## LAB ONE: REPORT ON AGENT EXECISE

## STUDENT NAME: ENYONG ABIGAIL

## Task 1: Introduction

## Brief explanation of architecture of the solution.

This piece of code sets up the foundation for a grid-based simulation environment where an agent like a character or robot can move around from a start point till it locates its target.

## TASK 2: set up infrastructure.

The IDE used for the development of this agent was PyCharm in which I was able to install and import libraries such as;

Pygame: for creating the visual display

Random: use foe generating random positions

Queue: for pathfinding algorithms and

Enum: for creating organized categories of different cells

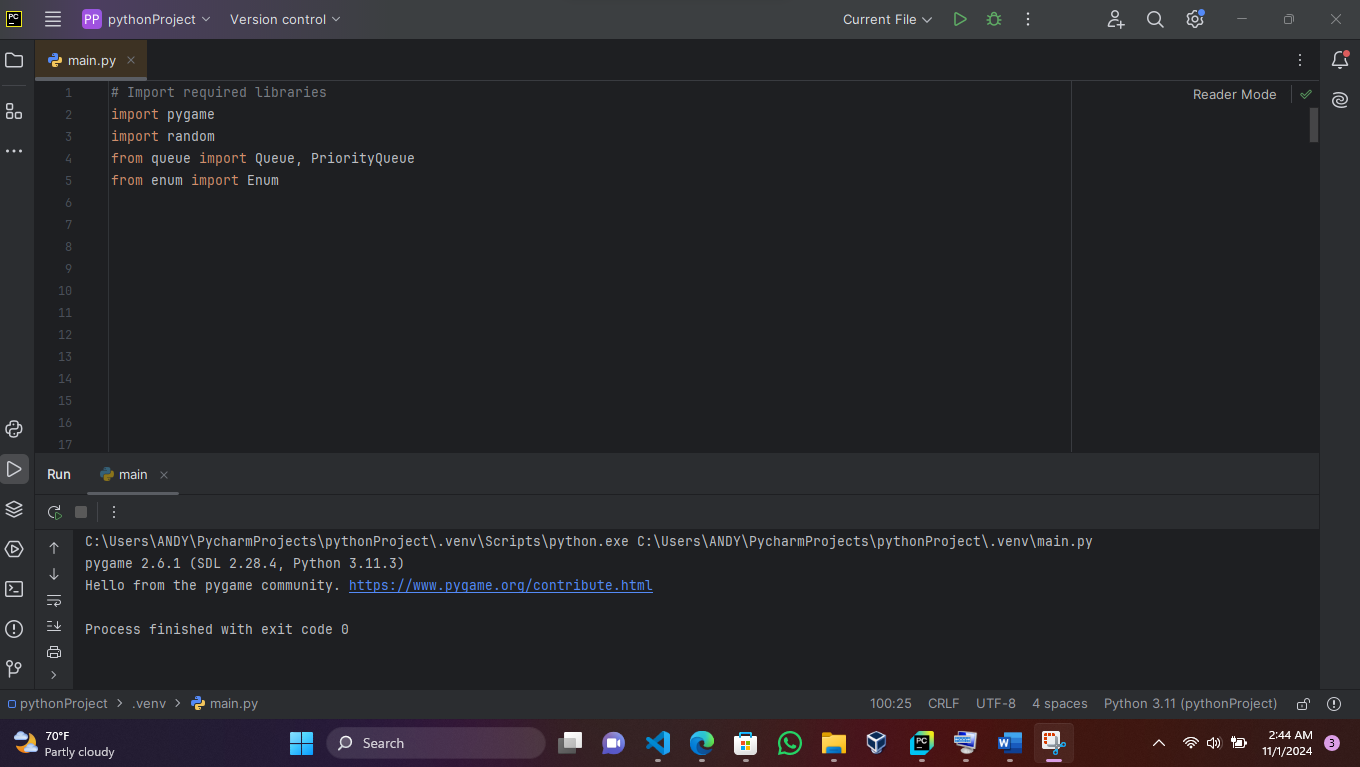


Fig 1: importation of libraries.

## TASK 3: Description of the agent functionality

The code defines two main structures. Firstly, it starts by creating a CellType class that defines what each cell in the grid can be:

empty space (0),

obstacle/wall (1),

target/goal (2),

agent/player (3) and

part of a path (4).

