Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

Task 1B

Detect and Encode Maze

[*Last Updated on:* ***19th October 2020, 16:39 Hrs***]

- 1. Understanding Mazes
  - Unique Encoding to represent Cells in a Maze
- 2. Implementation of Solution
  - Given
  - Problem Statement
  - Instructions
  - Running your solution
- 3. Testing the Solution
- Submission Instructions

The aim of this task is as follows:

- Detecting a maze from a given image using OpenCV techniques
- Encoding the maze as a matrix in a CSV file

This task is divided into **three** parts:

1. ***Understanding Mazes***
2. ***Implementation of Solution***
3. ***Testing the Solution***

> Make sure you go through the instructions thoroughly and in a sequential order. It will help you understand the tasks better.

Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

1. Understanding Mazes

In this section, we will first learn about what mazes are and how they are constructed.

- We start with a **Grid** first. Shown below in Figure 1 is a representation of a ***5 x 5*** grid where *firs number* represent *number of rows* and the *second number* shows *number of columns* in the grid

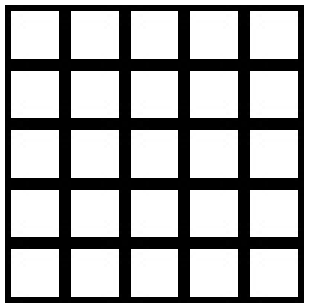

Figure 1: 5 x 5 Grid

- The smallest building element of a grid is defined as a **Cell**.
- A ***Cell*** is defined as a square with **four** sides where a ***Wall*** (represented by bold black lines around the cell) may or may not exist at each of the sides.
- A grid can be converted into a **Maze** by carving out passages through the grid. An example maze carved out of Figure 1 is shown in Figure 2.

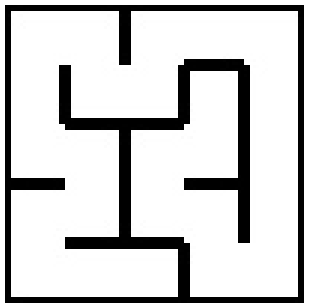

Figure 2: Example Maze from 5x5 Grid

- As we have converted grid of Figure 1 into a Maze in Figure 2, this maze also has dimensions o ***5 x 5***, hence there are ***25 cells*** in each of them.
- For example, consider the following cells in Figure 3.

Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

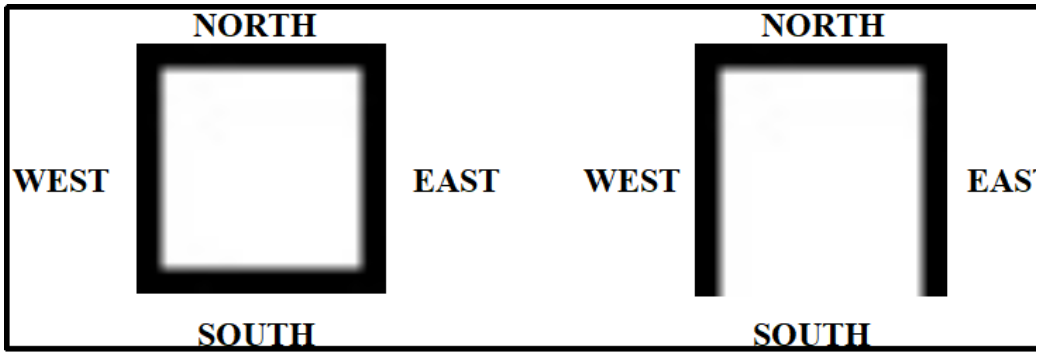

Figure 3: Two variants of Cells in a Maze

- The dark bands indicate presence of walls. **NORTH**, **SOUTH**, **EAST** and **WEST** are called **directions** in the cell.
- The first cell in Figure 3 has ***no open*** passages in either of the four directions of the cell while the second cell has an open passage in ***SOUTH*** and the passages to the other three directions are blocked by walls. Since each cell have four sides we can have a total of $4^2 = 16$ different ways in which a cell may be represented. Two of them are shown in Figure 3 and rest of the 14 configurations are shown in Figure 4.

