|  |  |  |
| --- | --- | --- |
| **Project Title** | **Connect4-AI** | |
| **Section C & A** | **Registration Number** | **Name** |
| Member-01 | 0090-BSCS-2019 (Sec-C) | M. Zeeshan Haider |
| Member-02 | 0112-BSCS-2019 (Sec-A) | Dil Awaiz Mehwish |

## **Installation Requirement (if any)**

* For Example: Random (import random)
* import pygame
* import sys
* import numpy as np
* import math
* import random

**Why you choose this algorithm (Advantage of Algorithm that you had use in your project**)

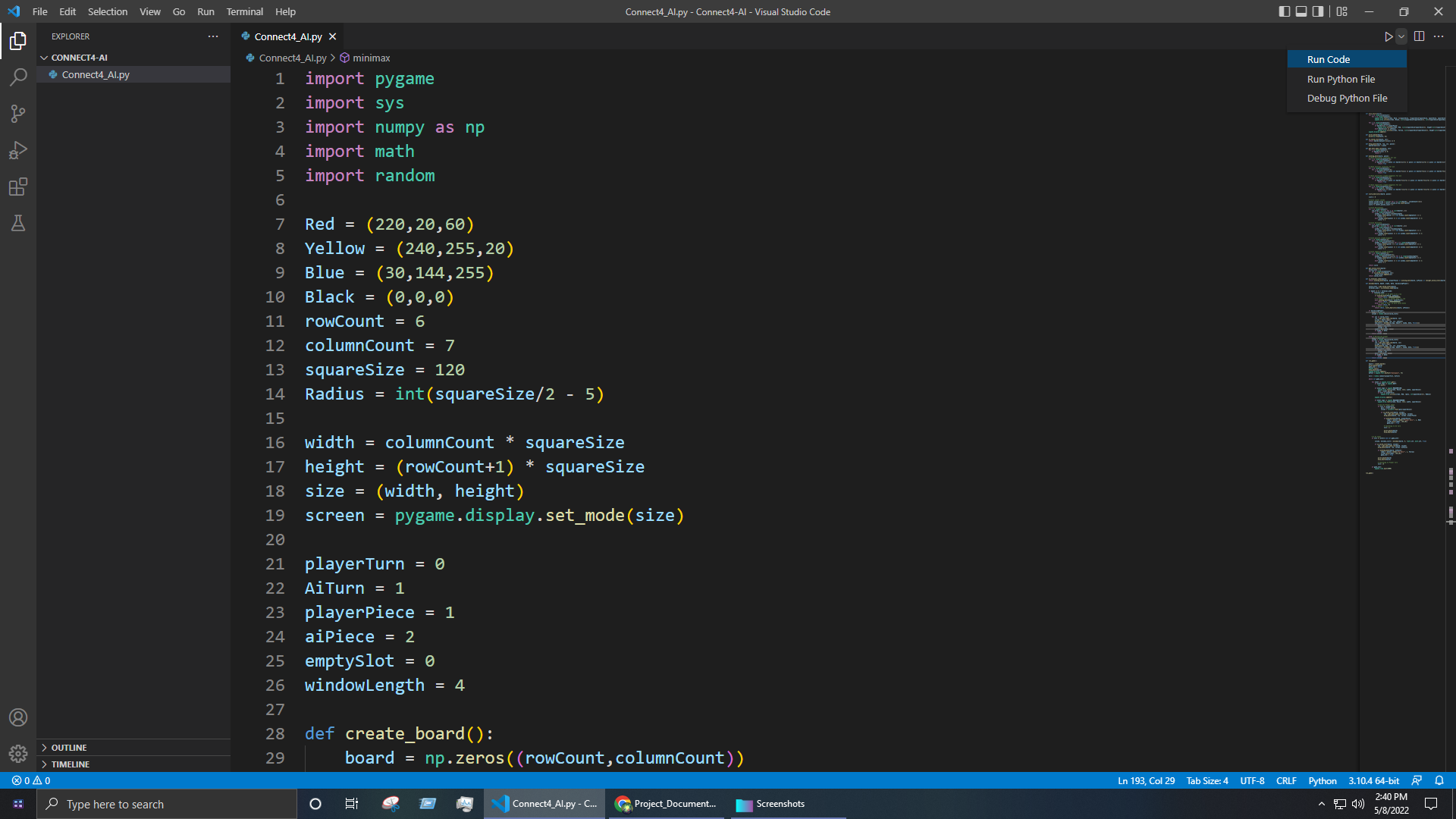
(Maximum 200 words)

