacher Activity 2]() to click on current state and then click on available actions.*

We can choose any state as our initial state.

Let’s choose 3. As you can see next the available actions for the agent is to either move to room 1 or room 2.

Initial State → Room 3
Available Actions → Room 1, Room 2



Reward Matrix

Action State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	-1	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	-1	100

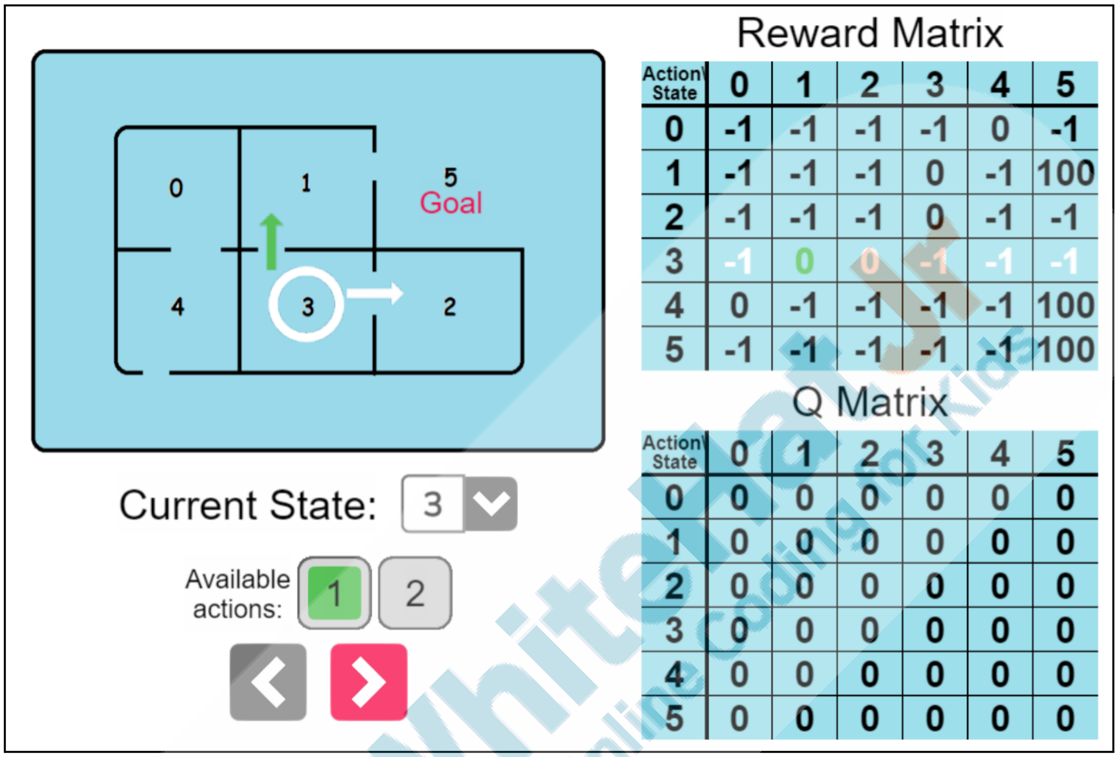
Q Matrix

Action State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Current State: 3

Available actions: 1 2

Among these actions, I am choosing 1, so for 1, it is highlighting the rewards in the Reward Matrix, which is 0.



Current State: 3

Reward Matrix

Action\State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	-1	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	-1	100

Q Matrix

Action\State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

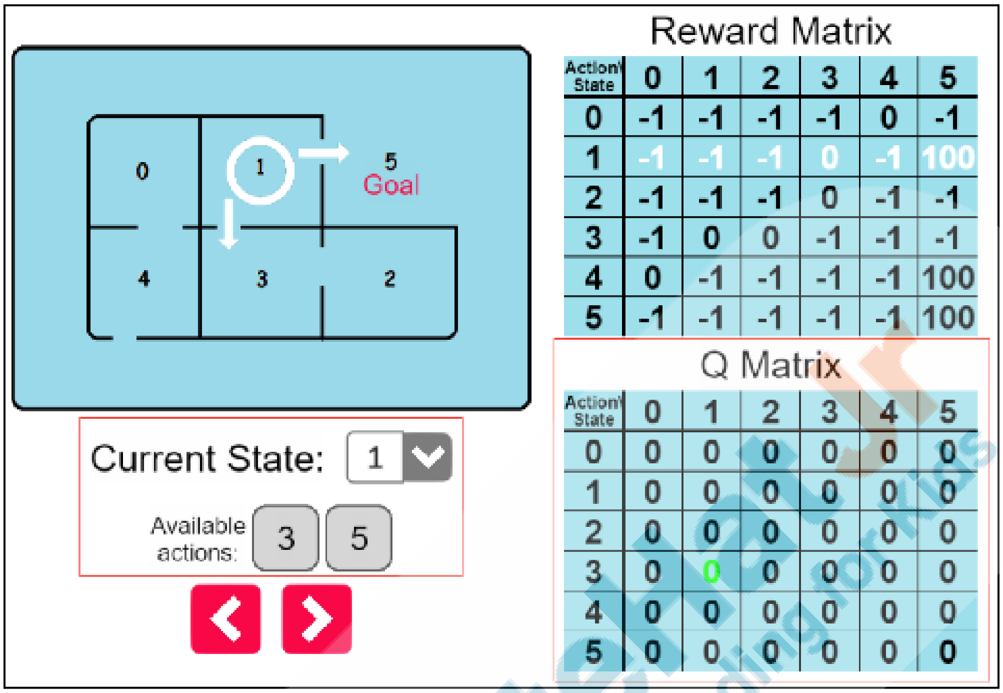
When the arrow is clicked, it means that the **action is chosen and this becomes the new state** of the agent, then the available actions are shown according to the new state.

