### Bibliotecas Utilizadas

O código utiliza várias bibliotecas Python, incluindo numpy para manipulação de matrizes, matplotlib para criar gráficos e animações, pandas para criar tabelas de dados, e FuncAnimation da biblioteca matplotlib.animation para criar uma animação.

### Parâmetros de Configuração do Modelo

start\_state: Define o ponto de partida do agente no ambiente.

gamma: Representa o fator de desconto utilizado no algoritmo de aprendizado por reforço Q-learning.

n\_actions: Define o número de ações que o agente pode realizar. Neste caso, são 4 ações possíveis (cima, baixo, esquerda, direita).

epsilon: Determina a probabilidade de o agente realizar uma ação aleatória (exploração) em oposição a uma ação gananciosa (exploração).

num\_iterations: Especifica o número de tentativas que o agente fará para chegar ao objetivo.

living\_penalty: Define a penalidade por continuar tentando (ou seja, fazer movimentos que não levam ao objetivo).

parede: Representa a penalidade por colidir com uma parede.

chegada: Define a recompensa por chegar ao objetivo.

falta: Especifica a penalidade por entrar em uma área de "falta".

### Definição do Ambiente do Labirinto

O ambiente do labirinto é representado como uma grade (matriz) em que:

0: Caminho livre.

1: Paredes.

2: Áreas de "falta" (onde o agente sofre penalidade).

9: Objetivo.

### Inicialização dos Valores Estimados de Ação (Q-values)

A matriz q\_values é inicializada com zeros e representa os valores Q para cada estado e ação possível.

Política Epsilon-Greedy

A função epsilon\_greedy\_policy define a política de tomada de decisão do agente, que pode ser aleatória (exploração) com probabilidade epsilon ou gananciosa (exploração) escolhendo a ação com o maior valor Q.

### Movimentação do Agente

A função move permite que o agente se mova no ambiente com base na ação escolhida. As ações são representadas como números (0: cima, 1: baixo, 2: esquerda, 3: direita). O código verifica se o movimento é válido (dentro do labirinto e não em uma parede) antes de executá-lo.

### Criação da Tabela Q

A função create\_q\_table cria uma tabela Q baseada nos valores Q estimados pelo agente. Ela extrai a ação com o maior valor Q para cada estado.

### Atualização dos Valores Q (Q-learning)

O código simula várias tentativas em um loop, onde o agente se move pelo ambiente. Durante o processo, ele atualiza os valores Q usando o algoritmo de aprendizado por reforço Q-learning. A fórmula usada para atualizar os valores Q é:

Q(s, a) = Q(s, a) + α \* [R + γ \* max(Q(s', a')) - Q(s, a)]

α é uma taxa de aprendizado fixa (0.1).

R é a recompensa obtida pelo agente.

γ é o fator de desconto.

s é o estado atual.

a é a ação tomada pelo agente.

s' é o próximo estado após a ação a.

a' é a ação no próximo estado.

### Atualização Gráfica

O código também cria uma animação para visualizar o agente se movendo no labirinto e sua trajetória.

### Log da Tabela Q

O código imprime um registro das tabelas Q após cada tentativa.