For board games like Connect-4, Tic-Tac-Toe and Checkers etc, Minimax algorithm along with Alpha-Beta pruning is the best option. Minimax algorithm is a beneficial problem-solving algorithm that helps perform a thorough assessment of the search space. It makes the working of two-player games much smoother, and zero chances of any glitch due to the Minimax algorithm’s accuracy. However, the most prominent advantage it offers is that it makes it possible to implement decision making in Artificial Intelligence. Minimax algorithm along with Alpha-Beta pruning is extremely beneficial, as it reduces the computation time and makes the search go deeper in the game tree, quickly. Moreover, Alpha-Beta pruning cutoff or prunes the evaluation in the game tree, when even a single possibility is found that proves the current move is worse than the previously examined move.

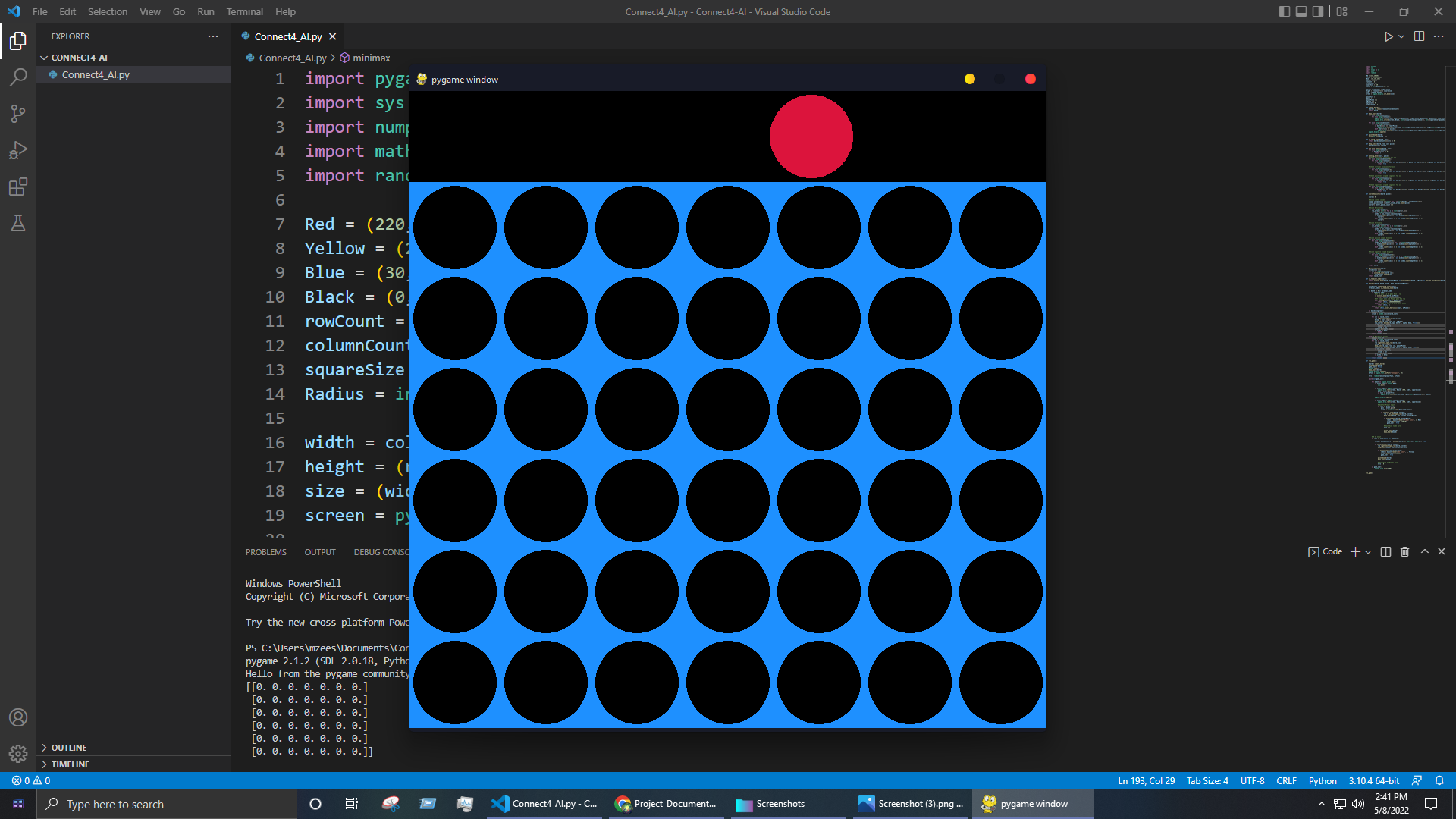
**Step by step visualization of Project.**