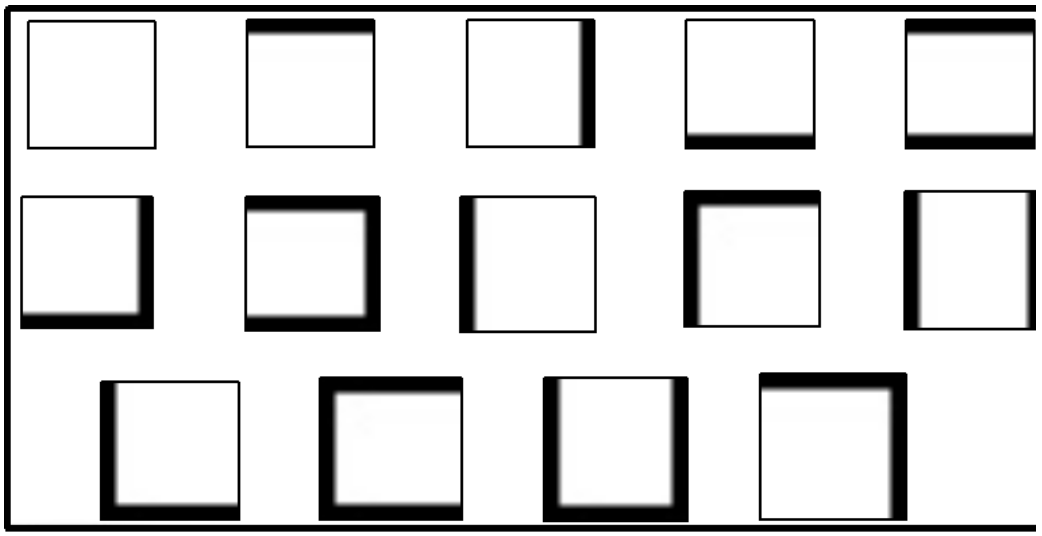

Figure 4: 14 different variants of a Cell in a Maze

- A maze as given in Figure 2, is nothing but a collection of different types of cells as given in Figure 4.

Unique Encoding to represent Cells in a Maze

Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

- As discussed above, there are a total of **16** different types of cells.
- Each cell can be represented using a unique ***Cell Number*** between **0** to **15**. This number is assigned on the basis of wall configuration of the cell.
- In Figure 5, we have assigned a binary weight to each of the directions of the cell. The ***cell number*** for each type of cell can be calculated by adding the weights for the directions in whic a wall exists and ignoring the weights of the directions where wall doesn't exist.
- For example, the ***cell number*** for cell in Figure 5 is **Weight of North wall (W_N)** + **Weight of West wall (W_W)** = **2 + 1** = **3**.

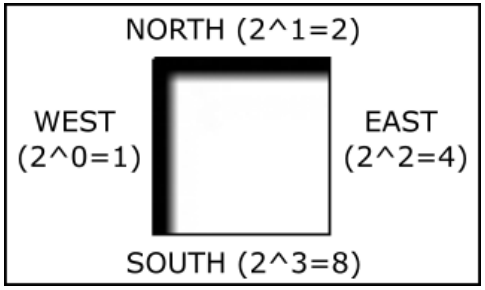

Figure 5: Cell Number calculation based on weights

- Similarly the ***cell numbers*** for the rest of the cell types can be calculated as given in Figure 6.

