# Task 02

Data Science Internship – Assignment

## Unsupervised Learning

Unsupervised learning is a type of machine learning where the model is not provided with labeled data. Instead, the model finds hidden patterns and structures within the data. In this maze task, the agent learns without being explicitly told the correct paths; instead, it explores and adapts based on its environment and rewards.

## Reinforcement learning

Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with its environment. The agent takes actions, receives feedback in the form of rewards or penalties, and adjusts its behavior to maximize cumulative rewards over time.

## Method Chosen

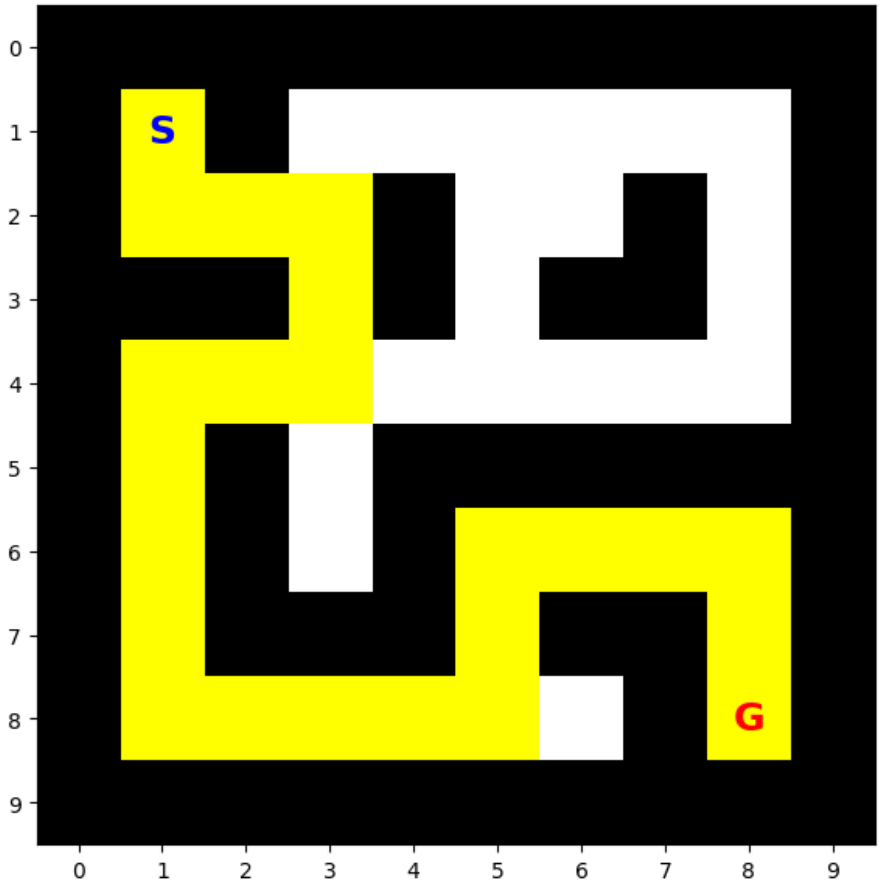
In this task, the Q-learning algorithm was employed to demonstrate unsupervised learning in navigating a maze. Q-learning, a reinforcement learning technique, allows an agent to learn optimal actions based on rewards without explicit supervision. In this scenario, the agent learns to navigate from a starting point to a goal within the maze by trial and error, adjusting its actions based on cumulative rewards.

To ensure success, Q-learning was implemented with a discount factor (gamma), learning rate (alpha), and exploration-exploitation balance (epsilon). This allowed the agent to balance learning from past experiences and exploring new paths. Additionally, epsilon decay helped the agent shift from exploration to exploiting learned knowledge as training progressed.

The model was designed with flexibility, such as dynamically setting the start and goal positions. The agent's movement is determined by the learned Q-values, leading to adaptive decision-making. This solution effectively addresses the problem of maze navigation by breaking it into exploration, learning, and path-finding.

## Final Output

After training, the agent successfully navigates from the start to the goal. It avoids walls and takes the shortest path as it learns to optimize its movements. The visualization of the learned path shows the agent's efficiency after thousands of iterations.