Chosen Action → Becomes New State

Also, the **Q-Matrix is updated with the reward value 0 for action** 1.

Now let's continue for new state 1.
As you can see next the available actions for the agent is to either move to room 3 or room 5.

New State → Room 1
Available Actions → Room 3, Room 5



Current State: 1

Available actions: 3 5

Reward Matrix

Action\ State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	-1	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	-1	100

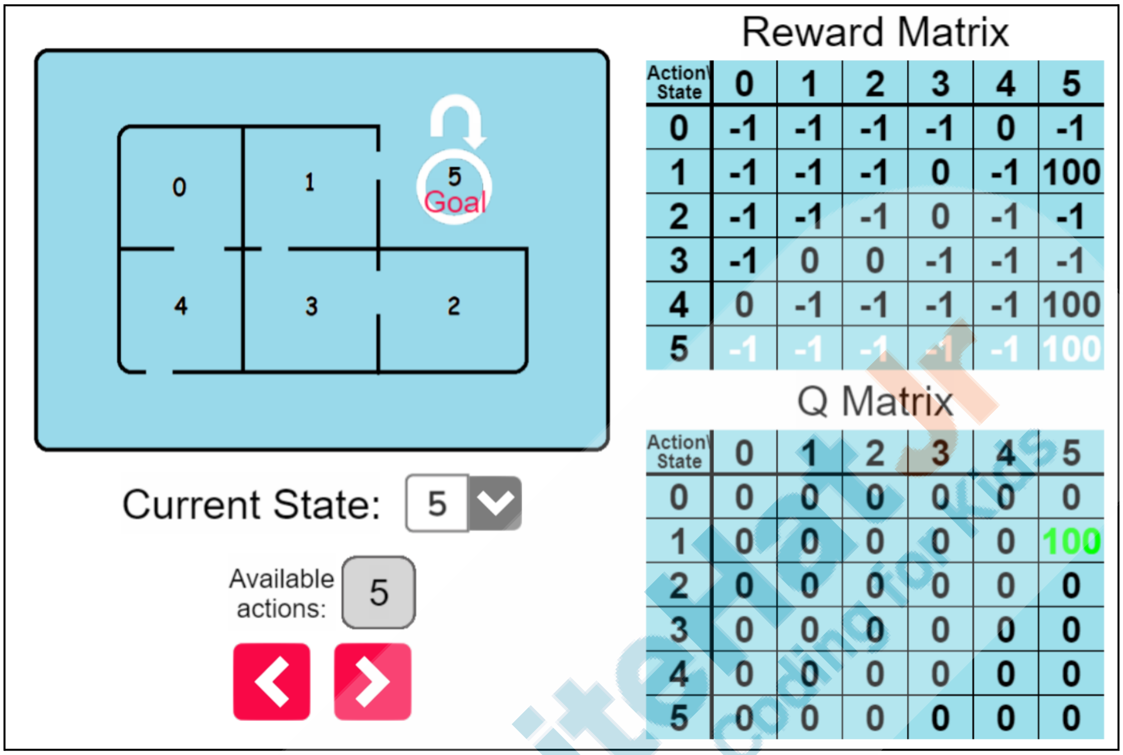
Q Matrix

Action\ State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

If I choose 5 then it’s a goal and according to the Reward matrix, the Q-Matrix will be updated.

New State → Room 5
Available Actions → None

Since we have reached the goal no further action is needed!



