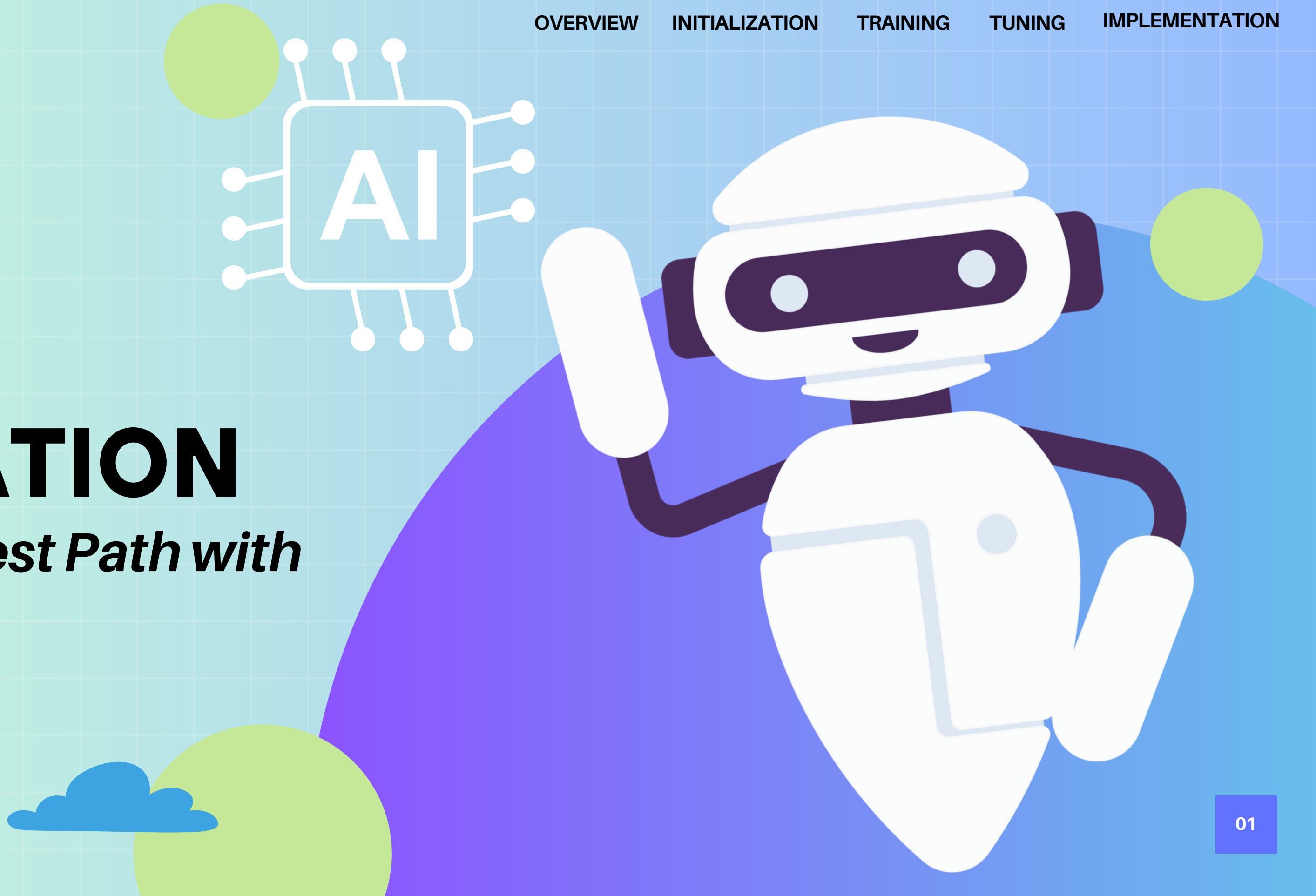
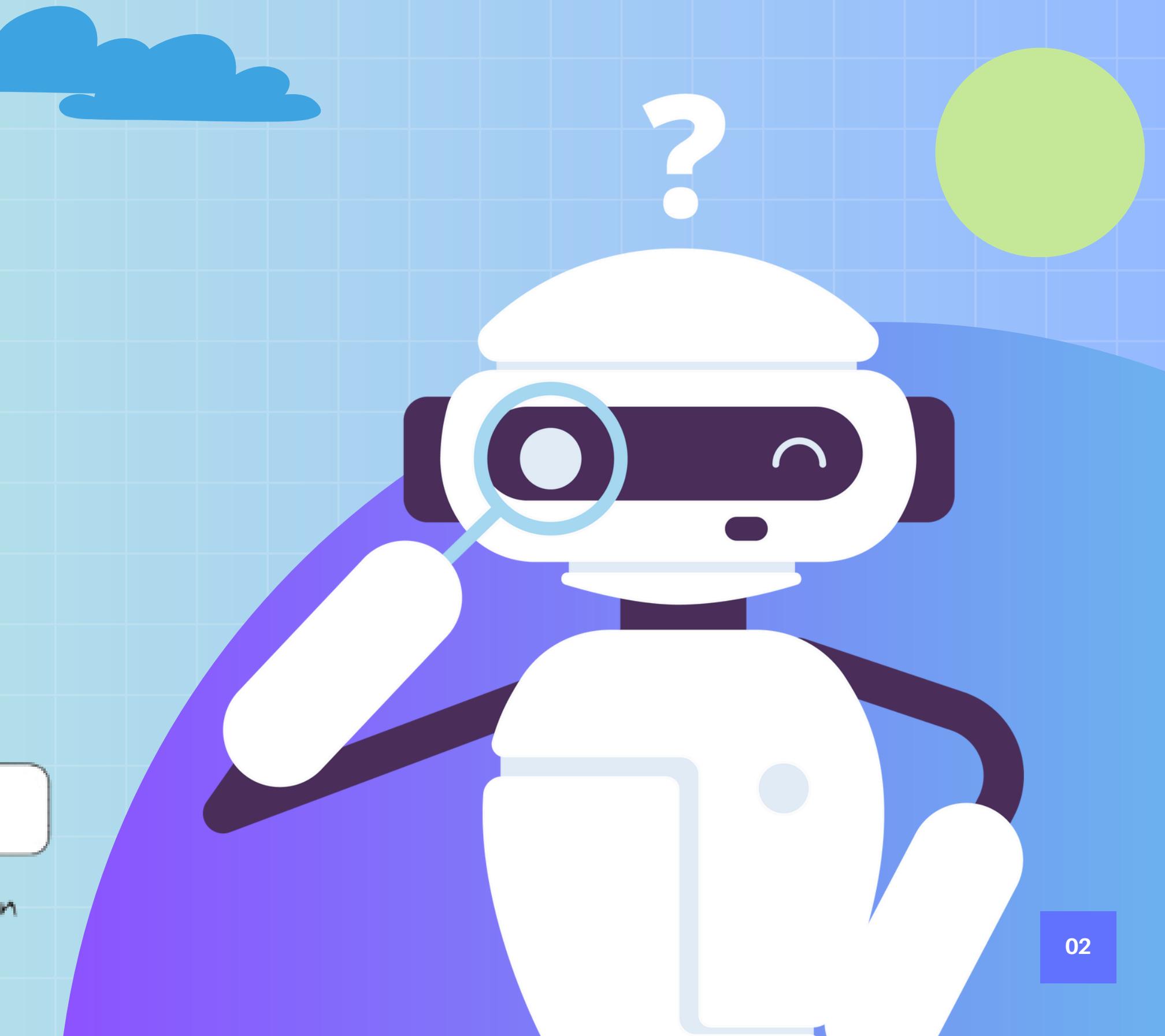
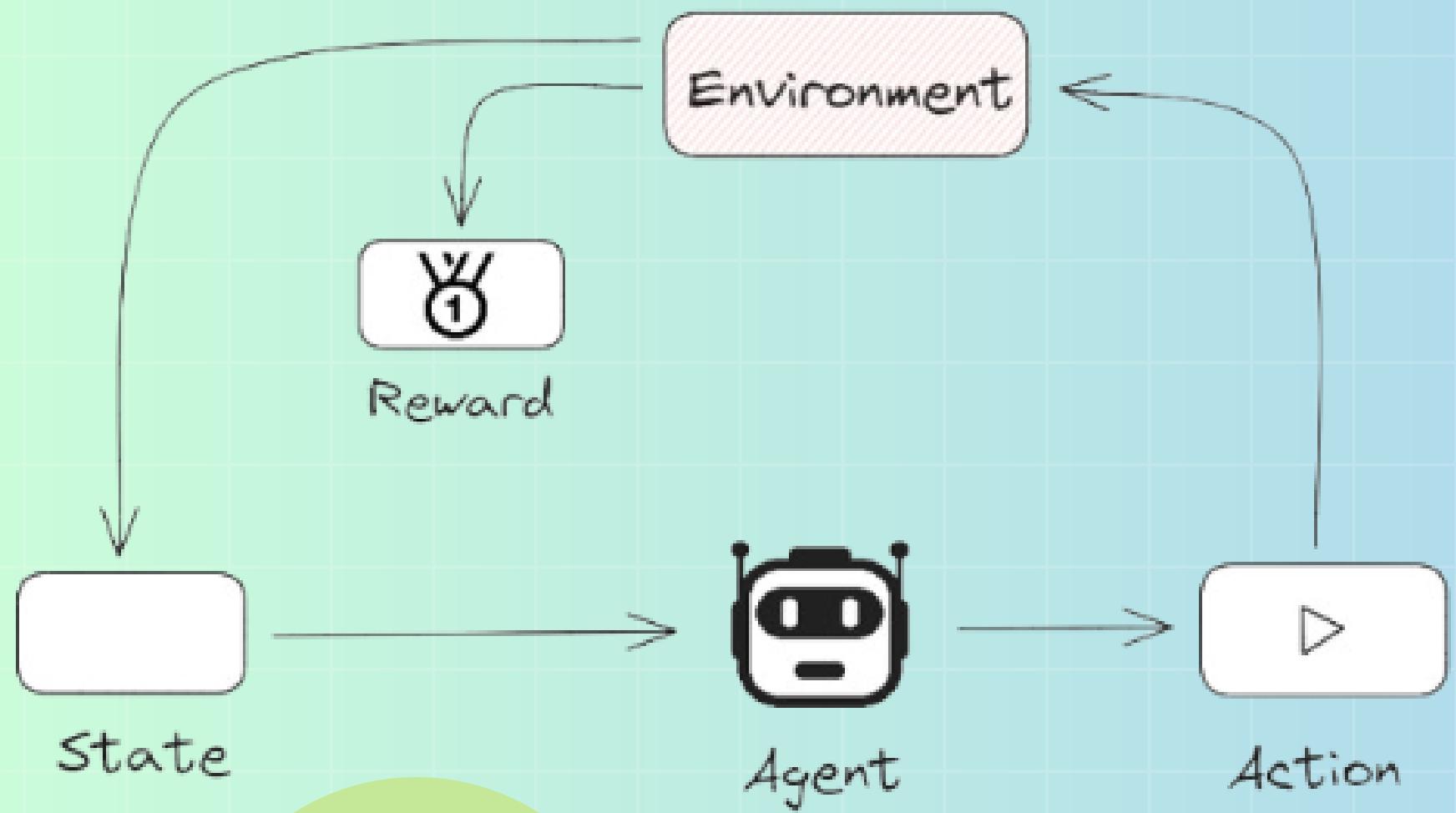


