

LABORATORIO_03

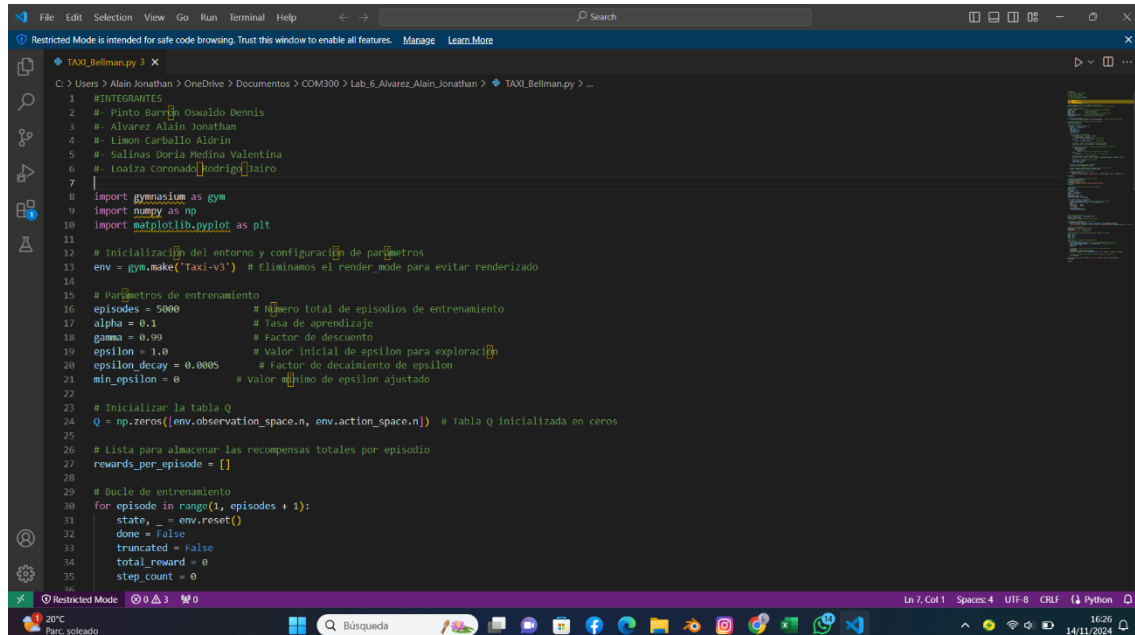
MATERIA: INTELIGENCIA ARTIFICIAL

INTEGRANTES

- Pinto Barrón Oswaldo Dennis
 - Alvarez Alain Jonathan
 - Limon Carballo Aldrin
 - Salinas Doria Medina Valentina
 - Loaiza Coronado Rodrigo Jairo AÑO: 2024
- Carrera: Ing. en Diseño y Animación Digital