Current State: 5

Available actions: 5

Reward Matrix

Action\State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	-1	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	-1	100

Q Matrix

Action\State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	100
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Now that we have understood the Q-Learning algorithm works, let's understand how this can be done programmatically.

To find the best action Q-learning uses **Bellman equation**.

Bellman equation is used to look at the Q-matrix and current reward and helps decide the action taken by the Agent for maximizing the reward.

Next_reward(state,action) = Current_reward(state,action)+ gamma x Q-matrix [Max (next_action,all actions)]

Let's continue with the same example as we had taken in the last class to understand this equation.

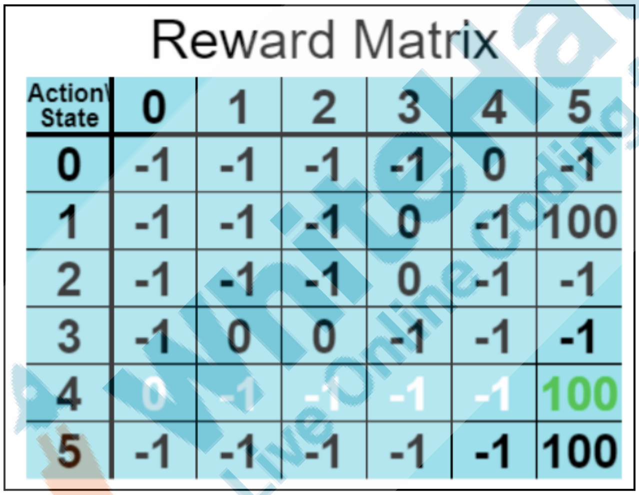
The current state is 4. Now the random action can be 0 or 5.

While exploring, suppose the agent chooses action 5. Then whether to take this action or not should be decided depending upon the Q-matrix. Thus, the reward has to be updated in the Q-matrix.

In Bellman's equation, the first term at the R.H.S is the reward for the current state and action.

The current state is 4 and the action is 5 so the reward is 100 while we look into the Reward matrix.

Current_reward(state,action)=Current_reward(4,5)=100



Reward Matrix

Action\ State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	-1	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	-1	100

Next is **gamma**. Gamma is the discount factor that varies between 0.8 to 0.99. We'll take it as 0.8 for our agent.

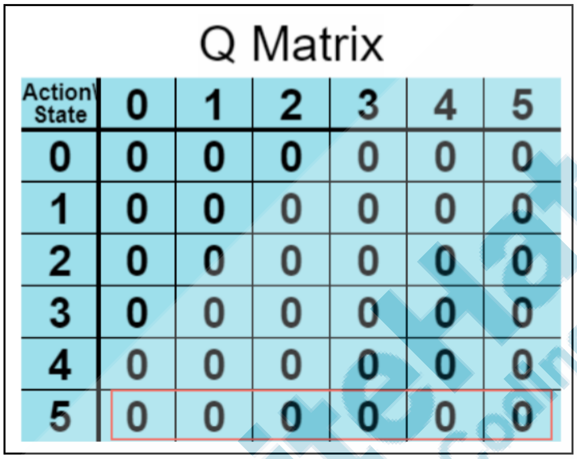
gamma=0.8

Note: *We are not diving deep for the mathematical calculation of gamma.*

q-matrix[Max(next_action, All Actions))] =

q-matrix[Max(5,)] = 0

This will look into the row with index number 5 and search for the maximum value of reward in the Q-matrix. This is 0.



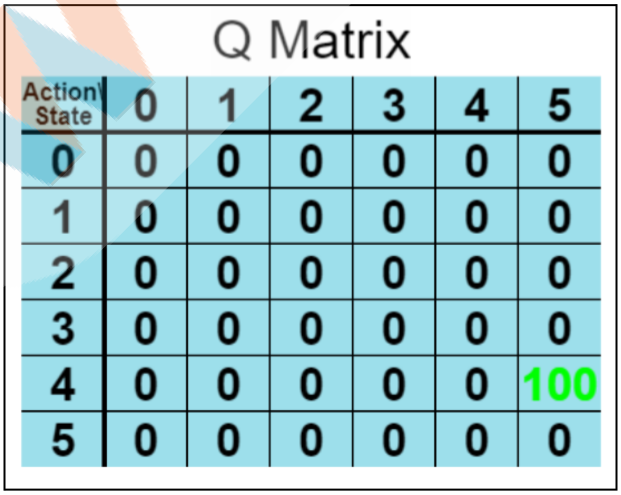
Q Matrix

Action\ State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

So, The next reward is calculated as:

Next_reward(state,action)=100 + 0.8 * 0

This reward is updated in the Q-matrix for state 4 and action 5.