SMART NAVIGATION

*Finding the Best Path with
Q-Learning*

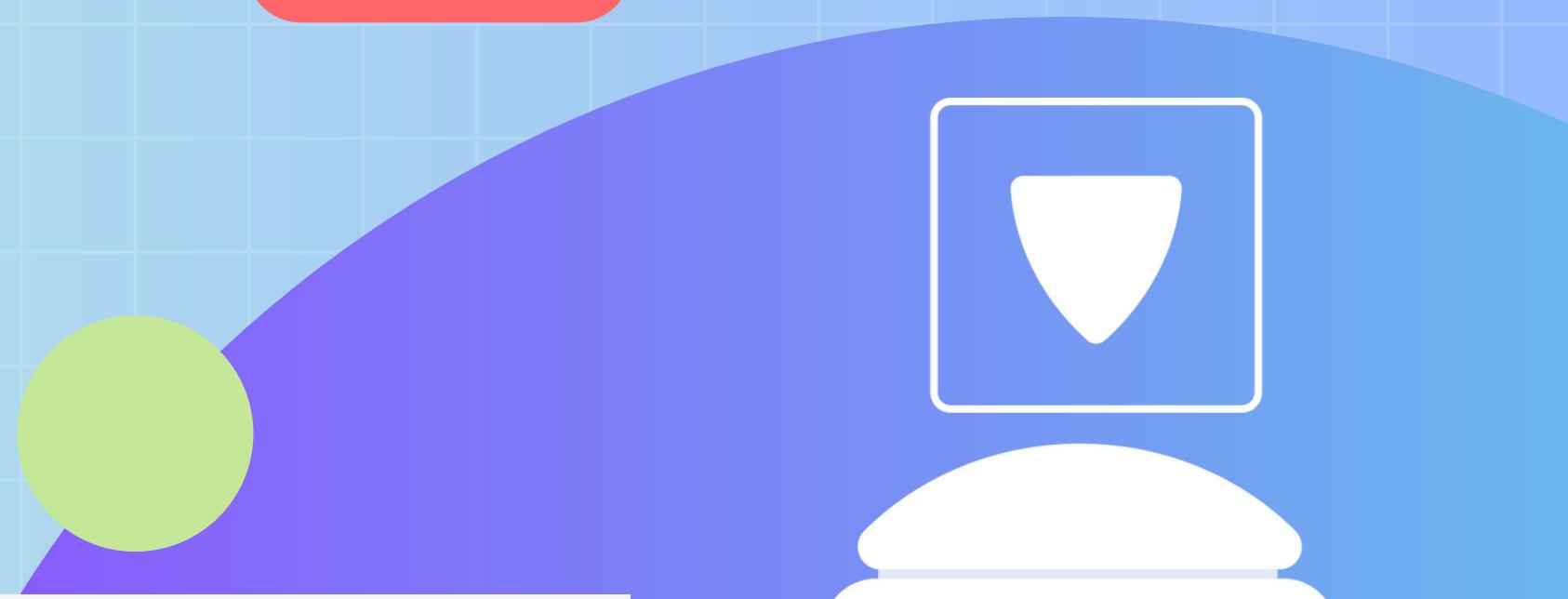


WHY Q-LEARNING ON NAVIGATION?

