Artificial Intelligence(AI)

Team members:

1- Yazan moqanasa 12029260

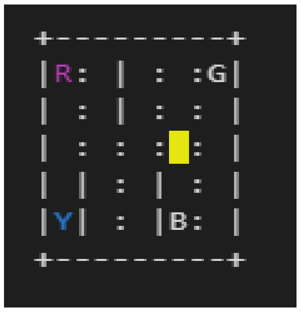
2- Mohammad soqy 12010807

The name of the game environment : **Taxi-v3**

Under the supervision of Dr : Baha Ghalib Thabet

Taxi game environment has been tested, Based on what the course instructor explained about Reinforcement learning (RL) and the basic Markov decision-making process (MVC),And by implementing some software commands to apply to reinforcement learning and basic Markov decision-making.

At the beginning, we simulate the taxi game, train it, and define the state area (observation_space) and the size of the workspace.

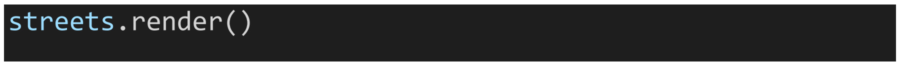


```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

# Red - 0, Green - 1, Yellow - 2, and Blue - 3 to capture configure it
street = gym. make ("Taxi-v3"). New env# versions are constantly

## Red - 0, Green - 1, Yellow - 2, and Blue - 3 to capture
# Each state is defined by 4 tuple entries: （taxi_row, taxi_col, passenger_location, destination)

#Exploring Open Street Map using Pandas and the Python API

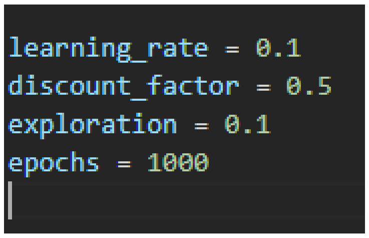


```
streets.render()
```

Then we set up the Q-Table as Q-Learning - Simplified Overview
Let's say the taxi has to cross a maze and reach the end point. There are mines in the way, and the robot can only move one piece at a time. If the taxi goes up a mine, the taxis are dead. The taxi must reach the end point in the shortest possible time.

If the taxi moves wrong then the point loss is 100 for example and the game ends.
If the bot reaches the final goal, the bot gets 100 points for the test.

And we initialized the hyperparameters as
learning_ratio = 0.1,
 discount_factor = 0.5,
exploration = 0.1,
epochs = 1000



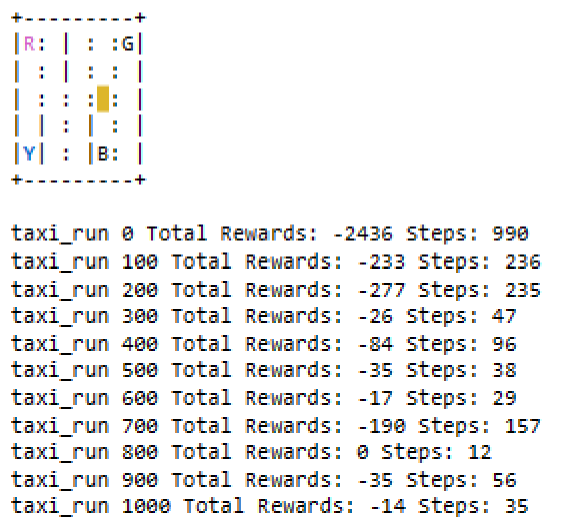
```
learning_rate = 0.1
discount_factor = 0.5
exploration = 0.1
epochs = 1000
```

We change epochs according to our needs:
5000 and **10000**

Goal :

(The goal is to train the model with a different number of epochs to find the best such policy It increases rewards and decreases the number of steps from S to G.)

We trained with different epoch settings: 1000, 5000, and 10000 The readings were according to the following tables:

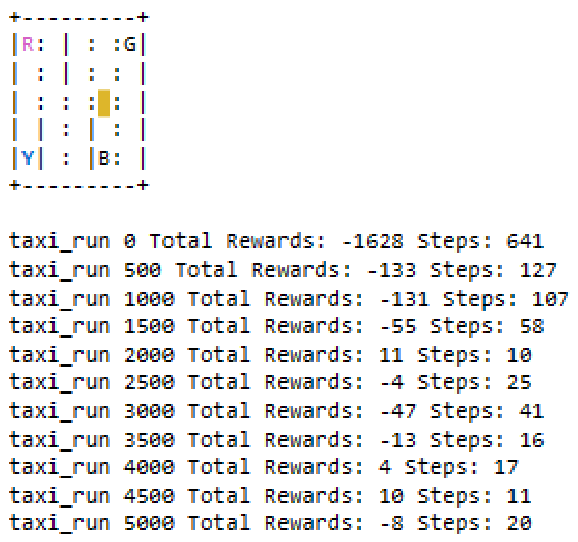