(Add screenshot of your project with details about how to run project successfully)

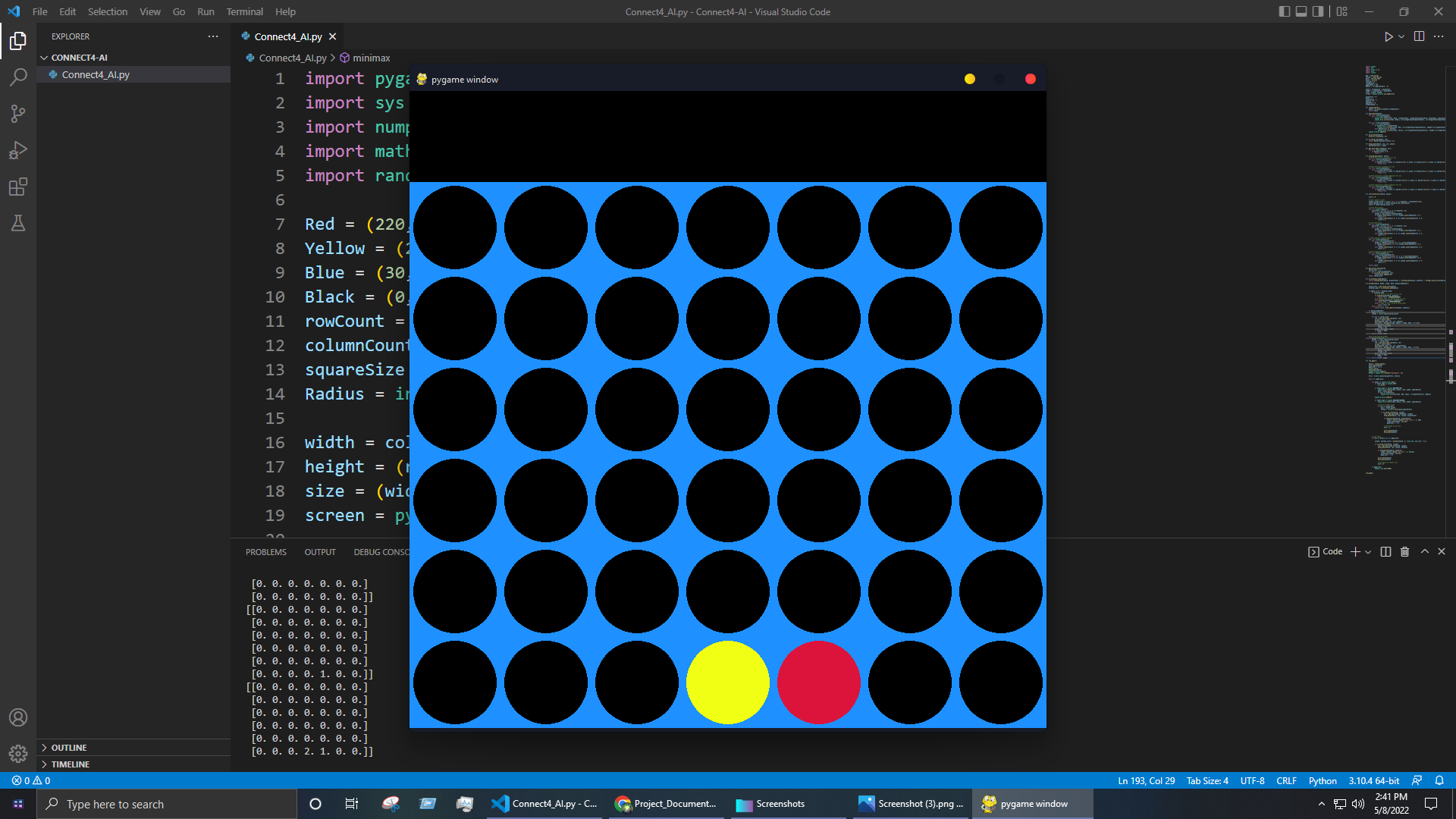
1. Open the project folder in Visual Studio Code and click ‘Run Code’