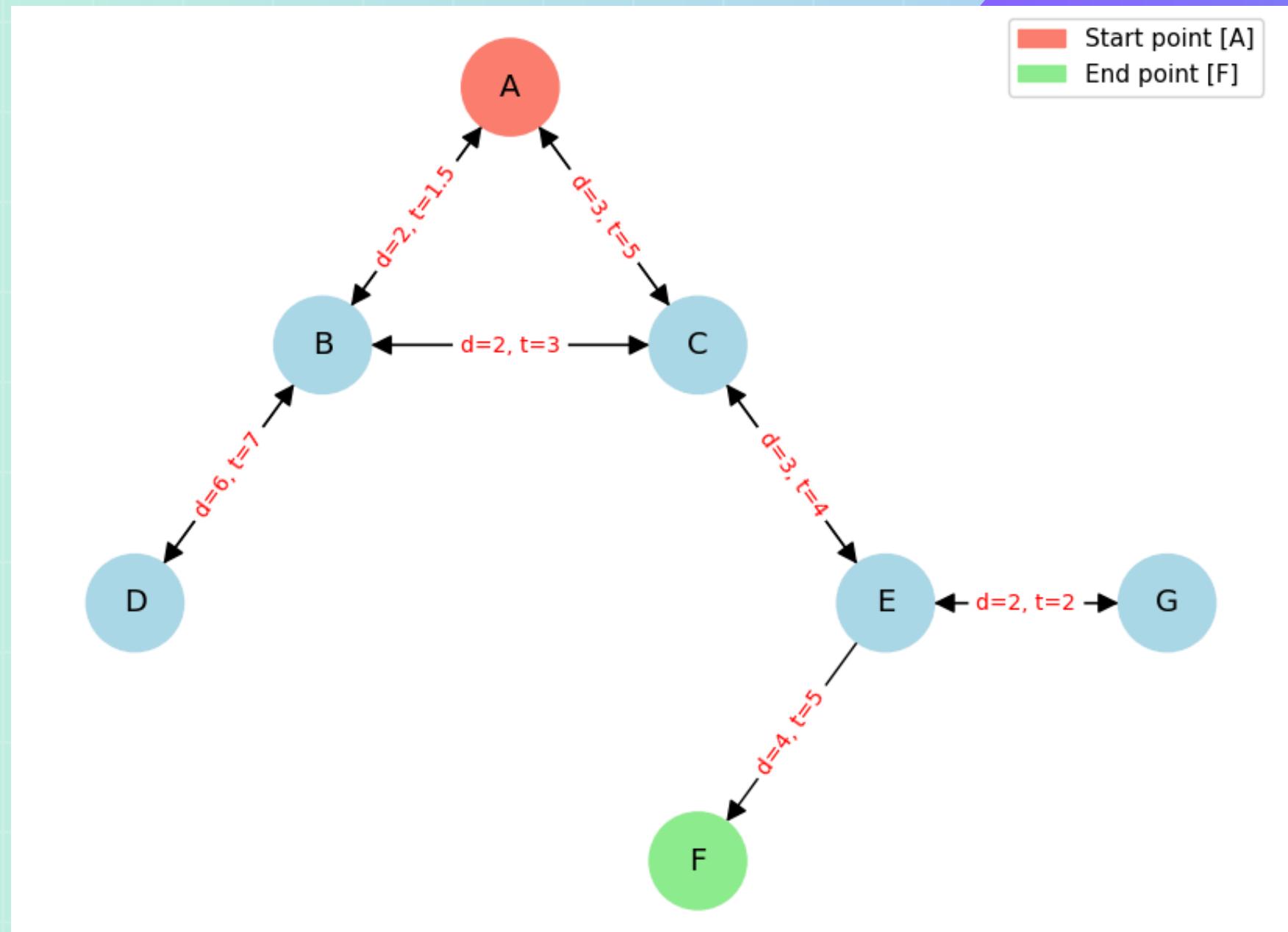


DEFINING MAP as Graph

```
# Define the map as a graph with distances and travel times
graph = {
    'A': {'B': {'distance': 2, 'time': 1.5}, 'C': {'distance': 3, 'time': 5}},
    'B': {'A': {'distance': 2, 'time': 1.5}, 'C': {'distance': 2, 'time': 3}, 'D': {'distance': 6, 'time': 7}},
    'C': {'A': {'distance': 3, 'time': 5}, 'B': {'distance': 2, 'time': 3}, 'E': {'distance': 3, 'time': 4}},
    'D': {'B': {'distance': 6, 'time': 7}},
    'E': {'C': {'distance': 3, 'time': 4}, 'F': {'distance': 4, 'time': 5}, 'G': {'distance': 2, 'time': 2}},
    'F': {}, # Destination, no need outgoing paths
    'G': {'E': {'distance': 2, 'time': 2}}
}
```



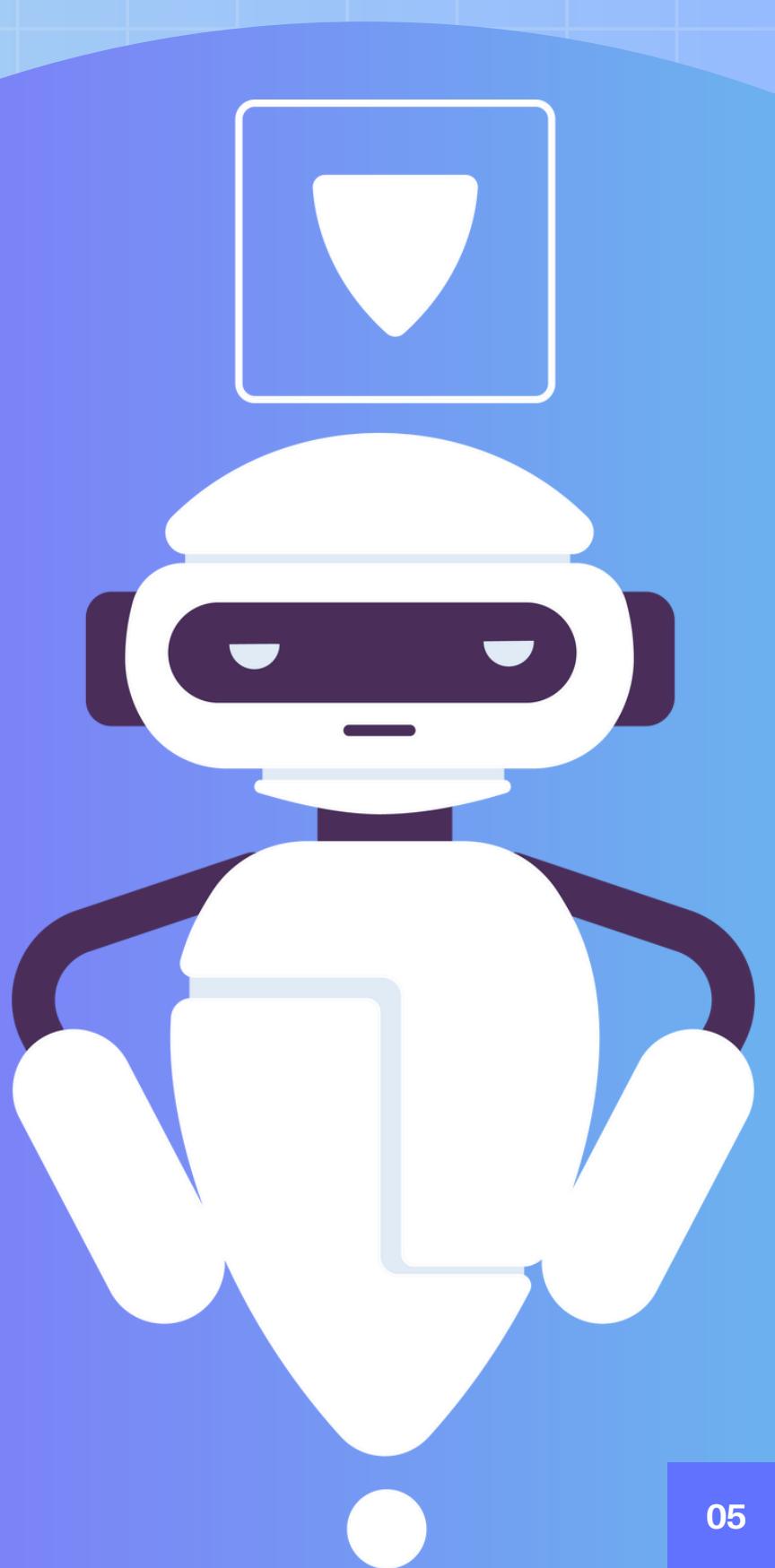
VISUALIZE GRAPH



INITIALIZE PARAMETER & HYPERPARAMETERS

```
# Parameters
q_table = np.zeros((len(states), len(actions))) # Q-table initialized with zeros

# Initialize Hyperparameters
learning_rate = 0.3 # Alpha
discount_factor = 0.8 # Gamma
epsilon = 0.3 # Exploration rate
episodes = 100 # Number of episodes
traffic_factor = 1.5 # Simulate traffic impact (1 = normal, >1 = heavy traffic)
```