## Improvements in Clarity, Performance, and Flexibility

- **Grid Visibility:** Disabled grid lines for a cleaner visualization of the maze.

```
plt.grid(False)
```

- **Custom Maze Colors:** Different colors for walls, open spaces, and paths improve visual clarity.

```
from matplotlib.colors import ListedColormap
```

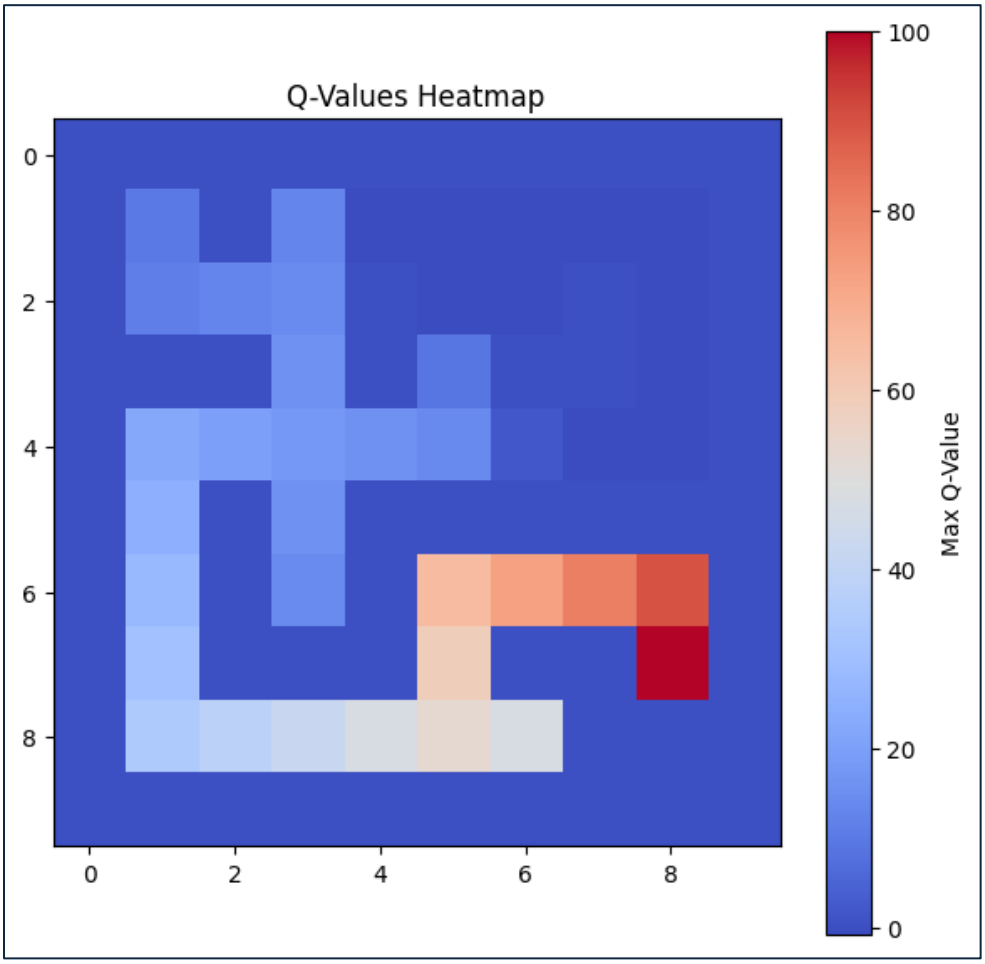
```
colors = ["white", "black", "yellow"]
cmap = ListedColormap(colors)
```

```
# Displaying the maze with the path, start, and goal points
plt.figure(figsize=(7,7))
plt.imshow(maze_copy, cmap=cmap, interpolation="nearest")
```

- **Dynamic Goal and Start Points:** This adds flexibility, making it easier to change the maze configuration without altering core logic.

```
# Function to get the learned path based on the Q-table
def get_learned_path(q_table, maze, start=(1,1), goal=(8,8)):
    state = start
    path = [state]
```

- **Visualization of Q-values:** A heatmap of the Q-values helps in understanding how the agent behaves in different states, providing deeper insight into the learning process.




The heatmap reveals how the agent perceives the environment, showing which states it considers more valuable based on the learned Q-values. Higher Q-values indicate states where the agent expects greater rewards, guiding its decision-making. The agent recognizes that those states lead to the most valuable outcome (reaching the goal).