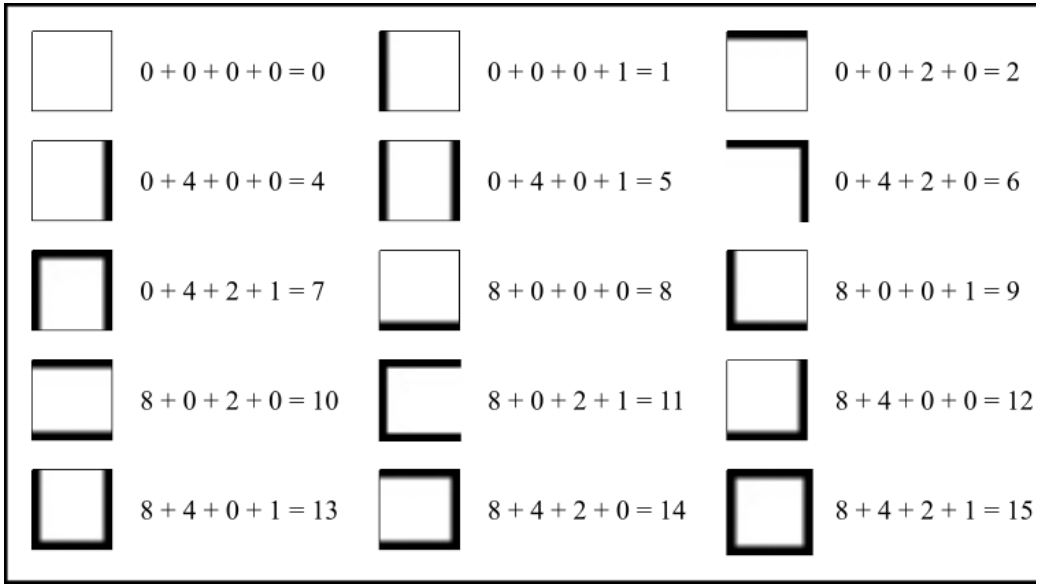

Figure 6: Cell Numbers for different types of cells

Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

- You will have noticed that the ***cell numbers*** are completely unambiguous and unique.
- Cell numbers are significant because these allow us to represent a maze as a ***2D array*** in whic each array element represents a cell of a maze by its corresponding ***cell number*** (as shown in Figure 7).

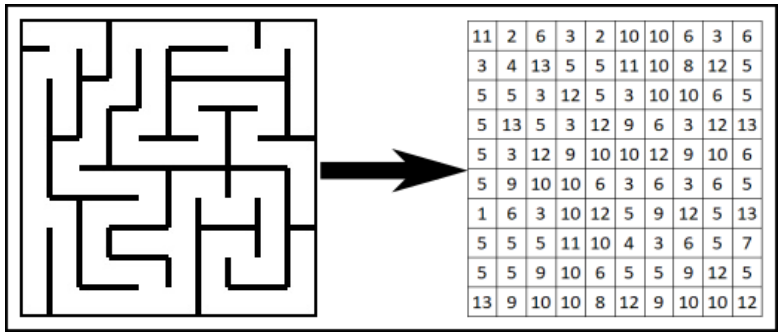

Figure 7: 2D array for a Maze using the Cell Numbers

---

2. Implementation of Solution

- Download the following zip file containing the files for Task 1B. Right-click on the hyperlink an select **Save Link As...** option to download.
  - **task_1b_detect_and_encode_maze.zip**
- You will find the following files/folder in the zip file:
  - ***test_cases*** folder - it contains all the maze images you will use to test your solution
  - ***task_1b.py*** file - you will implement your solution in this file
  - ***test_task_1b.py*** file - you will test your solution using this file
  - ***task_1b_cardinal.pyc*** - this file will help in testing the solution

---

Note: Do not tamper with the ***task_1b_cardinal.pyc*** file. It is required to be in the same directory / folder as the ***test_task_1b.py*** in order to test your solution.

Given

Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

A set of **10 maze images** are given in the ***test_cases*** folder. An example image is shown in Figure 8.