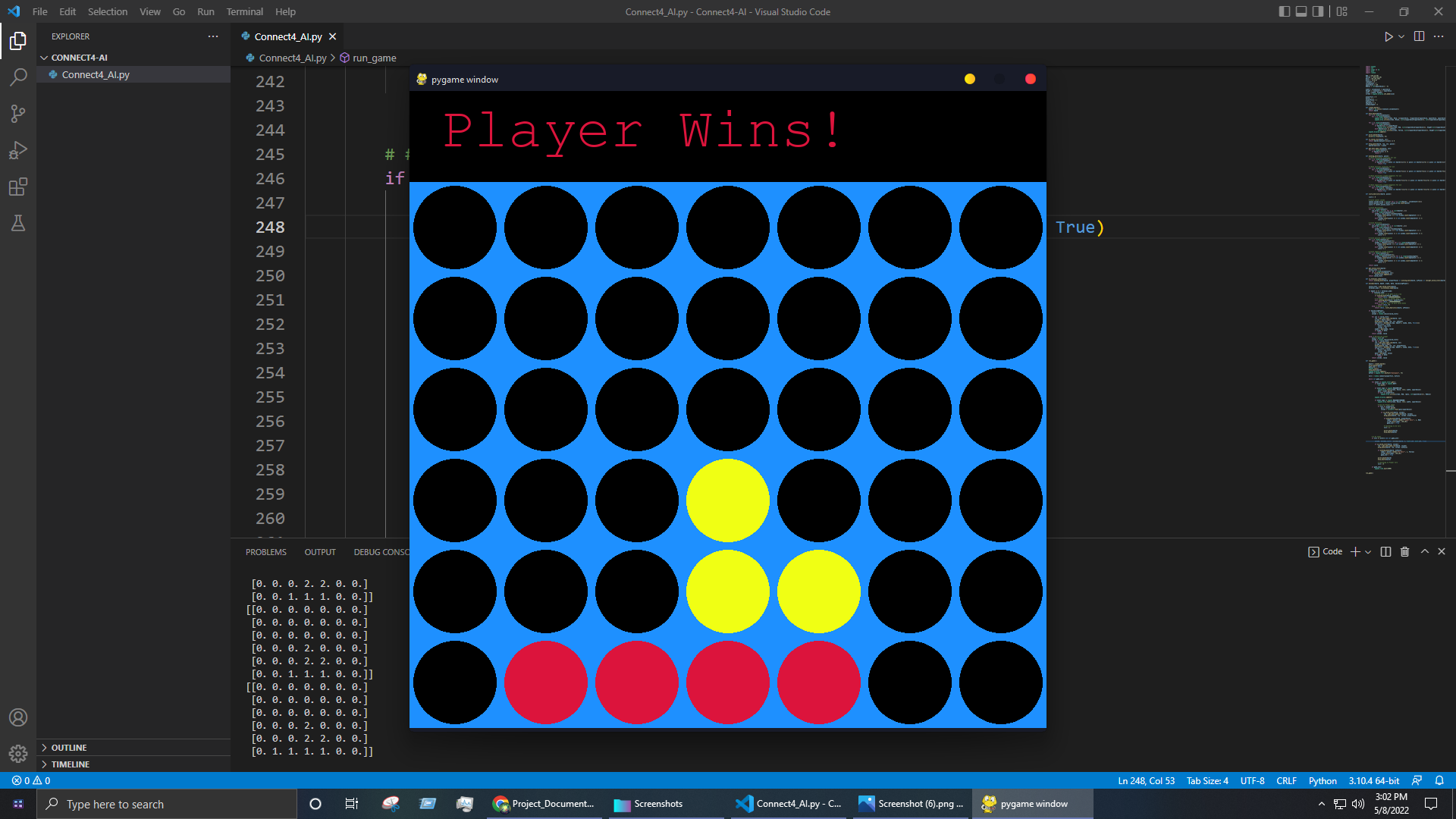
1. Move mouse cursor left or right to move the piece