```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+

taxi_run 0 Total Rewards: -2436 Steps: 990
taxi_run 100 Total Rewards: -233 Steps: 236
taxi_run 200 Total Rewards: -277 Steps: 235
taxi_run 300 Total Rewards: -26 Steps: 47
taxi_run 400 Total Rewards: -84 Steps: 96
taxi_run 500 Total Rewards: -35 Steps: 38
taxi_run 600 Total Rewards: -17 Steps: 29
taxi_run 700 Total Rewards: -190 Steps: 157
taxi_run 800 Total Rewards: 0 Steps: 12
taxi_run 900 Total Rewards: -35 Steps: 56
taxi_run 1000 Total Rewards: -14 Steps: 35
```

Algorithm :
Epochs 1000
for taxi_run in range (**Epochs**) ....
if (taxi_run) % 100 ==0:

Epochs 1000 Print if epoch % 100 == 0:	Rewards	Steps
Epoch 0	-2436	990
Epoch 100	-233	236
Epoch 200	-277	235
Epoch 300	-26	47
Epoch 400	-84	96
Epoch 500	-35	38
Epoch 600	-17	28
Epoch 700	-190	157
Epoch 800	0	12
Epoch 900	-35	56
Epoch 1000	-14	35

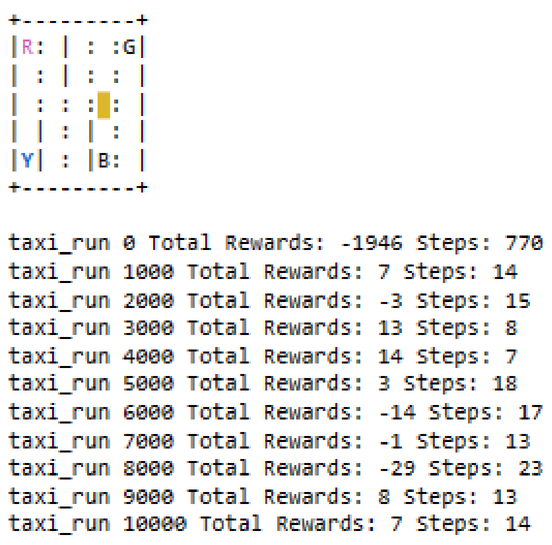


```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+

taxi_run 0 Total Rewards: -1628 Steps: 641
taxi_run 500 Total Rewards: -133 Steps: 127
taxi_run 1000 Total Rewards: -131 Steps: 107
taxi_run 1500 Total Rewards: -55 Steps: 58
taxi_run 2000 Total Rewards: 11 Steps: 10
taxi_run 2500 Total Rewards: -4 Steps: 25
taxi_run 3000 Total Rewards: -47 Steps: 41
taxi_run 3500 Total Rewards: -13 Steps: 16
taxi_run 4000 Total Rewards: 4 Steps: 17
taxi_run 4500 Total Rewards: 10 Steps: 11
taxi_run 5000 Total Rewards: -8 Steps: 20
```

Algorithm :
Epochs 5000
for taxi_run in range (**Epochs**) ....
if (taxi_run) % 500 ==0: ....

Epochs 5000 Print if epoch % 500 == 0:	Rewards	Steps
Epoch 0	-1628	641
Epoch 500	-133	127
Epoch 1000	-131	107
Epoch 1500	-55	58
Epoch 2000	11	10
Epoch 2500	-4	25
Epoch 3000	-47	41
Epoch 3500	-13	16
Epoch 4000	4	17
Epoch 4500	10	11
Epoch 5000	-8	20