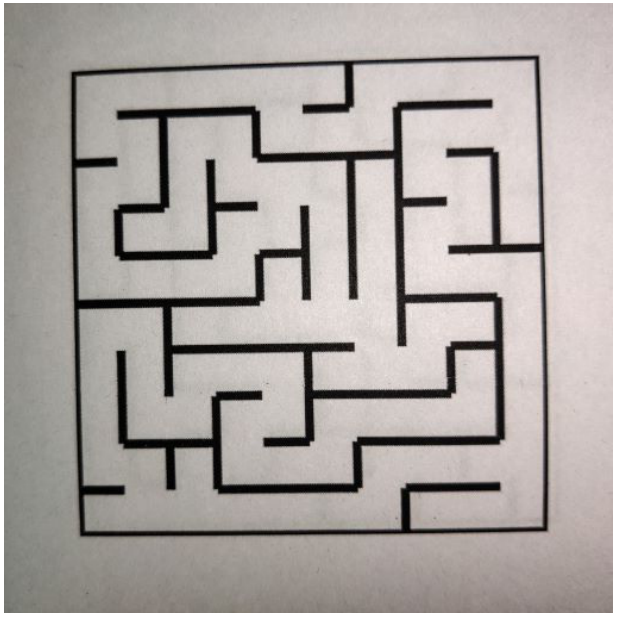

Figure 8: Example Maze Image

Problem Statement

- Use OpenCV techniques to detect the maze in the above given image and convert it into a ***2D maze array*** with appropriate ***Cell Numbers*** as shown in Figure 7.
- The `maze_array` is to be generated in the form of a ***nested list***.
- For example, if the given maze image is as in Figure 7, the `maze_array` will be:

```
maze_array = [[11, 2, 6, 3, 2, 10, 10, 6, 3, 6],
              [3, 4, 13, 5, 5, 11, 10, 8, 12, 5],
              [5, 5, 3, 12, 5, 3, 10, 10, 6, 5],
              [5, 13, 5, 3, 12, 9, 6, 3, 12, 13],
              [5, 3, 12, 9, 10, 10, 12, 9, 10, 6],
              [5, 9, 10, 10, 6, 3, 6, 3, 6, 5],
              [1, 6, 3, 10, 12, 5, 9, 12, 5, 13],
              [5, 5, 5, 11, 10, 4, 3, 6, 5, 7],
              [5, 5, 9, 10, 6, 5, 5, 9, 12, 5],
              [13, 9, 10, 10, 8, 12, 9, 10, 10, 12]]
```

Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

Instructions

- There are **two** functions pre-written in `task_1b.py` which *you have to modify*.
- The first function is `applyPerspectiveTransform()`.

Function Name	`applyPerspectiveTransform()`
Purpose	Takes a maze test case image as input and applies a Perspective Transfrom on it to isolate the maze
Input Arguments	`input_img` : [***numpy array***] maze image in the form of a numpy array
Output Arguments	`warped_img` : [***numpy array***] resultant warped maze image after applying Perspective Transform
Example Call	***warped_img = applyPerspectiveTransform(input_img)***

- Figure 9 shows the ***warped_img*** output from the function `applyPerspectiveTransform` on an example maze image.

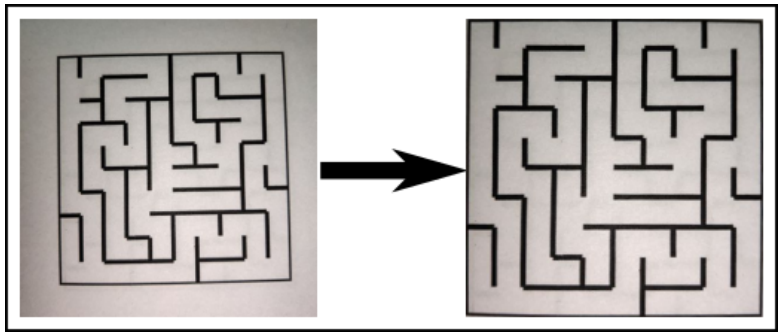

Figure 9: Output of applyPerspectiveTransform function

- The second function is `detectMaze()`.

