Task 4 - Description of task_4.py

The file **task_4.py** is a Python Client script that will help you in implementing Task 4.

It is somewhat similar to the **task_1b.py** script you submitted in Task 1B of Stage 1. There are these three functions in **task_4.py**: **connect_to_server()**, **send_to_receive_from_server()** and **find_new_path()** which are same as provided in **task_1b.py** script earlier. Team is expected to complete these functions as submitted in Stage 1.

The **task_4.py** script is mainly divided into two parts:

1. **Python Client** (that will talk to the ESP32 server on the robot to solve Task 4)
2. **User input** (that waits for the user to indicate the need for Reposition / Restart)

These two parts are running simultaneously as independent **threads**. The **main** function of **task_4.py** takes care of creating these threads and running them.

NOTE: Do not edit the main function of task_4.py file.

1. Python Client (that will talk to the ESP32 server on the robot to solve Task 4)

This part, as defined in the function **python_client()** of the **task_4.py** does the following:

- Imports **task_1a.py** and **image_enhancer.pyc** files. The **task_1a.py** present inside the **Codes** folder is the skeleton file provided in the Task 1A of Stage 1. Team is expected to either replace the **task_1a.py** file with the one submitted during Stage 1 or complete the **readImage()** and **solveMaze()** functions in the one provided now.
- It then changes the global variable **CELL_SIZE** of **task_1a.py** from **20** to **40**, since the cells in the image **Task4_maze.jpg** are of 40 x 40 pixels.
- Connects to the ESP32's Server IP address: **192.168.4.1** and the Port address: **3333**. It does this by calling the **connect_to_server()** function.
- According to the **Section 6** of Rulebook, the Python Client must send
  - the list of all digits in maze image
  - combination of digits for sum with their locations
  - the shortest path

  to the robot wirelessly using socket.

- But, as mentioned in the Problem_Statement.pdf, a list having all digits in image: **digits_list = [8, 0, 2, 2]** and a dictionary of combination of digits for sum with their locations: **combination = {2: (6, 9), 2: (8, 6)}** are **already provided for Task 4**.
- Since, such kind of dictionary cannot be declared in Python where we have duplicate entries of keys. Hence, we have created a function in **task_4.py** named **create_combination_dict()** which takes two inputs:
  - a list of integers having digits in the combination: **combination_digits = [2, 2]**
  - a list of tuples having respective locations of digits in the combination:

    combination_locations = [(6, 9), (8, 6)]

  The function takes care of such case and returns a dictionary as this:

  combination = {2: [(6, 9), (8, 6)]}.

  For example, if combination_digits = [2, 2, 8] and

  combination_locations = [(6, 9), (8, 6), (3, 5)]

  are passed to the create_combination_dict() function, it will return:

  combination = {2: [(6, 9), (8, 6)], 8: (3, 5)}.
- Team has to then edit the **python_client()** function in order to implement Task 4 according to the Problem_Statement.pdf. You are allowed to edit this function where the comments with "**NOTE:**" are provided. You should not edit any other part of this function. If found so, this will lead to the disqualification of your team.
- Make sure that while sending the above data from Python Client to ESP32's server on robot, it is sent in proper format: **#<data from client to server>#**.
- Similarly, while receiving the data from ESP32's server on robot to Python Client, it is received in proper format: **@<data from server to client>@**.

2. User input (that waits for the user to indicate the need for Reposition / Restart)

This part, as defined in the function **take_input_for_reposition_restart()** of the **task_4.py** does the following:

- Waits for the user to input either: **%** (for **Restart**) or **&** (for **Reposition**).
- If any other character is given as input, it will give the message on Terminal:

  You must enter either "%" OR "&" only !

  and waits again for the user input.

- If **%** is given as input, it will give the message on Terminal:

  One Restart for the run is taken !

  It then sends this user input to the ESP32's server on robot using **sock.sendall()** function, since for Restart, **%** should be sent as is and not like: **#%#**.

  It then terminates both the threads running **python_client()** and **take_input_for_reposition_restart()** and also the **task_4.py**.

  The robot should be placed again at the **NEW_START** location **(4,4)**, run the **task4.py** again but **no need to close the GUI_App**. The GUI will automatically update its parameters for the next run.

- If **&** is given as input, it will give the message on Terminal:

  One Reposition for the run is taken !

  It then sends this user input to the ESP32's server on robot using **sock.sendall()** function, since for Reposition, **&** should be sent as is and not like: **#&#**.

  Team should decide in which of the previously traversed cell the robot should be placed.

  Team can edit this part of the function **take_input_for_reposition_restart()** for implementing Task 4.