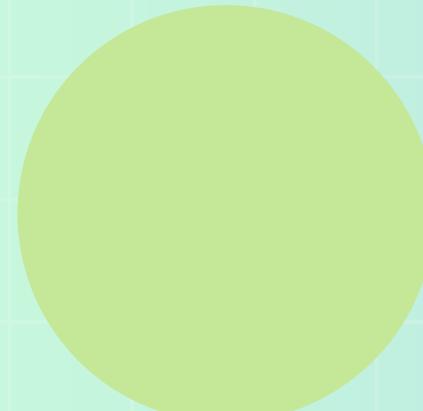
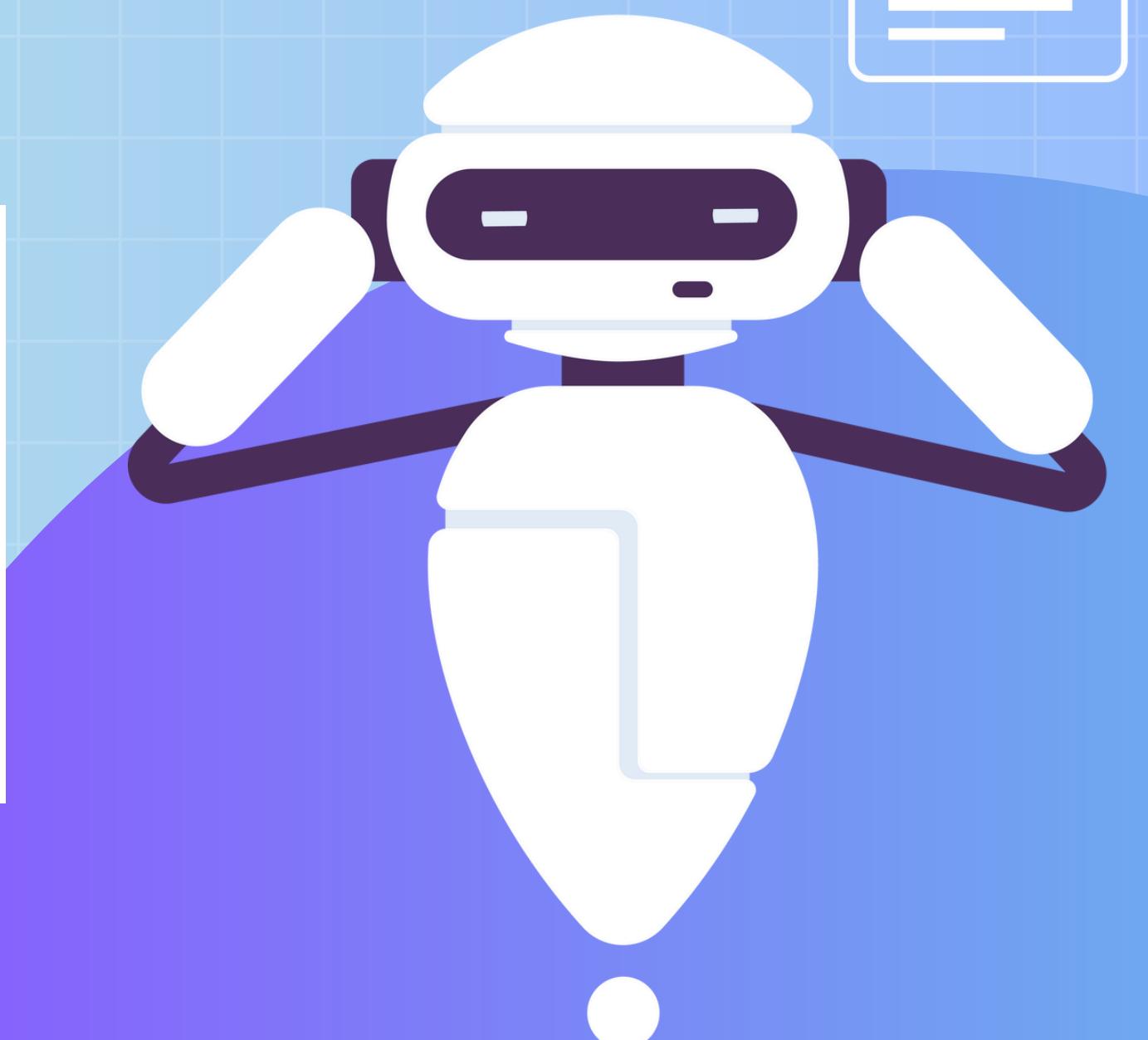
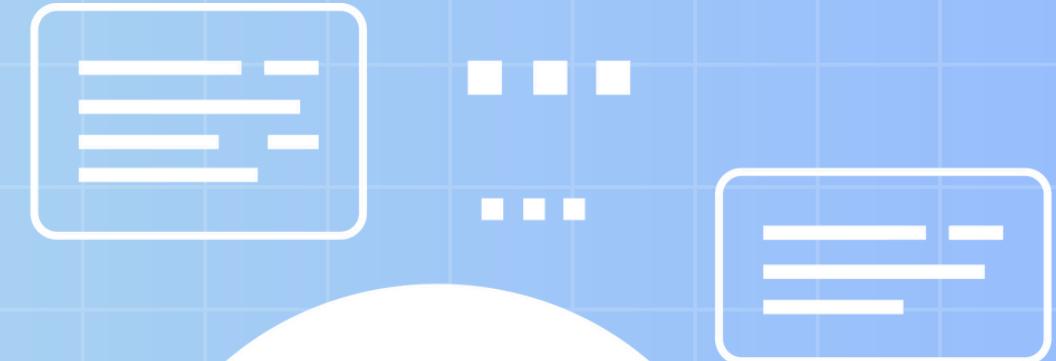




Q-TABLE

Learned Q-Table:

```
[[ 0.      -2.056   0.24    0.      0.      0.      0.      0.      ]  
 [-2.418    0.      -2.118  -1.469   0.      0.      0.      0.      ]  
 [-1.471  -2.464    0.      0.      0.181   0.      0.      0.      ]  
 [ 0.      -2.516   0.      0.      0.      0.      0.      0.      ]  
 [ 0.      0.      -1.459   0.      0.      0.087   0.      0.      ]  
 [ 0.      0.      0.      0.      0.      0.      0.      0.      ]  
 [ 0.      0.      0.      0.      0.      0.      0.      0.      ]]
```