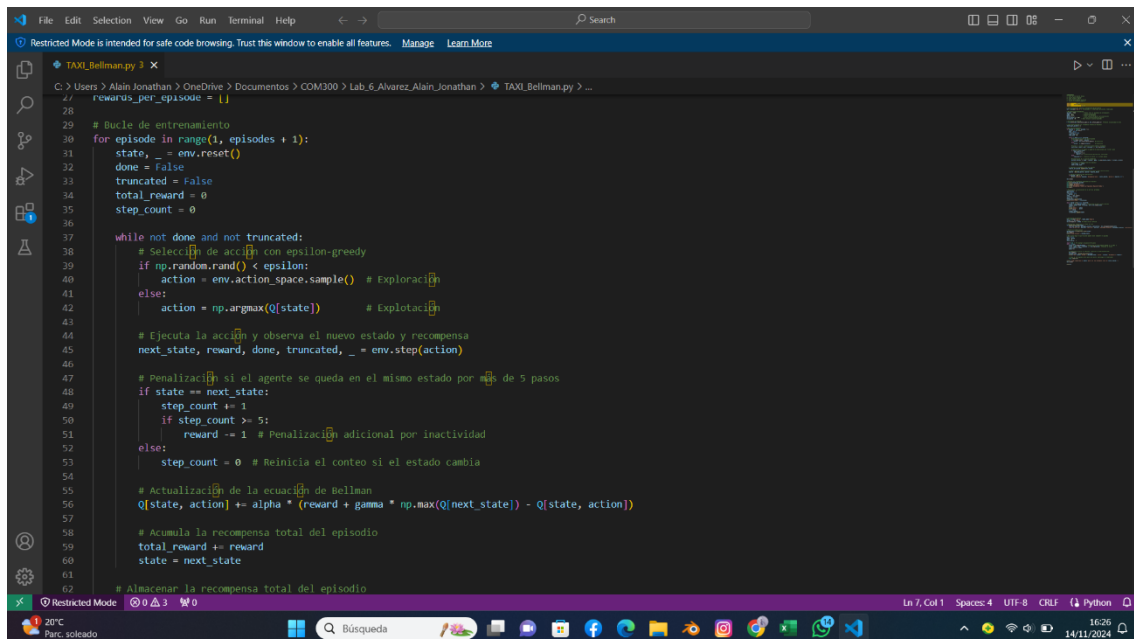
1.- RESUMEN EXPLICANDO LO TRABAJADO

Primero importamos nuestras librerías, las que usaremos, definimos nuestros parámetros de entrenamiento, con ϵ de 1 para que pueda hacer exploración su primera vez



```
1 #INTEGRANTES
2 # - Pinto Barrón Oswaldo Dennis
3 # - Alvarez Alain Jonathan
4 # - Limon Carballo Aldrin
5 # - Salinas Doria Medina Valentina
6 # - Loaiza Coronado Rodrigo Jairo
7
8 import gymnasium as gym
9 import numpy as np
10 import matplotlib.pyplot as plt
11
12 # Inicialización del entorno y configuración de parámetros
13 env = gym.make('Taxi-v3') # Eliminamos el render_mode para evitar renderizado
14
15 # Parámetros de entrenamiento
16 episodes = 5000 # Número total de episodios de entrenamiento
17 alpha = 0.1 # Tasa de aprendizaje
18 gamma = 0.99 # Factor de descuento
19 epsilon = 1.0 # Valor inicial de epsilon para exploración
20 epsilon_decay = 0.0005 # Factor de decaimiento de epsilon
21 min_epsilon = 0 # Valor mínimo de epsilon ajustado
22
23 # Inicializar la tabla Q
24 Q = np.zeros((env.observation_space.n, env.action_space.n)) # Tabla Q inicializada en zeros
25
26 # Lista para almacenar las recompensas totales por episodio
27 rewards_per_episode = []
28
29 # Bucle de entrenamiento
30 for episode in range(1, episodes + 1):
31     state, _ = env.reset()
32     done = False
33     truncated = False
34     total_reward = 0
35     step_count = 0
```

