ai-lab-03

February 16, 2023

1 Question 01

[6]:
```
import numpy as np
import math
array = np.array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                  ['A', 0, 0, 0, 0, 0, 0, 0, 0, 0],
                  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                  [0, 0, 0, 0, 'B', 0, 0, 0, 0, 0],
                  [0, 0, 'R', 0, 0, 0, 0, 0,'C', 0],
                  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                  [0, 0, 0, 0, 0, 0, 0, 'D', 0, 0],
                  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                  [0, 0, 0, 0, 0, 0, 0, 'E', 0, 0]])

locations = ['A', 'B', 'C', 'D', 'E']
x_R, y_R = np.where(array == 'R')
location_R = (x_R, y_R)

for loc in locations:
  x, y = np.where(array == loc)
  location = (x, y)
  dist = math.dist(location_R, location)
  print('Distance of',loc, 'from R:',str(dist))
```

```
Distance of A from R: 3.6055512754639896
Distance of B from R: 2.23606797749979
Distance of C from R: 6.0
Distance of D from R: 5.385164807134505
Distance of E from R: 7.0710678118654755
```

2 Question 02

[8]:
```
def driving(front,rear,right,left):
    if front <= 8:
        print("Apply Breaks")
    elif left <= 2:
```

```
        print("Move to Right Lane")
    elif right <= 2:
        print("Move to Left Lane")
    elif rear <= 0.5:
        print("Apply break there is an obstacle at the back")
    else:
        print("Drive")


front = float(input("Enter front reading: "))
rear = float(input("Enter rear reading: "))
right = float(input("Enter right reading: "))
left = float(input("Enter left reading: "))
driving(front,rear,right,left)
```

```
Enter front reading: 2
Enter rear reading: 3
Enter right reading: 4
Enter left reading: 5
Apply Breaks
```

3 Question 03

[11]:
```
temperature = []

for i in range (0,9):
    temp = float(input("Enter temperature :"))
    temperature.append(temp)

total = sum(temperature)
avg = total / 9

avg_F = (avg * 9/5) +32
print("Avg in Farenhite is ", avg)
```

```
Enter temperature :1
Enter temperature :2
Enter temperature :3
Enter temperature :4
Enter temperature :5
Enter temperature :5
Enter temperature :6
Enter temperature :8
Enter temperature :6
Avg in Farenhite is  4.444444444444445
```

4 Question 04

[13]:
```
import numpy as np


def init_room(n, m, p_dirt, p_blocked):
    room = np.random.choice(['C', 'B', 'D'], size=(n, m),␣
 ↪p=[1-p_blocked-p_dirt, p_blocked, p_dirt])
    return room


def can_move(pos, direction, room_shape):
    x, y = pos
    if direction == 'up' and x > 0:
        return True
    elif direction == 'down' and x < room_shape[0] - 1:
        return True
    elif direction == 'left' and y > 0:
        return True
    elif direction == 'right' and y < room_shape[1] - 1:
        return True
    else:
        return False

def move(pos, direction):
    x, y = pos
    if direction == 'up':
        return (x-1, y)
    elif direction == 'down':
        return (x+1, y)
    elif direction == 'left':
        return (x, y-1)
    elif direction == 'right':
        return (x, y+1)


def clean_cell(pos, room):
    room[pos] = 'C'
    return room


def is_clean(room):
    return np.all(room == 'C')


def run_vacuum(room, start_pos):
    path = [start_pos]
```

```
    current_pos = start_pos
    while not is_clean(room):
        for direction in ['up', 'down', 'left', 'right']:
            if can_move(current_pos, direction, room.shape):
                next_pos = move(current_pos, direction)
                if room[next_pos] == 'D':
                    room = clean_cell(next_pos, room)
                path.append(next_pos)
                current_pos = next_pos
                break
    return path

def display_path(room, path):
    for i in range(room.shape[0]):
        for j in range(room.shape[1]):
            if (i, j) in path:
                print('V', end=' ')
            else:
                print(room[i][j], end=' ')
        print()


room = init_room(5, 7, 0.2, 0.1)
start_pos = (0, 0)
path = run_vacuum(room, start_pos)
display_path(room, path)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8820\3288383479.py in <module>
     67 room = init_room(5, 7, 0.2, 0.1)
     68 start_pos = (0, 0)
---> 69 path = run_vacuum(room, start_pos)
     70 display_path(room, path)

~\AppData\Local\Temp\ipykernel_8820\3288383479.py in run_vacuum(room, start_pos
     44     path = [start_pos]
     45     current_pos = start_pos
---> 46     while not is_clean(room):
     47         for direction in ['up', 'down', 'left', 'right']:
     48             if can_move(current_pos, direction, room.shape):

~\AppData\Local\Temp\ipykernel_8820\3288383479.py in is_clean(room)
     38
     39 def is_clean(room):
---> 40     return np.all(room == 'C')
     41
```

```
     42

<__array_function__ internals> in all(*args, **kwargs)

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py in all(a,␣
 ↪axis, out, keepdims, where)
   2448
   2449     """
-> 2450     return _wrapreduction(a, np.logical_and, 'all', axis, None, out,
   2451                           keepdims=keepdims, where=where)
   2452

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py in␣
 ↪_wrapreduction(obj, ufunc, method, axis, dtype, out, **kwargs)
     68
     69 def _wrapreduction(obj, ufunc, method, axis, dtype, out, **kwargs):
---> 70     passkwargs = {k: v for k, v in kwargs.items()
     71                   if v is not np._NoValue}
     72

KeyboardInterrupt:
```