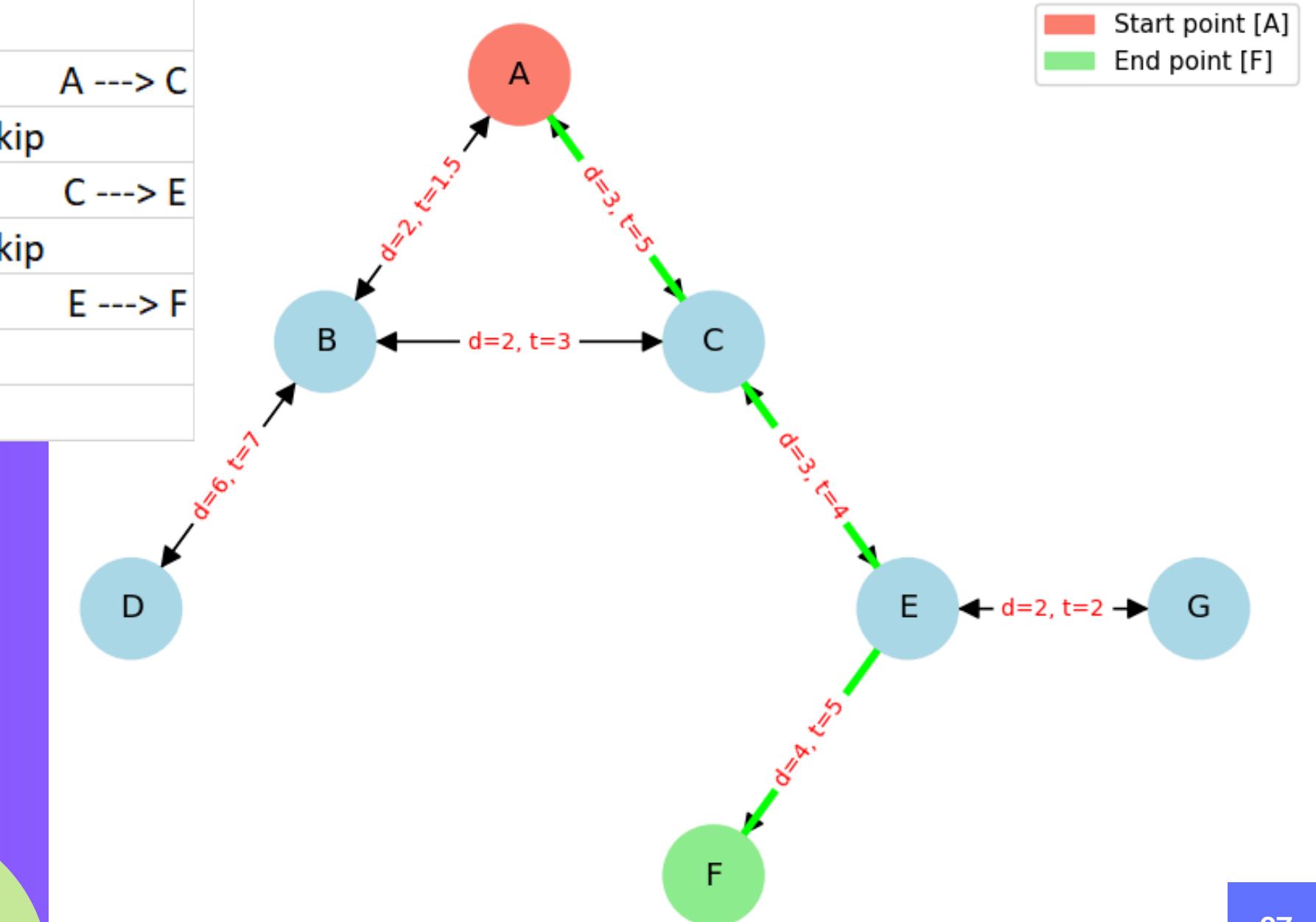


VISUALIZATION

	A	B	C	D	E	F	G	
A	0	-2,056	0,240	0	0	0	0	A ---> C
B	-2,418	0	-2,118	-1,469	0	0	0	skip
C	-1,471	-2,464	0	0	0,181	0	0	C ---> E
D	0	-2,516	0	0	0	0	0	skip
E	0	0	-1,459	0	0	0,087	0	E ---> F
F	0	0	0	0	0	0	0	
G	0	0	0	0	0	0	0	

[+] New route found:
A -> C -> E -> F
Total Distance: 10, Total Time: 21.0

Graph Visualization



FINDING POSSIBLE PATHS



```
# Test the function
multi_paths = find_multiple_paths('A', 'F', num_trials=10)

[-] Path did not reach destination: A -> C -> E -> G ----- (1)
[-] Path did not reach destination: A -> C -> E -> G ----- (2)
[-] Path did not reach destination: A -> B -> D ----- (3)

[+] New route found:
  A -> C -> E -> F
Total Distance: 10, Total Time: 21.0 ----- (4)

[=] Route already explored.
  A -> C -> E -> F ----- (5)

[=] Route already explored.
  A -> C -> E -> F ----- (6)

[=] Route already explored.
  A -> C -> E -> F ----- (7)

[=] Route already explored.
  A -> C -> E -> F ----- (8)

[=] Route already explored.
  A -> C -> E -> F ----- (9)

[-] Path did not reach destination: A -> B -> D ----- (10)
```

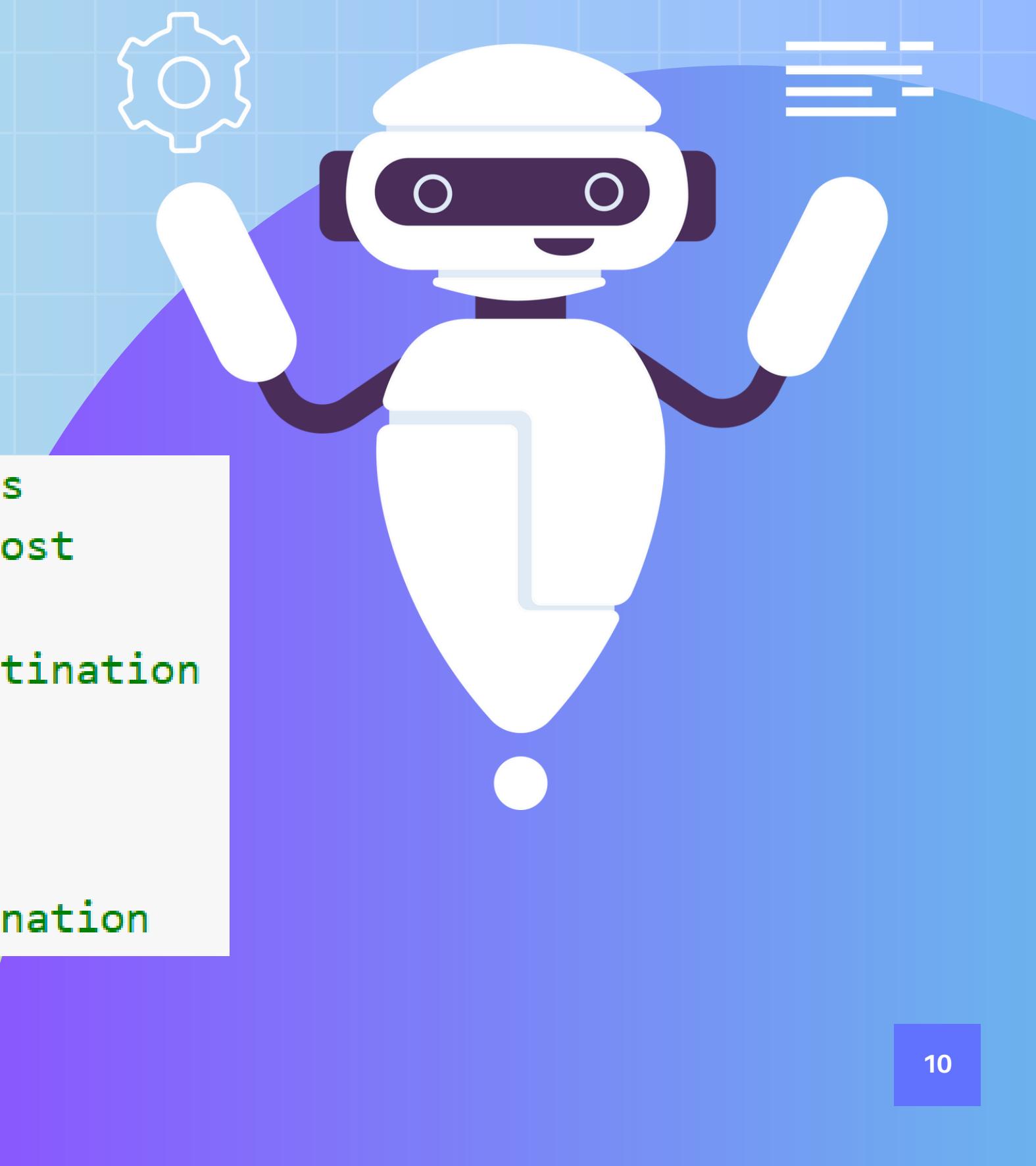
UPDATE HYPERPARAMETERS

```
q_table = np.zeros((len(states), len(actions))) # Reset Q-table with zeros

# Updated Hyperparameters
learning_rate = 0.1 # Alpha; previous = 0.3
discount_factor = 0.9 # Gamma; previous = 0.8
epsilon = 0.3 # Exploration rate; same as previous
episodes = 100 # Number of episodes; same as previous
traffic_factor = 1.5 # Simulate traffic impact; same as previous
```