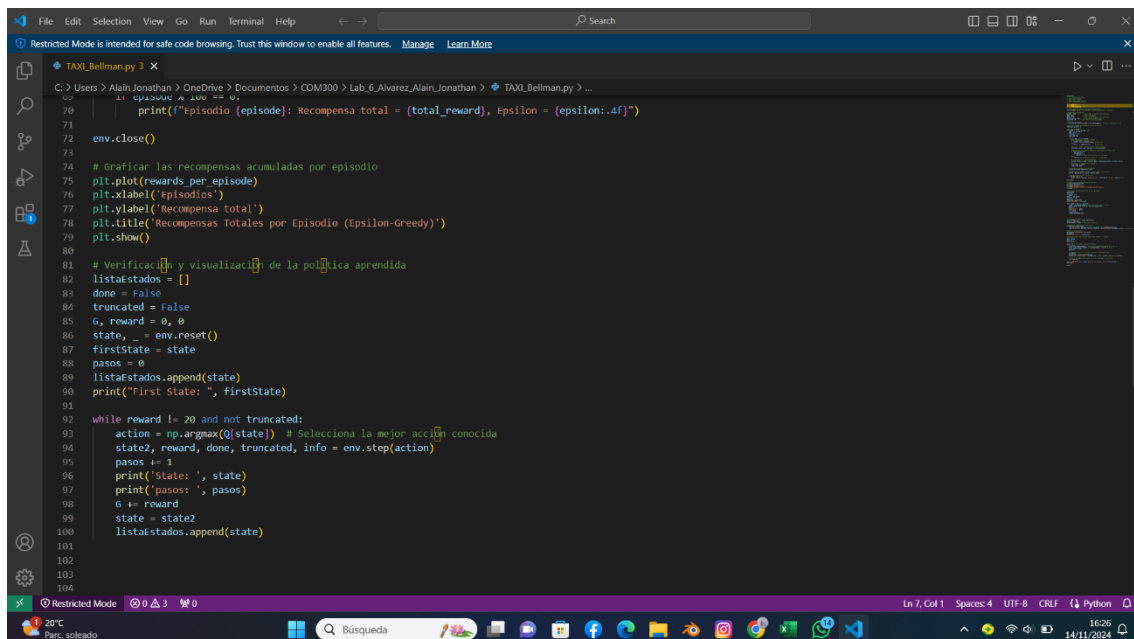
Luego con este bucle empezamos a hacer nuestra exploración y explotación, también vamos llenando nuestra tabla q, en el cual tendremos valores de acción y estado



```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

TAXI_Bellman.py 3 X
C:\Users\Alain Jonathan > OneDrive > Documentos > COM300 > Lab_6_Alvarez_Alain_Jonathan > TAXI_Bellman.py > ...
27 rewards_per_episode = 0
28
29 # Bucle de entrenamiento
30 for episode in range(1, episodes + 1):
31     state, _ = env.reset()
32     done = False
33     truncated = False
34     total_reward = 0
35     step_count = 0
36
37     while not done and not truncated:
38         # Selección de acción con epsilon-greedy
39         if np.random.rand() < epsilon:
40             action = env.action_space.sample() # Exploración
41         else:
42             action = np.argmax(Q[state]) # Explotación
43
44         # Ejecuta la acción y observa el nuevo estado y recompensa
45         next_state, reward, done, truncated, _ = env.step(action)
46
47         # Penalización si el agente se queda en el mismo estado por más de 5 pasos
48         if state == next_state:
49             step_count += 1
50             if step_count >= 5:
51                 reward -= 1 # Penalización adicional por inactividad
52             else:
53                 step_count = 0 # Reinicia el conteo si el estado cambia
54
55         # Actualización de la ecuación de Bellman
56         Q[state, action] += alpha * (reward + gamma * np.max(Q[next_state]) - Q[state, action])
57
58         # Acumula la recompensa total del episodio
59         total_reward += reward
60         state = next_state
61
62     # Almacenar la recompensa total del episodio
```