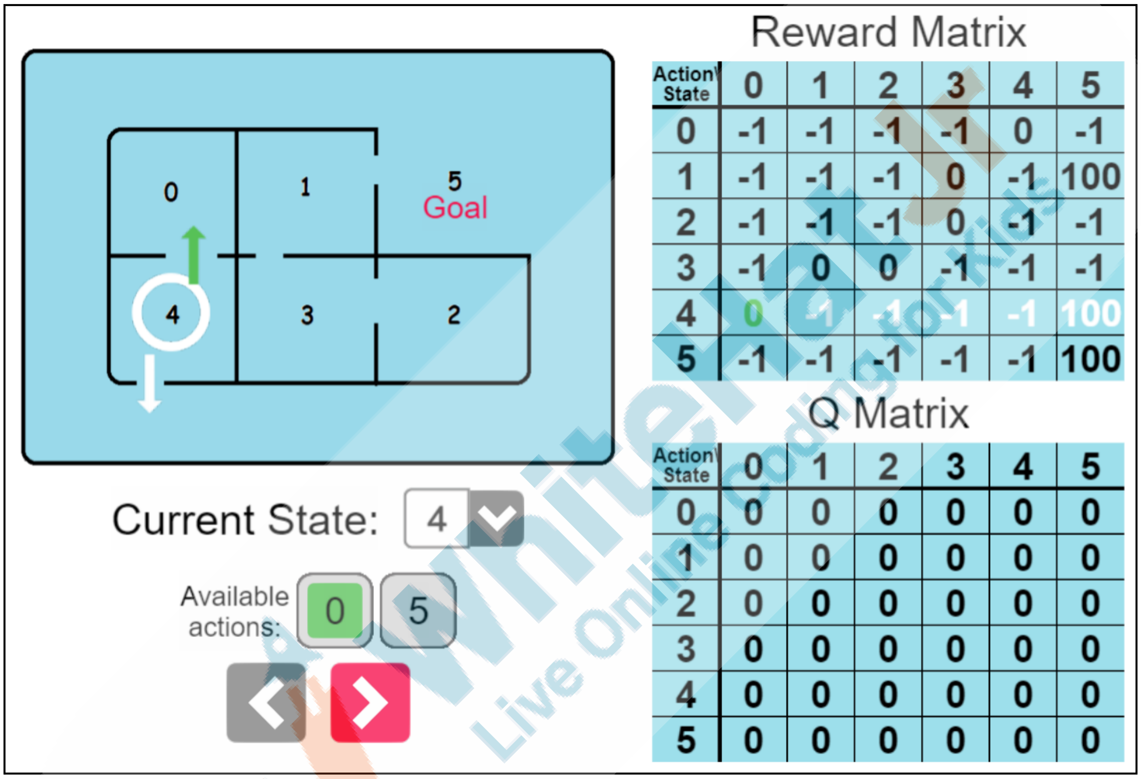


Q Matrix

Action\ State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	100
5	0	0	0	0	0	0

Similarly, let’s try with state 4 and action 0. It will give only 4 as the next state.

Note: *The teacher can take similar examples for state 4 and action 0. But since we are not reaching the goal the updated value in Q-matrix will be 0.*