UPDATE RULES & REWARDS

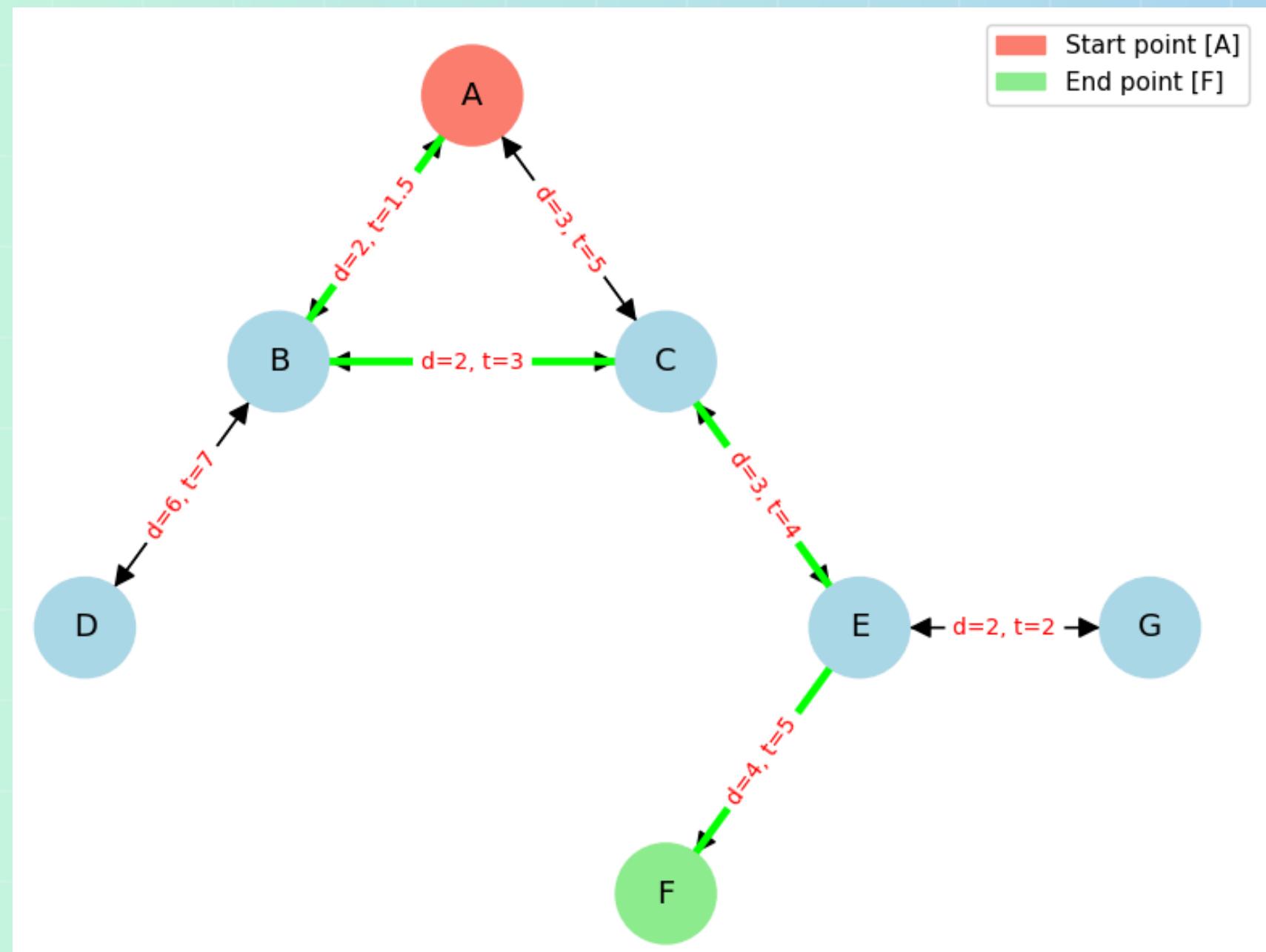
```
# Reward: Positive rewards with penalties for revisits
reward = 1 / (distance + time + 1e-7) # Inverse of cost
if next_state == 'F':
    reward += 15 # Bonus reward for reaching the destination
elif next_state in visited_states:
    reward -= 5 # Additional penalty for revisiting
else:
    reward -= 1 # Penalty for not reaching the destination
```



UPDATED Q-TABLE

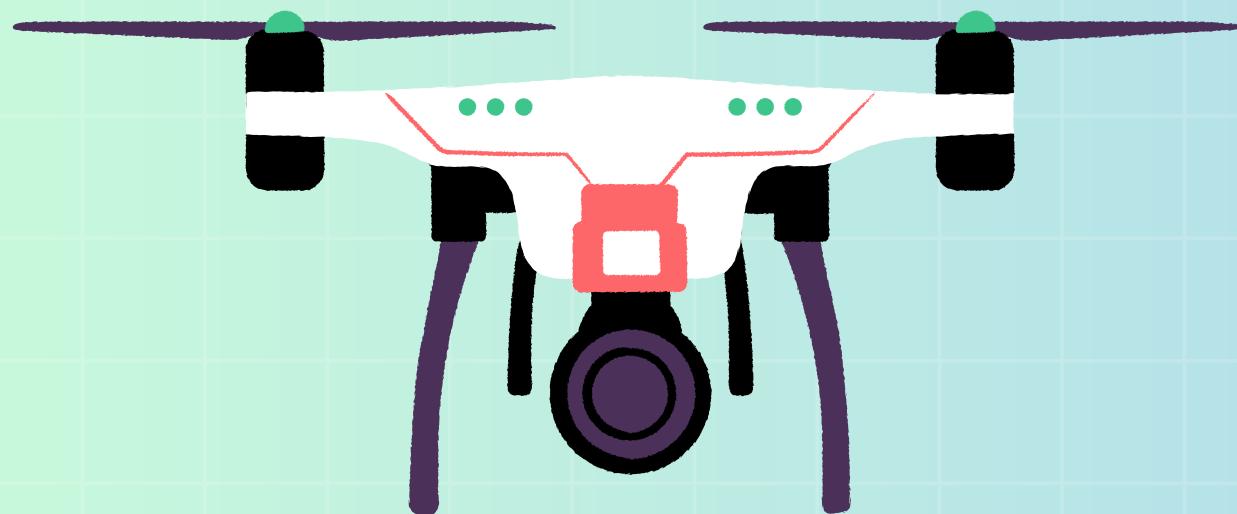
	A	B	C	D	E	F	G	
A	0	8,667	-1,005	0	0	0	0	A ---> B
B	-1,291	0	10,551	-1,047	0	0	0	B ---> C
C	-0,932	-0,921	0	0	12,685	0	0	C ---> E
D	0	-1,699	0	0	0	0	0	skip
E	0	0	-0,489	0	0	15,087	0	E ---> F
F	0	0	0	0	0	0	0	
G	0	0	0	0	0	0	0	