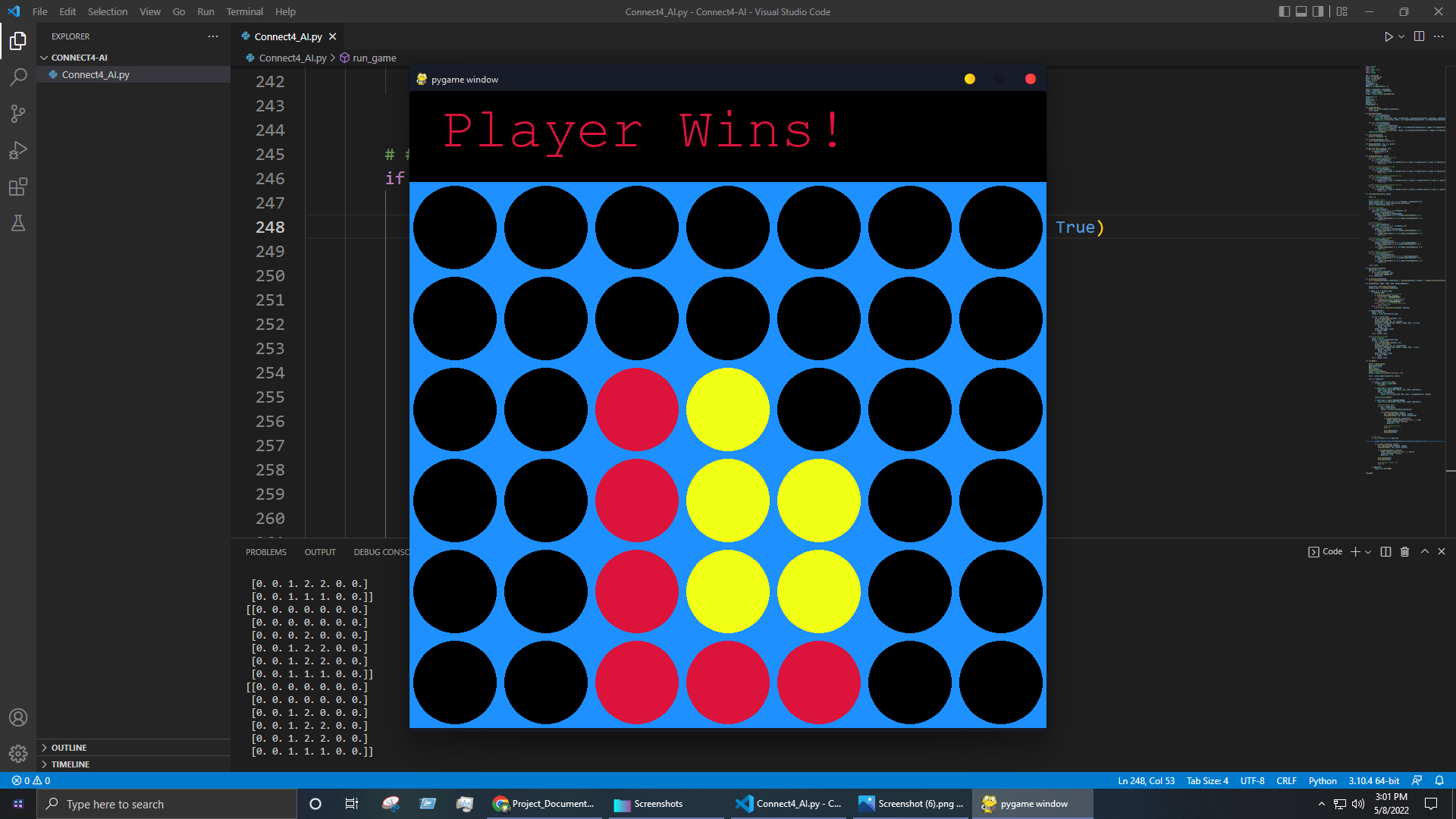


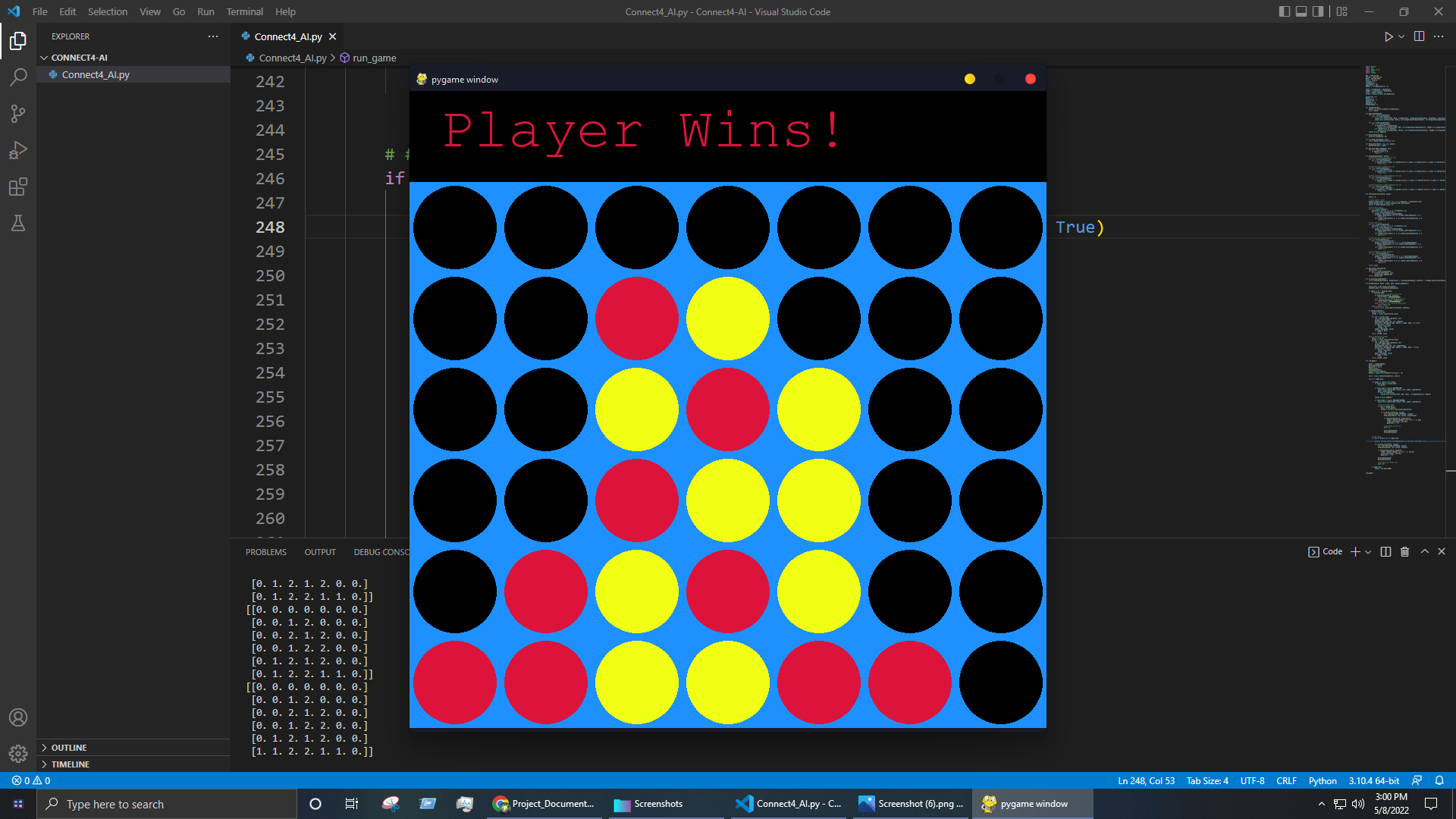
1. Left click or right click on mouse to drop the piece