Current State: 4

Available actions: 0 5

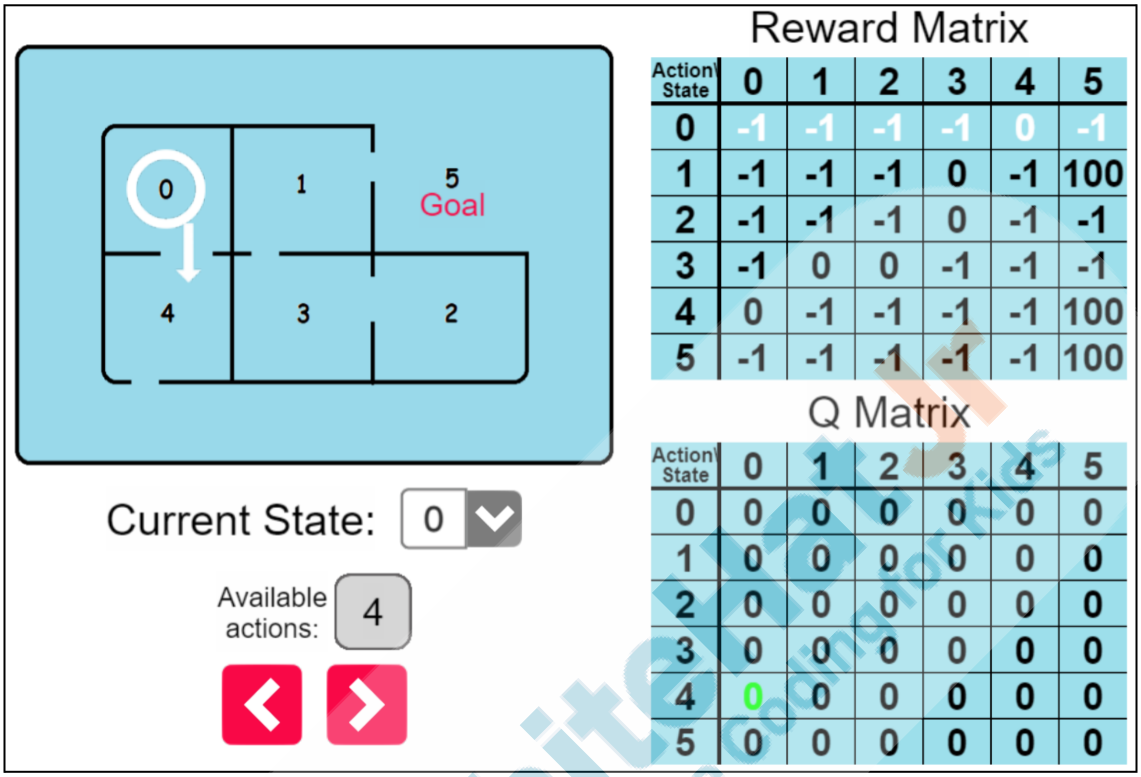
Reward Matrix

Action\ State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	-1	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	-1	100

Q Matrix

Action\ State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

The reward is 0 and hence the Q-matrix will be updated accordingly.



Current State: 0

Available actions: 4

Reward Matrix

Action\State	0	1	2	3	4	5
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	-1	-1
4	0	-1	-1	-1	-1	100
5	-1	-1	-1	-1	-1	100

Q Matrix

Action\State	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Now that it is clear to you that Q-matrix and Bellman equation is an important part of the Q-learning algorithm. Let's start implementing it.

Teacher Stops Screen Share

So now it's your turn.
Please share your screen with me.

Teacher Starts Slideshow
Slide 13 to 15
Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you.
Can you solve it?

Let's try. I will guide you through it.

Teacher Ends Slideshow

STUDENT-LED ACTIVITY - 20 mins

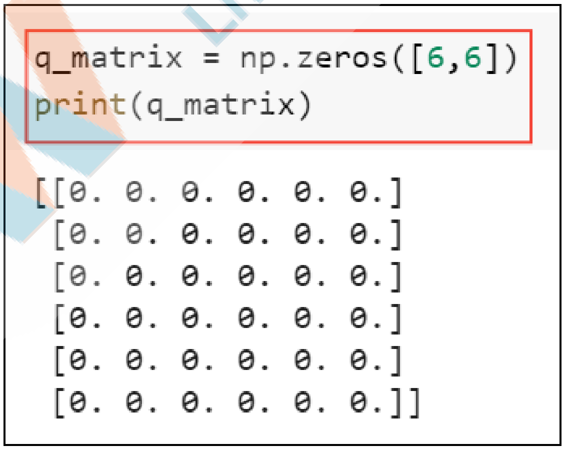
- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

Student Initiates Screen Share

ACTIVITY

- **Create Q-matrix**
- **Write Bellman's equation to update Q-matrix**

Teacher Action	Student Action
Open Student Activity 1 for previous class code. In this, we had created a Reward matrix and according to the given state, we've found the possible actions. 1. First, create a Q-matrix. Initially, all zeroes should be there. Create a numpy array of size 6X6. Print this matrix.	