5 Question 05

[14]:
```
import random

def draw_board(board):
    print('   |   |')
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
    print('   |   |')

def input_player_letter():
    letter = ''
    while not (letter == 'X' or letter == 'O'):
        print('Do you want to be X or O?')
        letter = input().upper()
    if letter == 'X':
```

```
        return ['X', 'O']
    else:
        return ['O', 'X']

def who_goes_first():
    if random.randint(0, 1) == 0:
        return 'computer'
    else:
        return 'player'

def play_again():
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')

def make_move(board, letter, move):
    board[move] = letter

def is_winner(b, l):
    return ((b[7] == l and b[8] == l and b[9] == l) or
    (b[4] == l and b[5] == l and b[6] == l) or
    (b[1] == l and b[2] == l and b[3] == l) or
    (b[7] == l and b[4] == l and b[1] == l) or
    (b[8] == l and b[5] == l and b[2] == l) or
    (b[9] == l and b[6] == l and b[3] == l) or
    (b[7] == l and b[5] == l and b[3] == l) or
    (b[9] == l and b[5] == l and b[1] == l))

def get_board_copy(board):
    dupe_board = []
    for i in board:
        dupe_board.append(i)
    return dupe_board

def is_space_free(board, move):
    return board[move] == ' '

def get_player_move(board):
    move = ' '
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not is_space_free(board,␣
 ↪int(move)):
        print('What is your next move? (1-9)')
        move = input()
    return int(move)

def choose_random_move(board, moves):
  possible_moves = []
  for i in moves:
```

```
    if is_space_free(board, i):
      possible_moves.append(i)
  if len(possible_moves) != 0:
    return random.choice(possible_moves)
  else:
    return None

def get_computer_move(board, letter):
  if letter == 'X':
    player_letter = 'O'
  else:
    player_letter = 'X'
  for i in range(1, 10):
    copy = get_board_copy(board)
    if is_space_free(copy, i):
      make_move(copy, letter, i)
      if is_winner(copy, letter):
        return i
  for i in range(1, 10):
    copy = get_board_copy(board)
    if is_space_free(copy, i):
      make_move(copy, player_letter, i)
      if is_winner(copy, player_letter):
        return i
  move = choose_random_move(board, [1, 3, 7, 9])
  if move != None:
    return move
  if is_space_free(board, 5):
    return 5
  return choose_random_move(board, [2, 4, 6, 8])

def is_board_full(board):
  for i in range(1, 10):
    if is_space_free(board, i):
      return False
  return True

print('Welcome to Tic Tac Toe!')
while True:
  the_board = [' '] * 10
  player_letter, computer_letter = input_player_letter()
  turn = who_goes_first()
  print('The ' + turn + ' will go first.')
  playing = True
  while playing:
    if turn == 'player':
      draw_board(the_board)
```

```
      move = get_player_move(the_board)
      make_move(the_board, player_letter, move)
      if is_winner(the_board, player_letter):
        draw_board(the_board)
        print('You have won the game!')
        playing = False
      else:
        if is_board_full(the_board):
          draw_board(the_board)
          print('The game is a tie!')
          break
        else:
          turn = 'computer'
    else:
      move = get_computer_move(the_board, computer_letter)
      make_move(the_board, computer_letter, move)
      if is_winner(the_board, computer_letter):
        draw_board(the_board)
        print('The computer has won. You lose.')
        playing = False
      else:
        if is_board_full(the_board):
          draw_board(the_board)
          print('The game is a tie!')
          break
        else:
          turn = 'player'
  if not play_again():
    break
```

```
Welcome to Tic Tac Toe!
Do you want to be X or O?
O
The player will go first.
   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |
What is your next move? (1-9)
5
```

```
   |   |
   |   |
   |   |
-----------
   |   |
   | O |
   |   |
-----------
   |   |
 X |   |
   |   |
What is your next move? (1-9)
0
What is your next move? (1-9)
7
   |   |
 O |   |
   |   |
-----------
   |   |
   | O |
   |   |
-----------
   |   |
 X |   | X
   |   |
What is your next move? (1-9)
2
   |   |
 O | X |
   |   |
-----------
   |   |
   | O |
   |   |
-----------
   |   |
 X | O | X
   |   |
What is your next move? (1-9)
4
   |   |
 O | X |
   |   |
-----------
   |   |
 O | O | X
   |   |
```

```
-----------
   |   |
 X | O | X
   |   |
What is your next move? (1-9)
9
   |   |
 O | X | O
   |   |
-----------
   |   |
 O | O | X
   |   |
-----------
   |   |
 X | O | X
   |   |
The game is a tie!
Do you want to play again? (yes or no)
no
```