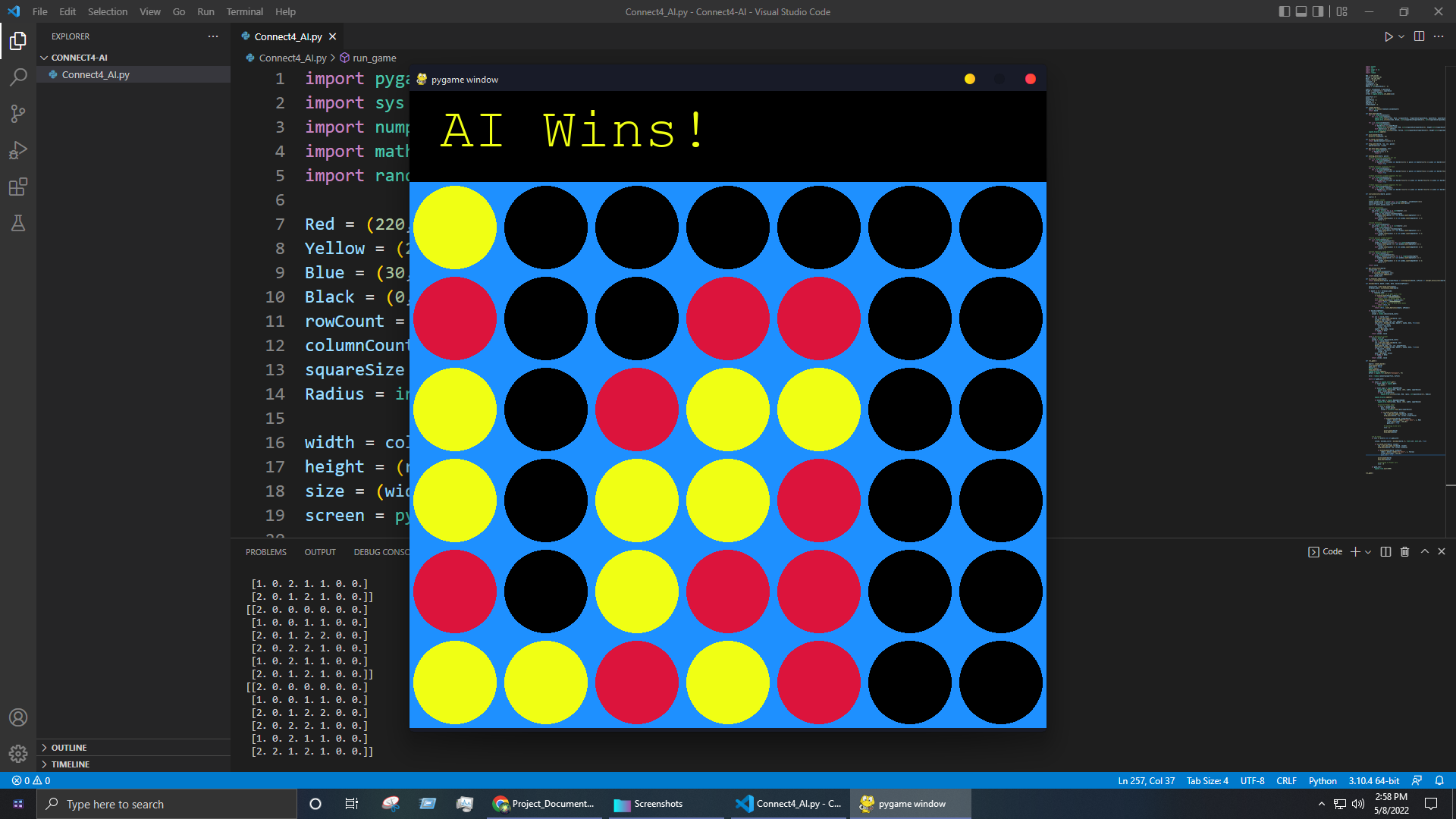


1. Connect 4 pieces horizontally, vertically or diagonally before AI to win the game









**Write challenges of project that you had face while implementation.**

The challenges we faced were:

* How to implement horizontal & vertical score heuristic for AI
* How to implement positive & negative sloped diagonal score heuristic for AI
* How to implement center column score heuristic for AI
* How to implement a function to get all the valid slots available
* How to implement a function to get the terminal node for Minimax algorithm
* How to implement the Maximizing player & Minimizing player
* How to implement Alpha-Beta pruning

**What are the limitation of the Project?**

The limitation of the project is the depth of the Minimax algorithm i.e. how deep the Minimax algorithm can expand and explore the game tree. It depends upon the processing power of the system running the algorithm. We are currently running the Minimax algorithm with depth ‘5’ on our system and it’s working fine but it might start to lag or even freeze another system with low processing power so we have to decrease the depth of the Minimax algorithm in order to make it run smoothly but it will also greatly affect the AI’s accuracy to make decisions.