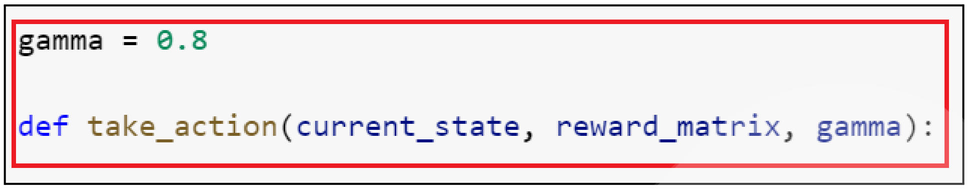


```
q_matrix = np.zeros([6,6])
print(q_matrix)
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

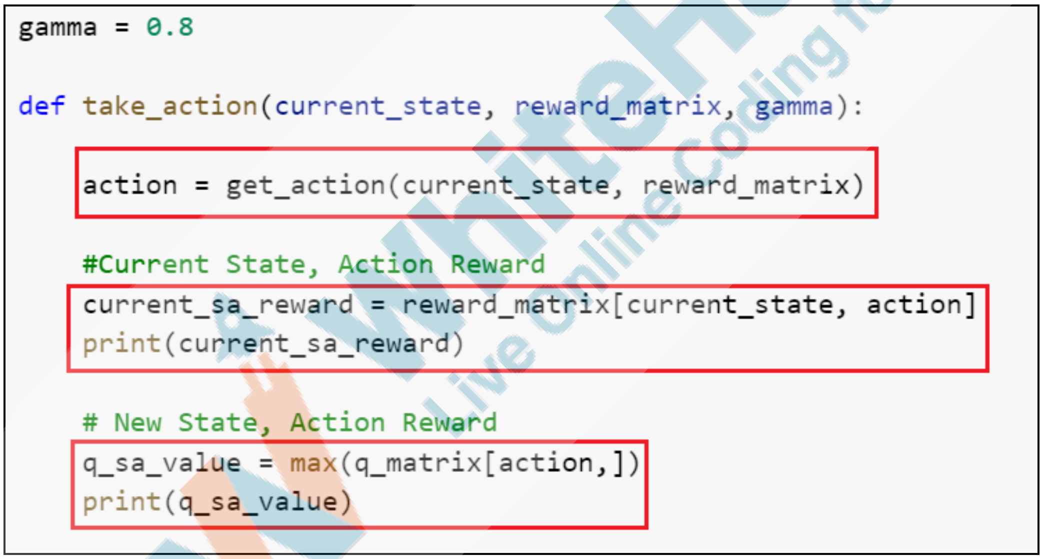
2. Define **gamma** and a function to take the action. We know that the Bellman equation is required to take the	

next action. It takes the **current state, Reward matrix** and **gamma.**



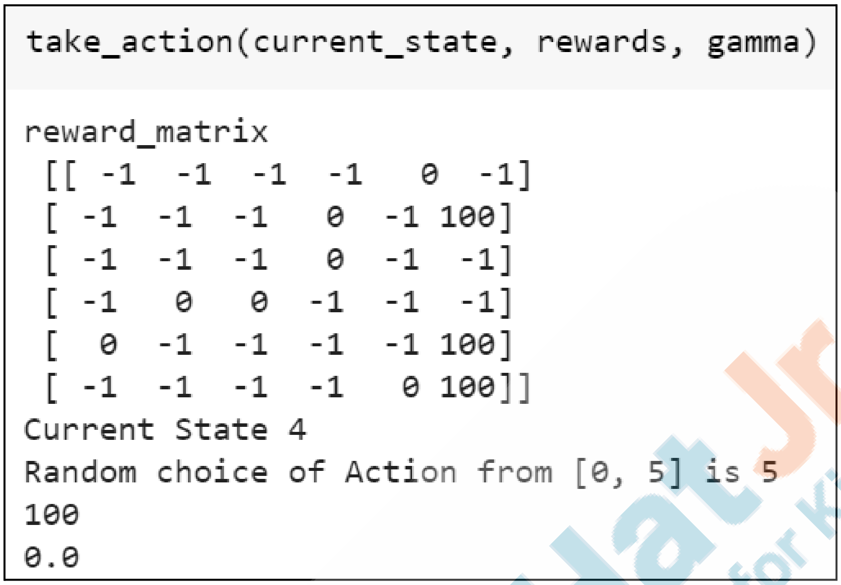
```
gamma = 0.8