6 Question 06

[15]:
```python
import heapq

def shortest_path(network, source, destination):
  distances = {node: float('inf') for node in network}
  distances[source] = 0
  priority_queue = [(0, source)]
  while priority_queue:
    (current_distance, current_node) = heapq.heappop(priority_queue)
    if current_distance > distances[current_node]:
      continue
    for neighbor, cost in network[current_node].items():
      distance = current_distance + cost
      if distance < distances[neighbor]:
        distances[neighbor] = distance
        heapq.heappush(priority_queue, (distance, neighbor))
  return distances[destination]


network = {
    '0': {'1': 4, '6': 7},
    '1': {'0': 4, '2': 9, '6': 11, '7': 20},
    '2': {'1': 9, '3': 6, '4': 2},
    '3': {'2': 6, '4': 10, '5': 5},
    '4': {'2': 2, '3': 10, '5': 15, '1': 1, '8': 5},
```

```
    '5': {'3': 5, '4': 15, '8': 12},
    '6': {'0': 7, '1': 11, '7': 1},
    '7': {'1': 20, '4': 1, '6': 1, '8': 3},
    '8': {'4': 5, '5': 12, '7': 3},
}

source = '0'
destination = '8'

min_distance = shortest_path(network, source, destination)

print("The shortest distance from",source,"to",destination,"is",min_distance)
```

```
The shortest distance from 0 to 8 is 11
```

7 Question 07

[17]:
```
print("Converting a task into a learning agent involves several steps. Firstly,␣
 ↪you should recognize the inputs, outputs, and the existing algorithms that␣
 ↪are being utilized to solve the task. Following this, you need to create a␣
 ↪machine learning model that can comprehend and learn from the inputs and␣
 ↪outputs and adapt its parameters accordingly. For instance, if you want to␣
 ↪transform a snake game into a learning agent, you will need to collect data␣
 ↪about the snake's movements and the obstacles in the game. This data can be␣
 ↪employed to train a neural network that forecasts the next movement of the␣
 ↪snake based on its current location and the presence of barriers. The model␣
 ↪can be constantly updated using the snake's actual movements in the game,␣
 ↪which will enable it to improve its accuracy over time by learning from its␣
 ↪errors.")
```

```
Converting a task into a learning agent involves several steps. Firstly, you
should recognize the inputs, outputs, and the existing algorithms that are being
utilized to solve the task. Following this, you need to create a machine
learning model that can comprehend and learn from the inputs and outputs and
adapt its parameters accordingly. For instance, if you want to transform a snake
game into a learning agent, you will need to collect data about the snake's
movements and the obstacles in the game. This data can be employed to train a
neural network that forecasts the next movement of the snake based on its
current location and the presence of barriers. The model can be constantly
updated using the snake's actual movements in the game, which will enable it to
improve its accuracy over time by learning from its errors.
```

[ ]:
```

```