```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+

taxi_run 0 Total Rewards: -1946 Steps: 770
taxi_run 1000 Total Rewards: 7 Steps: 14
taxi_run 2000 Total Rewards: -3 Steps: 15
taxi_run 3000 Total Rewards: 13 Steps: 8
taxi_run 4000 Total Rewards: 14 Steps: 7
taxi_run 5000 Total Rewards: 3 Steps: 18
taxi_run 6000 Total Rewards: -14 Steps: 17
taxi_run 7000 Total Rewards: -1 Steps: 13
taxi_run 8000 Total Rewards: -29 Steps: 23
taxi_run 9000 Total Rewards: 8 Steps: 13
taxi_run 10000 Total Rewards: 7 Steps: 14
```

Algorithm :

Epochs 10000

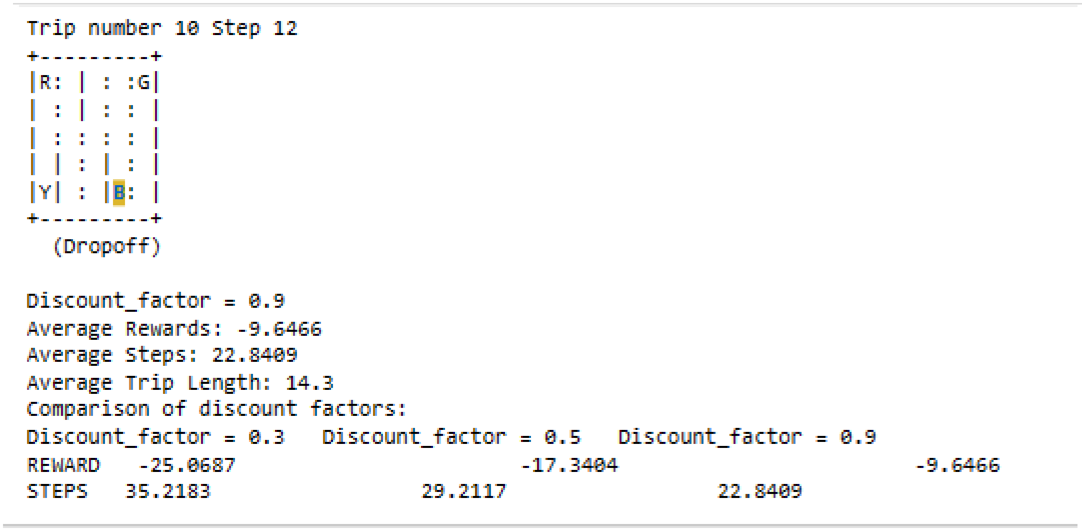
for taxi_run in range (**Epochs**) ....

if (taxi_run) % 1000 ==0: ....

Epochs 10000 Print if epoch % 1000 == 0:	Rewards	Steps
Epoch 0	-1946	770
Epoch 1000	7	-3
Epoch 2000	-3	8
Epoch 3000	13	8
Epoch 4000	14	7
Epoch 5000	3	8
Epoch 6000	-14	17
Epoch 7000	-1	13
Epoch 8000	-29	23
Epoch 9000	8	13
Epoch 10000	7	14

The best thing is Epochs 10000 because it has fewer steps and higher rewards compared to Epochs 1000 and Epochs 5000. Take the best policy.

We repeated the last training with 10,000 epochs with different discount_factor values as indicated
In the following table,
After each training, we had the agent play 10 times and calculated the average rewards and
Average steps as measures of evaluation.
We compared the
results and evaluated which discount factor yielded better rewards and fewer steps.



```
Trip number 10 Step 12
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (Dropoff)

Discount_factor = 0.9
Average Rewards: -9.6466
Average Steps: 22.8409
Average Trip Length: 14.3
Comparison of discount factors:
Discount_factor = 0.3   Discount_factor = 0.5   Discount_factor = 0.9
REWARD   -25.0687                      -17.3404                        -9.6466
STEPS    35.2183                 29.2117                 22.8409
```

Play 10 times	Average Rewards	Average Steps
Discount_factor = 0.3	25.06-	35.21
Discount_factor = 0.5	17.34-	29.21
Discount_factor = 0.9	9.64-	22.84

he discount factor = 0.9 was calculated

Average rewards calculated: -9.6466
Average steps calculated: 22.8409
The average flight length was calculated: 14.3
We worked on comparing discount factors:
Discount Factor = 0.3 Discount Factor = 0.5 Discount Factor = 0.9

```
              0.3            0.5                 0.9
REWARD   -25.0687      -17.3404          -9.6466
STEPS    35.2183        29.2117          22.8409
```

```python
import gym
import random
import numpy as np

#Red – 0 , Green – 1, Yellow – 2, and Blue – 3 for pick up
streets = gym.make("Taxi-v3").env #New versions keep getting released; if -v3
doesn't work, try -v2 or -v4


##Red – 0 , Green – 1, Yellow – 2, and Blue – 3 for pick up
#Each state is defined by a 4 entries tuple:  (taxi_row, taxi_col,
passenger_location, destination)
initial_state = streets.encode(2, 3, 2, 0)
streets.s = initial_state
streets.render()

#State Space:  25 possible taxi positions, 5 possible locations of the passenger
# 25*5*4 = 500