def take_action(current_state, reward_matrix, gamma):
```

3. Call the **get_action()** function to get the available actions.
4. Create a variable **current_sa_reward** to store the current reward.
5. Check the maximum value of reward for action in Q-matrix.



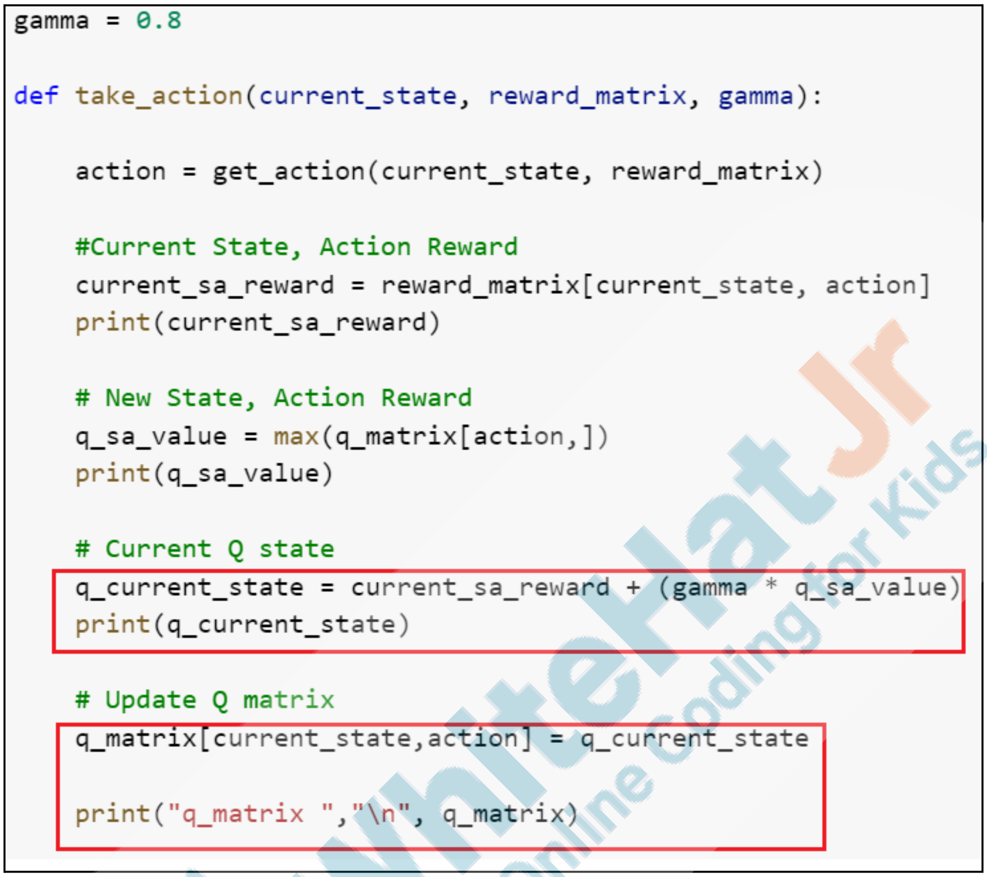
```
gamma = 0.8

def take_action(current_state, reward_matrix, gamma):

    action = get_action(current_state, reward_matrix)

    #Current State, Action Reward
    current_sa_reward = reward_matrix[current_state, action]
    print(current_sa_reward)

    # New State, Action Reward
    q_sa_value = max(q_matrix[action,])
    print(q_sa_value)
```

6. Call the **take_action()** function and check the output.



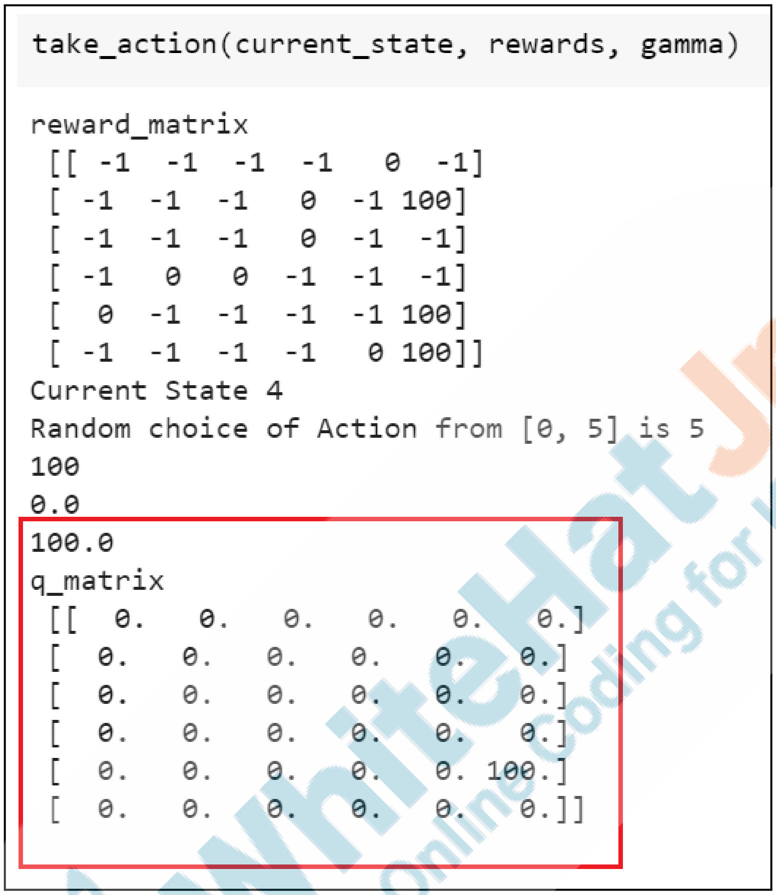
```
take_action(current_state, rewards, gamma)

reward_matrix
 [[ -1  -1  -1  -1   0  -1]
 [ -1  -1  -1   0  -1 100]
 [ -1  -1  -1   0  -1  -1]
 [ -1   0   0  -1  -1  -1]
 [  0  -1  -1  -1  -1 100]
 [ -1  -1  -1  -1   0 100]]
Current State 4
Random choice of Action from [0, 5] is 5
100
0.0
```

7. Find the reward after taking the action using Bellman equation.
8. Update the current Q-matrix with the calculated reward.
9. Print the Q-matrix.