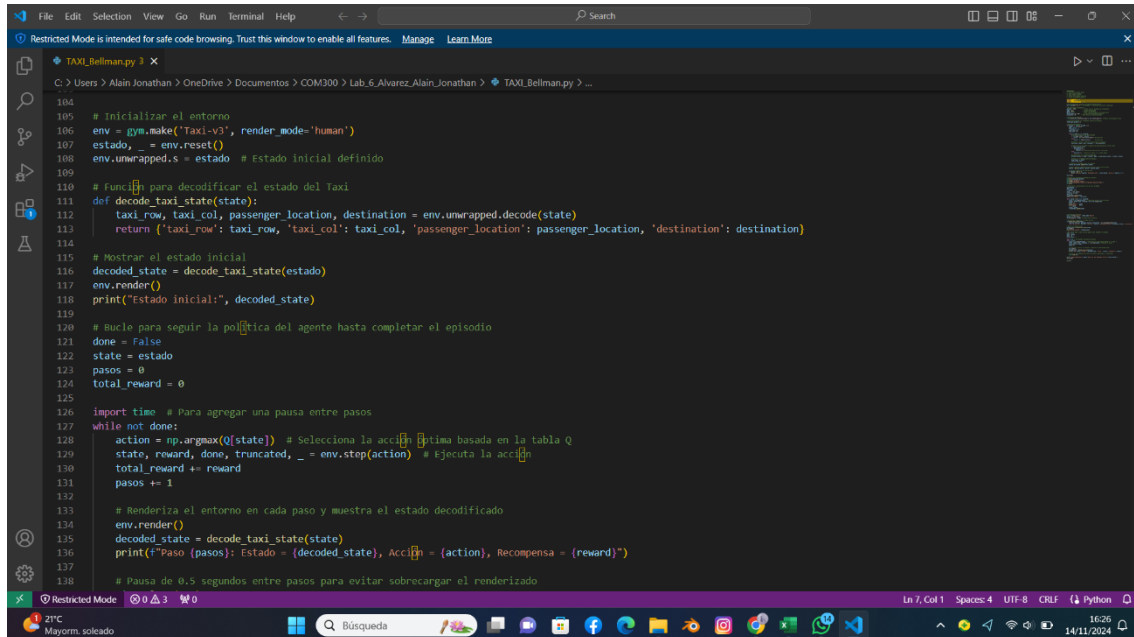
Ahora según aquella tabla hacemos de nuevo una explotación con el siguiente bucle, teniendo en cuenta ya mejores valores empezamos a hacer nuestra explotación con una mejor tabla q



```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

TAXI_Bellman.py 3 X
C:\Users\Alain Jonathan > OneDrive > Documentos > COM300 > Lab_6_Alvarez_Alain_Jonathan > TAXI_Bellman.py > ...
69 if episode % 100 == 0:
70     print(f"Episodio {episode}: Recompensa total = {total_reward}, Epsilon = {epsilon:.4f}")
71
72 env.close()
73
74 # Graficar las recompensas acumuladas por episodio
75 plt.plot(rewards_per_episode)
76 plt.xlabel('Episodios')
77 plt.ylabel('Recompensa total')
78 plt.title('Recompensas Totales por Episodio (Epsilon-Greedy)')
79 plt.show()
80
81 # Verificación y visualización de la política aprendida
82 listEstados = []
83 done = False
84 truncated = False
85 G, reward = 0, 0
86 state, _ = env.reset()
87 firstState = state
88 pasos = 0
89 listEstados.append(state)
90 print(f"First State: ", firstState)
91
92 while reward != 20 and not truncated:
93     action = np.argmax(Q[state]) # Selecciona la mejor acción conocida
94     state2, reward, done, truncated, info = env.step(action)
95     pasos += 1
96     print(f'State: ', state)
97     print(f'pasos: ', pasos)
98     G += reward
99     state = state2
100     listEstados.append(state)
101
102
103
104
```

Al ultimo solamente graficamos y mostramos nuestra línea de código



```
1084
1085 # Inicializar el entorno
1086 env = gym.make('Taxi-v3', render_mode='human')
1087 estado, _ = env.reset()
1088 env.unwrapped.s = estado # Estado inicial definido
1089
1090 # Función para decodificar el estado del Taxi
1091 def decode_taxi_state(state):
1092     taxi_row, taxi_col, passenger_location, destination = env.unwrapped.decode(state)
1093     return {'taxi_row': taxi_row, 'taxi_col': taxi_col, 'passenger_location': passenger_location, 'destination': destination}
1094
1095 # Mostrar el estado inicial
1096 decoded_state = decode_taxi_state(estado)
1097 env.render()
1098 print("Estado inicial:", decoded_state)
1099
1100 # Bucle para seguir la política del agente hasta completar el episodio
1101 done = False
1102 state = estado
1103 pasos = 0
1104 total_reward = 0
1105
1106 import time # Para agregar una pausa entre pasos
1107 while not done:
1108     action = np.argmax(Q[state]) # Selecciona la acción óptima basada en la tabla Q
1109     state, reward, done, truncated, _ = env.step(action) # Ejecuta la acción
1110     total_reward += reward
1111     pasos += 1
1112
1113     # Renderiza el entorno en cada paso y muestra el estado decodificado
1114     env.render()
1115     decoded_state = decode_taxi_state(state)
1116     print(f"Paso {pasos}: Estado = {decoded_state}, Acción = {action}, Recompensa = {reward}")
1117
1118     # Pausa de 0.5 segundos entre pasos para evitar sobrecargar el renderizado
```