PATH VISUALIZATION



[+] New route found:
A -> B -> C -> E -> F
Total Distance: 11, Total Time: 20.25

FINDING POSSIBLE PATHS



```
# Test the function
multi_paths = find_multiple_paths('A', 'F', num_trials=10)
```

```
[+] New route found:
  A -> B -> C -> E -> F
  Total Distance: 11, Total Time: 20.25
----- (1)

[-] Path did not reach destination: A -> B -> C -> E -> G
----- (2)

[+] New route found:
  A -> C -> E -> F
  Total Distance: 10, Total Time: 21.0
----- (3)

[=] Route already explored.
  A -> B -> C -> E -> F
----- (4)

[=] Route already explored.
  A -> B -> C -> E -> F
----- (5)

[=] Route already explored.
  A -> B -> C -> E -> F
----- (6)

[=] Route already explored.
  A -> C -> E -> F
----- (7)

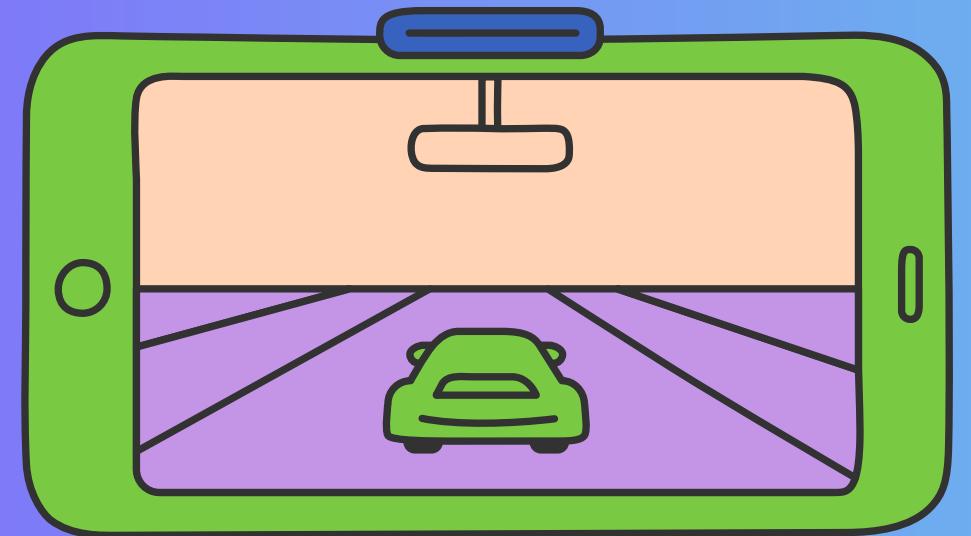
[-] Path did not reach destination: A -> B -> C -> A
----- (8)

[=] Route already explored.
  A -> B -> C -> E -> F
----- (9)

[=] Route already explored.
  A -> B -> C -> E -> F
----- (10)
```

DEFINE NEW MAP

```
# Example usage:  
graph = {  
    'A': {'B': {'distance': 2, 'time': 1.5}, 'C': {'distance': 3, 'time': 5}},  
    'B': {'A': {'distance': 2, 'time': 1.5}, 'C': {'distance': 2, 'time': 3}, 'D': {'distance': 6, 'time': 7}},  
    'C': {'A': {'distance': 3, 'time': 5}, 'B': {'distance': 2, 'time': 3}, 'E': {'distance': 3, 'time': 4}},  
    'D': {'B': {'distance': 6, 'time': 7}},  
    'E': {'C': {'distance': 3, 'time': 4}, 'F': {'distance': 4, 'time': 5}, 'G': {'distance': 2, 'time': 2}},  
    'F': {'E': {'distance': 4, 'time': 5}},  
    'G': {'E': {'distance': 2, 'time': 2}},  
}
```



CREATE OBJECT FROM CLASS

```
# Initialize PathFinder with custom start and end points
path_finder = PathFinder(graph=graph, start_point='D', end_point='F')
```



CREATE OBJECT FROM CLASS

```
# Initialize PathFinder with custom start and end points
path_finder = PathFinder(graph=graph, start_point='D', end_point='F')
```

TRAIN THE AGENT

```
# Train the agent
path_finder.train()

Agent training Done!
```



SHOW THE Q-TABLE

```
# Access the Q-table (after training)
q_table = path_finder.get_q_table()
```

Q-table:

```
[[ 0.   -1.291 -0.963  0.    0.    0.    0.    ],
 [-1.005  0.    10.551 -1.339  0.    0.    0.    ],
 [-0.932 -0.921  0.    0.    12.685  0.    0.    ],
 [ 0.    8.492  0.    0.    0.    0.    0.    ],
 [ 0.    0.    -0.489  0.    0.    15.087  0.    ],
 [ 0.    0.    0.    0.    0.    0.    0.    ],
 [ 0.    0.    0.    0.    0.    0.    0.    ]]
```