```
gamma = 0.8

def take_action(current_state, reward_matrix, gamma):

    action = get_action(current_state, reward_matrix)

    #Current State, Action Reward
    current_sa_reward = reward_matrix[current_state, action]
    print(current_sa_reward)

    # New State, Action Reward
    q_sa_value = max(q_matrix[action,])
    print(q_sa_value)

    # Current Q state
    q_current_state = current_sa_reward + (gamma * q_sa_value)
    print(q_current_state)

    # Update Q matrix
    q_matrix[current_state,action] = q_current_state

    print("q_matrix ","\n", q_matrix)
```

10. Run the code and check the Q-matrix.



```
take_action(current_state, rewards, gamma)
```

```
reward_matrix
 [[ -1  -1  -1  -1   0  -1]
 [ -1  -1  -1   0  -1 100]
 [ -1  -1  -1   0  -1  -1]
 [ -1   0   0  -1  -1  -1]
 [  0  -1  -1  -1  -1 100]
 [ -1  -1  -1  -1   0 100]]
Current State 4
Random choice of Action from [0, 5] is 5
100
0.0
100.0
q_matrix
 [[  0.    0.    0.    0.    0.    0.]
 [  0.    0.    0.    0.    0.    0.]
 [  0.    0.    0.    0.    0.    0.]
 [  0.    0.    0.    0.    0.    0.]
 [  0.    0.    0.    0.    0. 100.]
 [  0.    0.    0.    0.    0.    0.]]
```

Now, you can see at Q-matrix(4,5), the reward is 100. Thus, the reward is maximum if the agent takes 5 as the next action.

We can try the same by changing the current state and finding the Q-matrix depending upon the next action.

Great!!

So, we were able to calculate the new reward using the Bellman equation. The Q-matrix is used to take the next action.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Teacher Starts Slideshow Slide 16 to 21
Activity details **Following are the WRAP-UP session deliverables:** ● Appreciate the student. ● Revise the current class activities. ● Discuss the quizzes.
WRAP-UP QUIZ Click on In-Class Quiz
Continue WRAP-UP Session Slide 22 to 27
Activity Details **Following are the session deliverables:** ● Explain the facts and trivia ● Next class challenge ● Project for the day ● Additional Activity (Optional)
FEEDBACK ● **Appreciate and compliment the student for trying to learn a difficult concept.** ● **Get to know how they are feeling after the session.** ● **Review and check their understanding.**

Teacher Action	Student Action
You get “hats-off” for your excellent work!	*Make sure you have given at least 2 hats-off during the class for:*

In the next class, we'll make the agent find the best path and reach the goal using the Q-matrix.	Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections	
Teacher Clicks ✖ End Class	

ACTIVITY LINKS

Activity Name	Description	Links
Teacher Activity 1	Reward Matrix	https://whitehatjrcontent.s3.ap-south-1.amazonaws.com/Teacher-Resources/COCOS_Applets/POC/Coding/SimpleQ-RL/rewardsOnly/index.html
Teacher Activity 2	Q-Matrix	https://whitehatjrcontent.s3.ap-south-1.amazonaws.com/Teacher-Resources/COCOS_Applets/POC/Coding/SimpleQ-RL/noEpisodes/index.html
Teacher Activity 3	Previous Code (C123)	https://colab.research.google.com/drive/1MbRRulO1qM_w7UwdF7CtgzK6E3FWUnhJ?usp=sharing

Teacher Activity 4	Reference Code	https://colab.research.google.com/drive/1tEz6SXS3unMYvWOPqaeiuGp1hMNyJLzI?usp=sharing
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/a4350271-d767-4133-992d-fb35ab25264c.pdf
Teacher Reference 2	Project Solution	https://colab.research.google.com/drive/1zm5JIb8WW3m-SybSaeEkOwSu_NN2Jv-8?usp=sharing
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/7504f080-8ec6-4612-8628-b343f00f37fd.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/33a00463-7066-43c6-b526-7b2a4d30396a.pdf
Student Activity 1	Boilerplate Code	https://colab.research.google.com/drive/1muekBviAZpSAp7YYz6AfiqZIkTgeO-Oi?usp=sharing