#Action space:6 --> N,S,E,W, DROP-OFF, PICKUP
#Rewards: CORRECT FINAL DEST. +20, STEP -1, INCORRECT PICK/DROP -10

q_table = np.zeros([streets.observation_space.n, streets.action_space.n]) # 500 ,
6
#q_table.size
# a 2D array that represent every possible state and action in the virtual space
and initialize all of them to 0
total_reward_G = 0
learning_rate = 0.1
discount_factor = 0.5
exploration = 0.1
epochs = 1000

for taxi_run in range(epochs): #Start training (the agent plays the number of
epochs)
    state = streets.reset()
    done = False
    total_reward_G = 0
    steps=0
    while not done:#each epoch/play contains this number of actions, starting from
pickup a passenger until drop-off
        steps +=1
        random_value = random.uniform(0, 1)
        if (random_value < exploration):
            action = streets.action_space.sample() # Explore a random action
        else:
            action = np.argmax(q_table[state]) # Return the action with the
highest q-value

        next_state, reward, done, info = streets.step(action) # Do the above
action

        prev_q = q_table[state, action]
        next_max_q = np.max(q_table[next_state])
        # see RL-2 PPT file --- slide# 5
        new_q = (1 - learning_rate) * prev_q + learning_rate * (reward +
discount_factor * next_max_q)
        total_reward_G += reward
        q_table[state, action] = new_q
        #streets.render()
        state = next_state

    if (taxi_run) % 100 ==0:
         print('taxi_run {} Total Rewards: {} Steps:
{}'.format(taxi_run,total_reward_G,steps))
         #streets.render()
```

```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+

taxi_run 0 Total Rewards: -1578 Steps: 618
taxi_run 100 Total Rewards: -382 Steps: 313
taxi_run 200 Total Rewards: -209 Steps: 158
taxi_run 300 Total Rewards: -73 Steps: 67
taxi_run 400 Total Rewards: -106 Steps: 91
taxi_run 500 Total Rewards: -89 Steps: 65
taxi_run 600 Total Rewards: -55 Steps: 58
taxi_run 700 Total Rewards: 12 Steps: 9
taxi_run 800 Total Rewards: -2 Steps: 23
taxi_run 900 Total Rewards: 1 Steps: 20
```

```python
from IPython.display import clear_output
from time import sleep
import gym
import random
import numpy as np

streets = gym.make("Taxi-v3").env
streets.render()

discount_factors = [0.3, 0.5, 0.9]
avg_rewards = []
avg_steps = []

for discount_factor in discount_factors:
    q_table = np.zeros([streets.observation_space.n, streets.action_space.n])  #
500, 6
    total_rewards = []
    total_steps = []

    epochs = 10000

    for taxi_run in range(epochs):
        state = streets.reset()
        done = False
        total_reward_G = 0
        steps = 0

        while not done:
            steps += 1
            random_value = random.uniform(0, 1)

            if random_value < exploration:
                action = streets.action_space.sample()
            else:
                action = np.argmax(q_table[state])

            next_state, reward, done, info = streets.step(action)

            prev_q = q_table[state, action]
            next_max_q = np.max(q_table[next_state])
            new_q = (1 - learning_rate) * prev_q + learning_rate * (reward +
discount_factor * next_max_q)
            q_table[state, action] = new_q
            state = next_state

            total_reward_G += reward
        if taxi_run % 1000 == 0:
            print('Taxi Run {} Total Rewards: {} Steps: {}'.format(taxi_run,
total_reward_G, steps))

        total_rewards.append(total_reward_G)
        total_steps.append(steps)

    avg_reward = np.mean(total_rewards)
    avg_step = np.mean(total_steps)

    avg_rewards.append(avg_reward)
    avg_steps.append(avg_step)

    lengths = []
    for tripnum in range(1, 11):
        state = streets.reset()
        done = False
        trip_length = 0

        while not done and trip_length < 25:
            action = np.argmax(q_table[state])
            next_state, reward, done, info = streets.step(action)
            clear_output(wait=True)
            print("Trip number " + str(tripnum) + " Step " + str(trip_length))
            print(streets.render(mode='ansi'))
            sleep(.2)
            state = next_state
            trip_length += 1
        lengths.append(trip_length)

    avg_len = np.mean(lengths)
    print(f"step={taxi_run}")
    print(f"Discount_factor = {discount_factor}")
    print(f"Average Rewards: {avg_reward}")
    print(f"Average Steps: {avg_step}")
    print(f"Average Trip Length: {avg_len}")

print("Comparison of discount factors:")
print("Discount_factor = 0.3\tDiscount_factor = 0.5\tDiscount_factor = 0.9")
print(f"REWARD\t {avg_rewards[0]}\t\t\t{avg_rewards[1]}\t\t\t{avg_rewards[2]}")
print(f"STEPS\t{ avg_steps[0]}\t\t\t{avg_steps[1]}\t\t\t{avg_steps[2]}")
if taxi_run % 1000 == 0:
    print('Taxi Run {} Total Rewards: {} Steps: {}'.format(taxi_run,
total_reward_G, steps))
```

```
Trip number 10 Step 11
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
  (Dropoff)

Discount_factor = 0.9
Average Rewards: -9.6149
Average Steps: 22.865
Average Trip Length: 13.7
Comparison of discount factors:
Discount_factor = 0.3    Discount_factor = 0.5    Discount_factor = 0.9
REWARD  -25.0493                -17.2015                 -9.6149
Steps   -25.0493                29.2222                  22.865
```