Function Name	`detectMaze()`
Purpose	Takes the warped maze image as input and returns the maze encoded in form of a 2D array
Input Arguments	`warped_img` : [***numpy array***] resultant warped maze image after applying Perspective Transform

Welcome to NB theme!
Rulebook
Task 0
Task 1
CoppeliaSim Tutorials
1A - Explore OpenCV
1B - Detect and Encode Maze
1C - Design Ball Balance Platform
Task 2
Task 3
Task 4
Task 5
Practice Task
Instructions for Task 6
Task 6 Scene Details
Coding Standard
Git and GitHub
Live Session 1 - 24th October 2020
Live Session 2 - 21st November 2020
Live Session 3 - 12th December 2020
Live Session 4 - 10th January 2021
Changelog

Function Name	`detectMaze()`
Output Arguments	`maze_array` : [***nested list of lists***] encoded maze in the form of a 2D array
Example Call	***maze_array = detectMaze(warped_img)***

- Figure 10 shows the ***maze_array*** output from the function `detectMaze` for the ***warped_img*** input as in Figure 9.

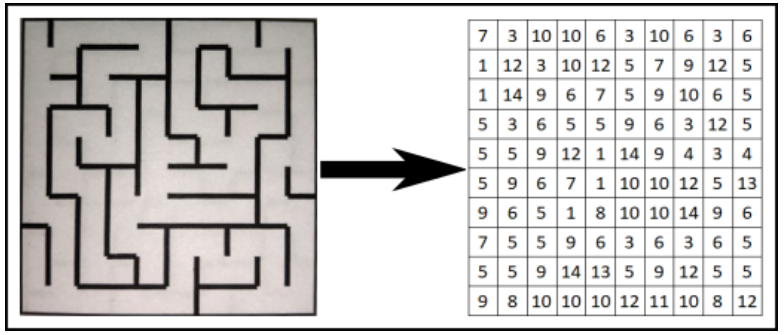

Figure 10: Output of detectMaze function

Running your solution

- To test and run your solution, do the following:
  1. Open Anaconda Prompt or Terminal and navigate to the directory / folder **task_1b_detect_and_encode_maze** on your system.
  2. Activate the Conda environment created in Task 0.
  3. Run the command: `python task_1b.py` to execute your solution. Your should get an output similar to Figure 11 below.

Welcome to NB theme!

Rulebook

Task 0

Task 1

CoppeliaSim Tutorials

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

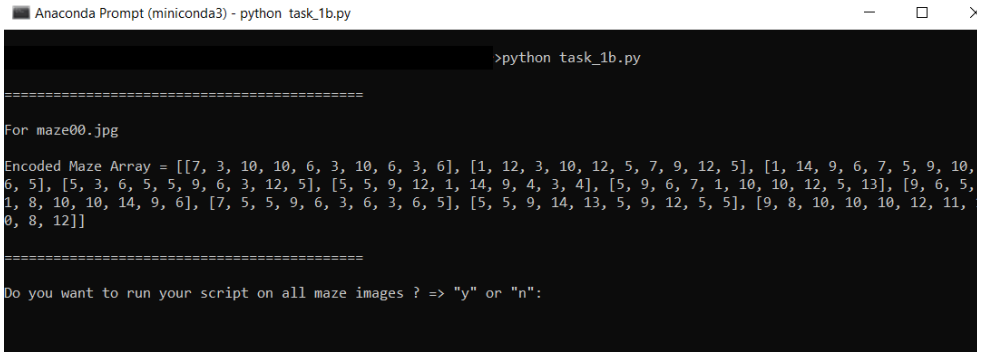

Figure 11: Executing task_1b.py

4. You can choose to run your script for rest **9 maze images** from ***test_cases*** folder by providing "**y**" as an input.