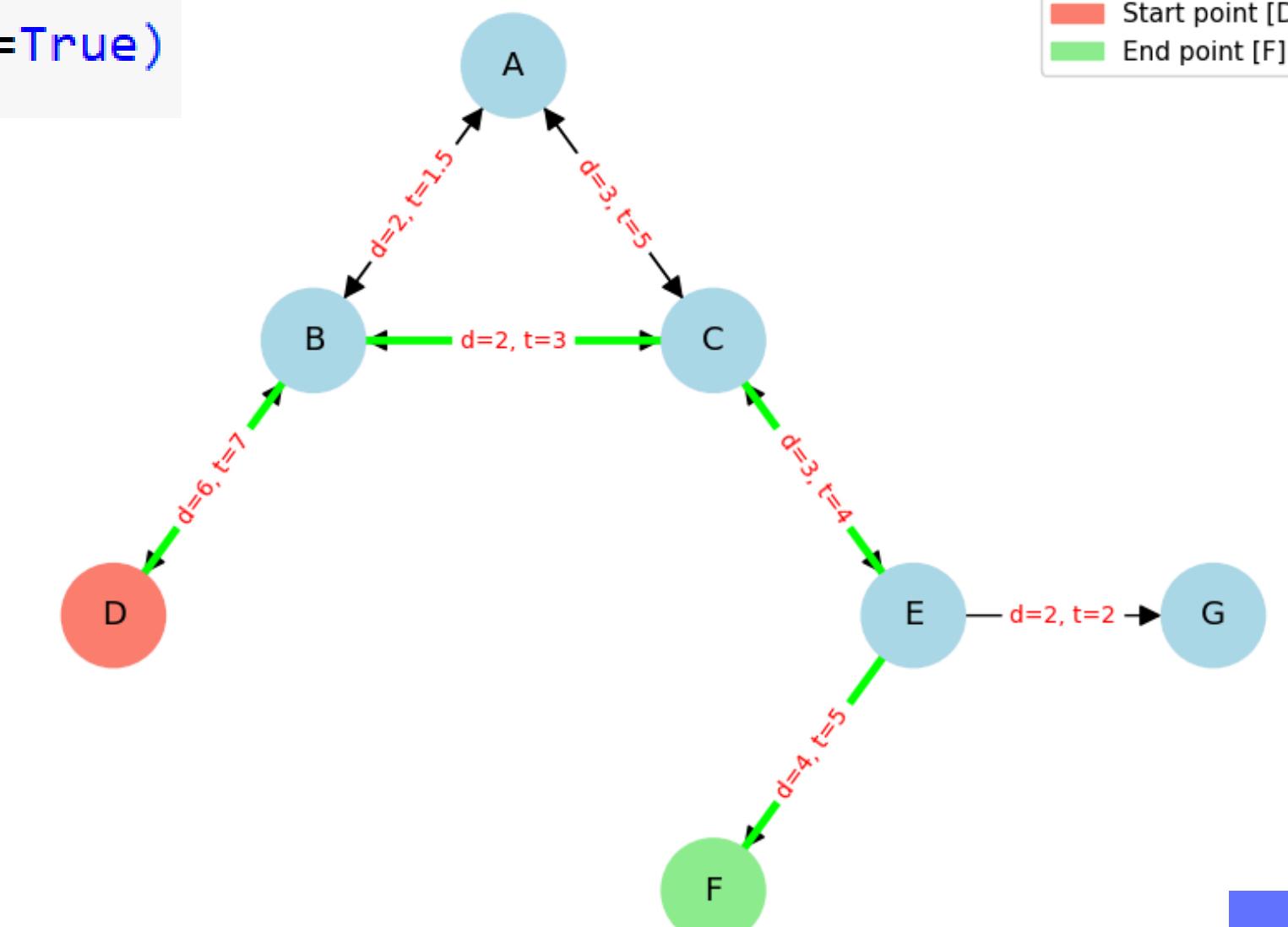


SHOW THE OPTIMAL PATH

```
# Get the optimal path and visualize the path in graphic
optimal_path = path_finder.get_optimal_path(show_graph=True)
```

Optimal Path: D -> B -> C -> E -> F
Total Distance: 15, Total Time: 28.5

Graph Visualization





FIND MULTIPLE PATHS

```
# Find multiple paths
multiple_paths = path_finder.find_multiple_paths(num_trials=10)

[-] Path did not reach destination: D -> B -> D
----- (1)
No valid actions from state G.
[-] Path did not reach destination: D -> B -> C -> E -> G
----- (2)

[+] New route found:
  D -> B -> C -> E -> F
Total Distance: 15, Total Time: 28.5
----- (3)

[=] Route already explored.
  D -> B -> C -> E -> F
----- (4)

[=] Route already explored.
  D -> B -> C -> E -> F
----- (5)

[=] Route already explored.
  D -> B -> C -> E -> F
----- (6)

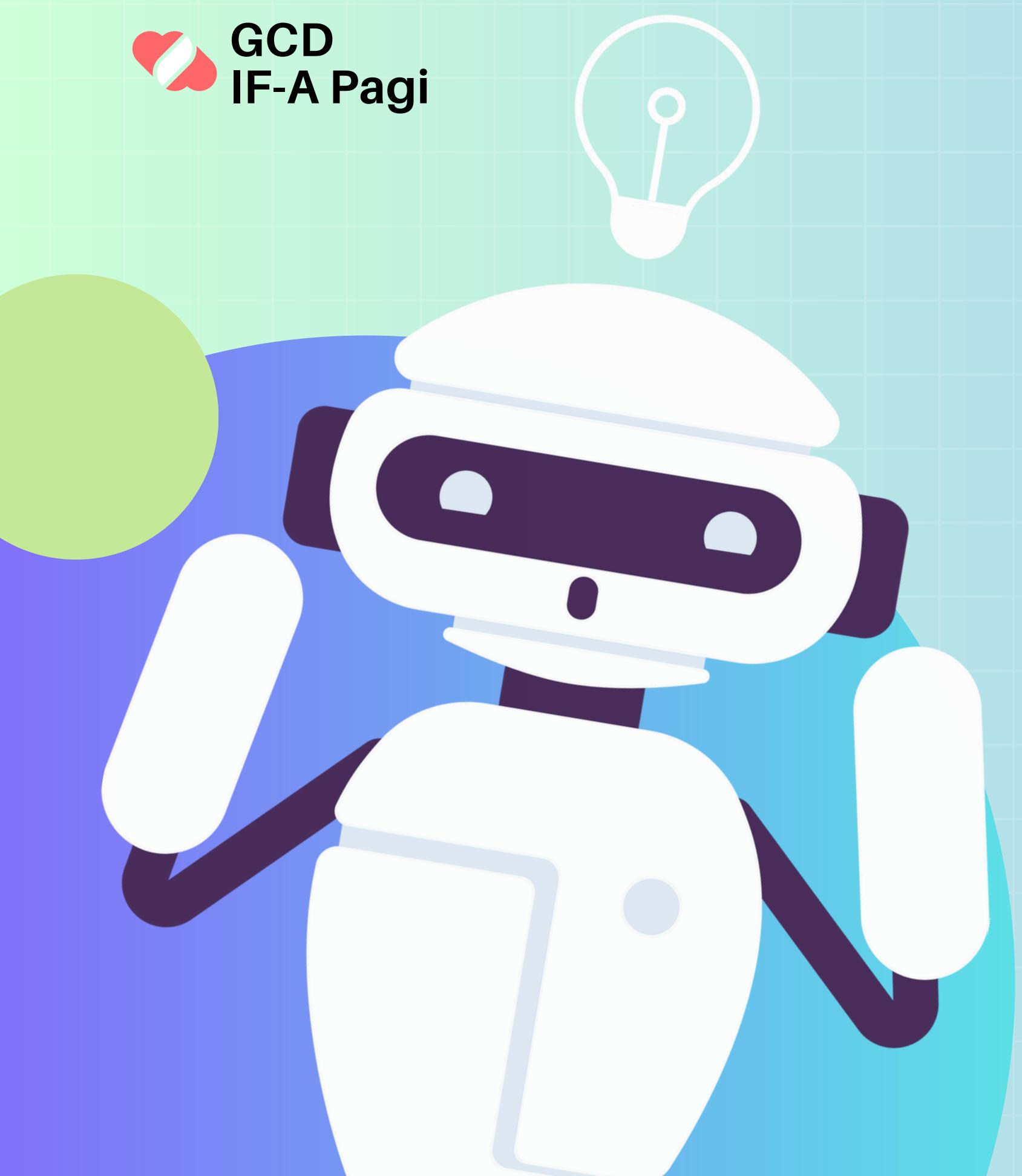
[=] Route already explored.
  D -> B -> C -> E -> F
----- (7)

[=] Route already explored.
  D -> B -> C -> E -> F
----- (8)

[+] New route found:
  D -> B -> A -> C -> E -> F
Total Distance: 18, Total Time: 33.75
----- (9)

[=] Route already explored.
  D -> B -> C -> E -> F
----- (10)
```





CONCLUSION & RECOMMENDATION