---

3. Testing the Solution

- The final step before submitting **Task 1B** is to test and evaluate your solution.
- For testing your solution, run the script ***test_task_1b.py*** in Anaconda Prompt / Terminal with the command: `python test_task_1b.py` .
- If your `task_1b.py` script executed without any errors, you will see the output resemble Figur 12.

Welcome to NB theme!

Rulebook

Task 0

Task 1

CoppeliaSim Tutorials

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

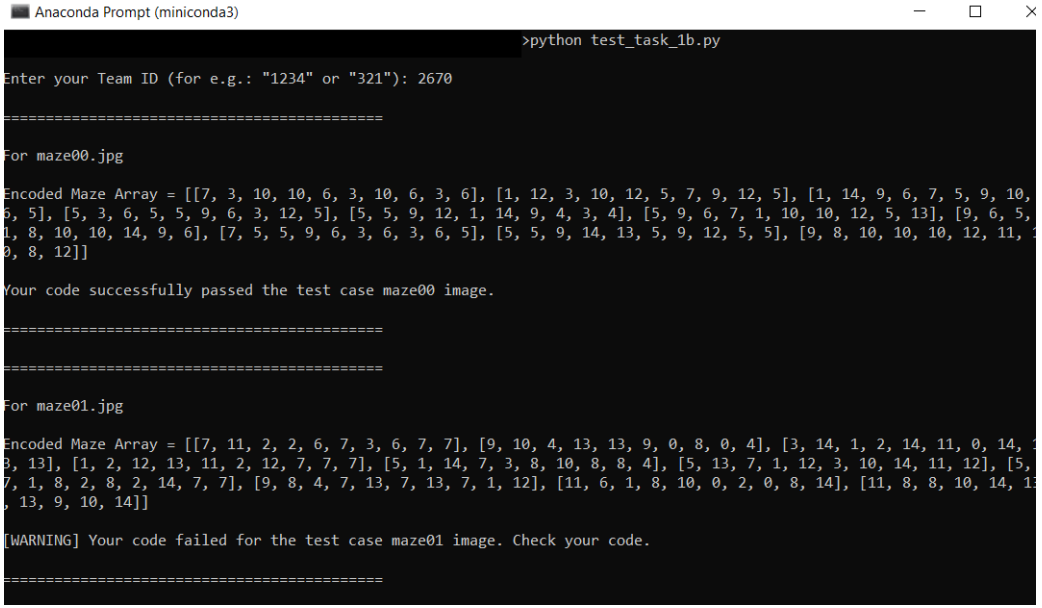

Figure 12: Executing test_task_1b.py

Note: Make sure that your code `task_1b.py` passes all the test case maze images given in the ***test_cases*** folder.

- If your code executed without errors, you will also see a new file ***task_1b_output.csv*** generate in the **task_1b_detect_and_encode_maze** folder. This output file is encrypted and should resemble Figure 13.

Welcome to NB theme!

Rulebook

Task 0

Task 1

CoppeliaSim Tutorials

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

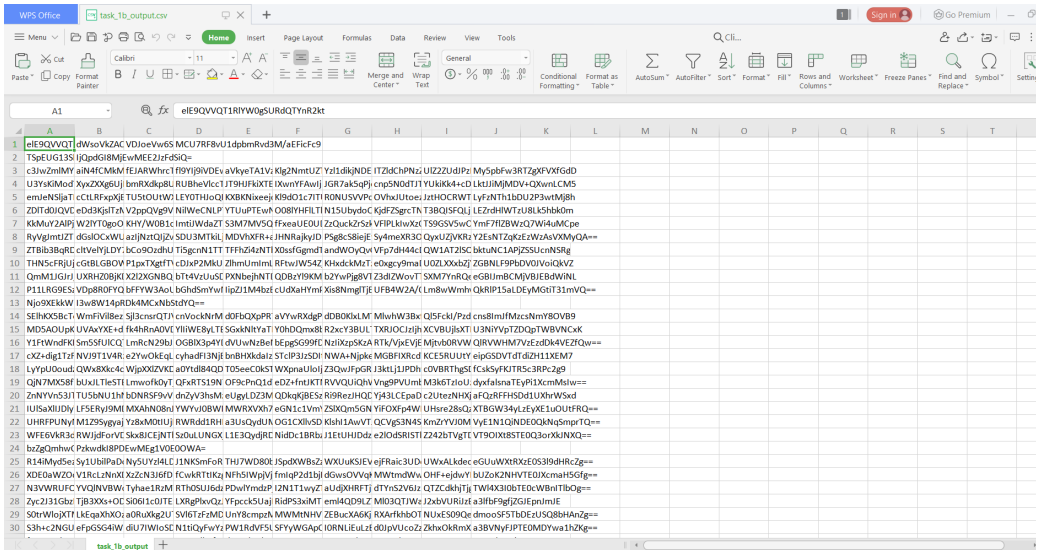

Figure 13: Generation of task_1b_output.csv file

Submission Instructions

For **Task_1B submission** you have to upload a **.zip** file. To create the appropriate file please follow instructions given below:

1. Create a new folder named `NB_<Team-ID>_Task_1B` .
    - For example: if your team ID is **9999** then you need to create a folder named `NB_9999_Task_1B` .
2. Now copy and paste following files into this folder:
    - `task_1b.py` (with modified **applyPerspectiveTransform()** and **detectMaze()** functions)
    - `task_1b_output.csv` (generated after running `test_task_1b.py` in ***task_1b_detect_and_encode_maze*** folder)
3. Compress this folder into a `NB_9999_Task_1B.zip` file.
4. Now go to the eYRC Portal and follow the instructions to upload this **.zip** file for **Task_1B** as shown in Figure 14.

Welcome to NB theme!

Rulebook

Task 0

Task 1

CoppeliaSim Tutorials

1A - Explore OpenCV

1B - Detect and Encode Maze

1C - Design Ball Balance Platform

Task 2

Task 3

Task 4

Task 5

Practice Task

Instructions for Task 6

Task 6 Scene Details

Coding Standard

Git and GitHub

Live Session 1 - 24th October 2020

Live Session 2 - 21st November 2020

Live Session 3 - 12th December 2020

Live Session 4 - 10th January 2021

Changelog

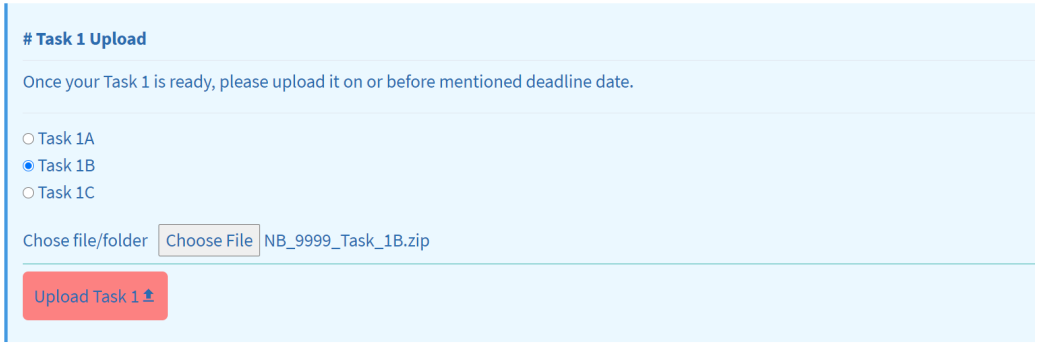

Figure 14: Submission of **NB_9999_Task_1B.zip** file on eYRC portal

5. Congrats, you have successfully completed Task 1B !

ALL THE BEST !!