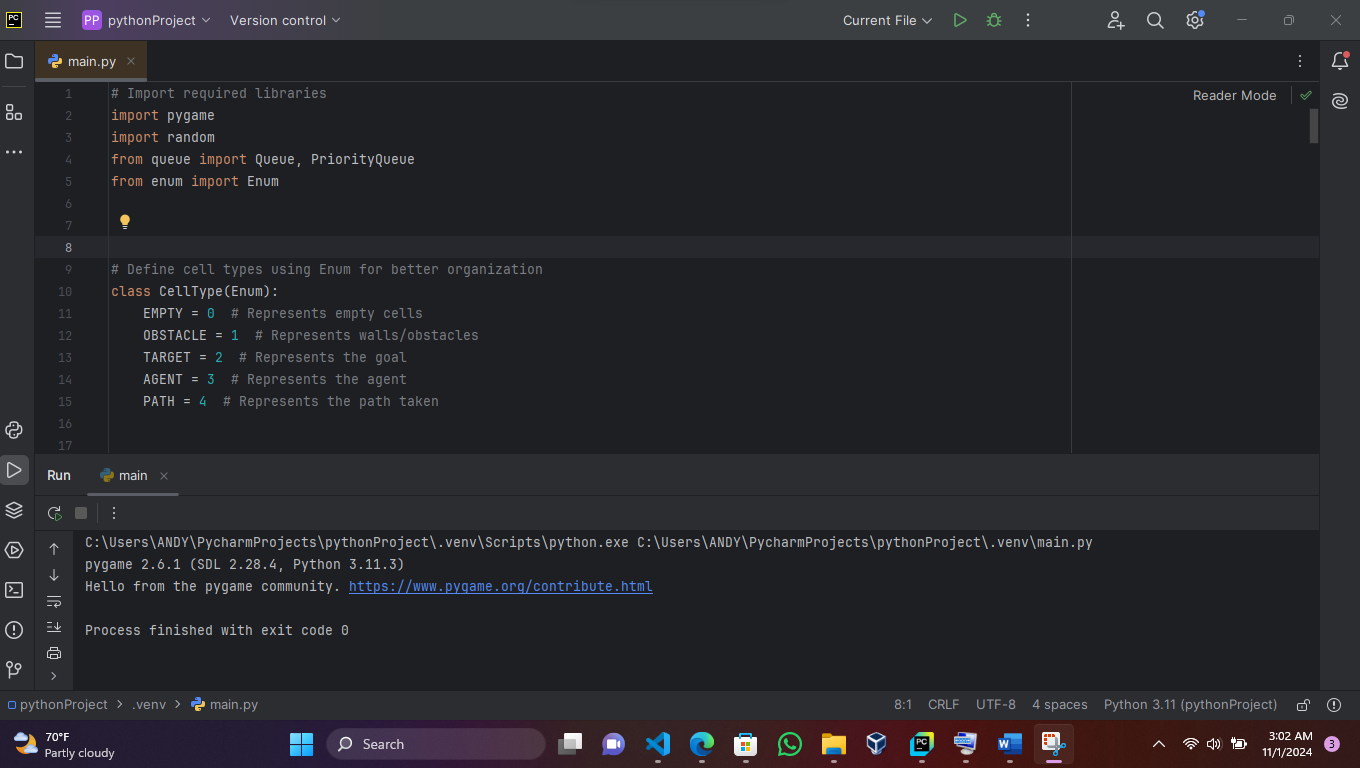


Fig 2: definition of celltypes.

It then goes ahead to set up a COLORS dictionary that matches each cell type to a specific color for display purposes - empty cells are yellow, obstacles are green, the target is red, the agent is voilet, and the path is blue.

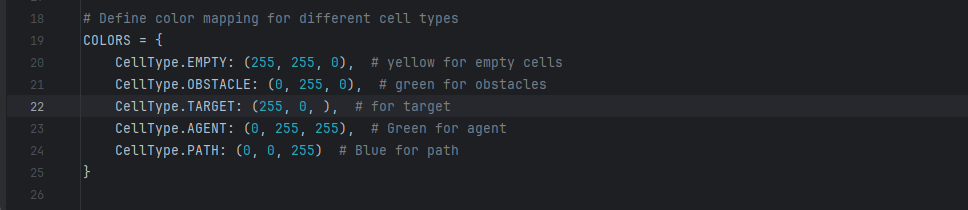


Fig 3: assigning color to each cell type.

The Environment class is where the main setup happens. When creating a new environment, it takes three inputs: width and height which is used to determine the grid size and cell size to determine how big each cell appears on screen, the default which is 30 pixels. The class creates an empty grid using these dimensions, where each cell starts as EMPTY. It also prepares to track the positions of the target.

The code achieves its purpose by using nested lists (a grid) to represent a 2D environment where each cell can be one of the defined types.

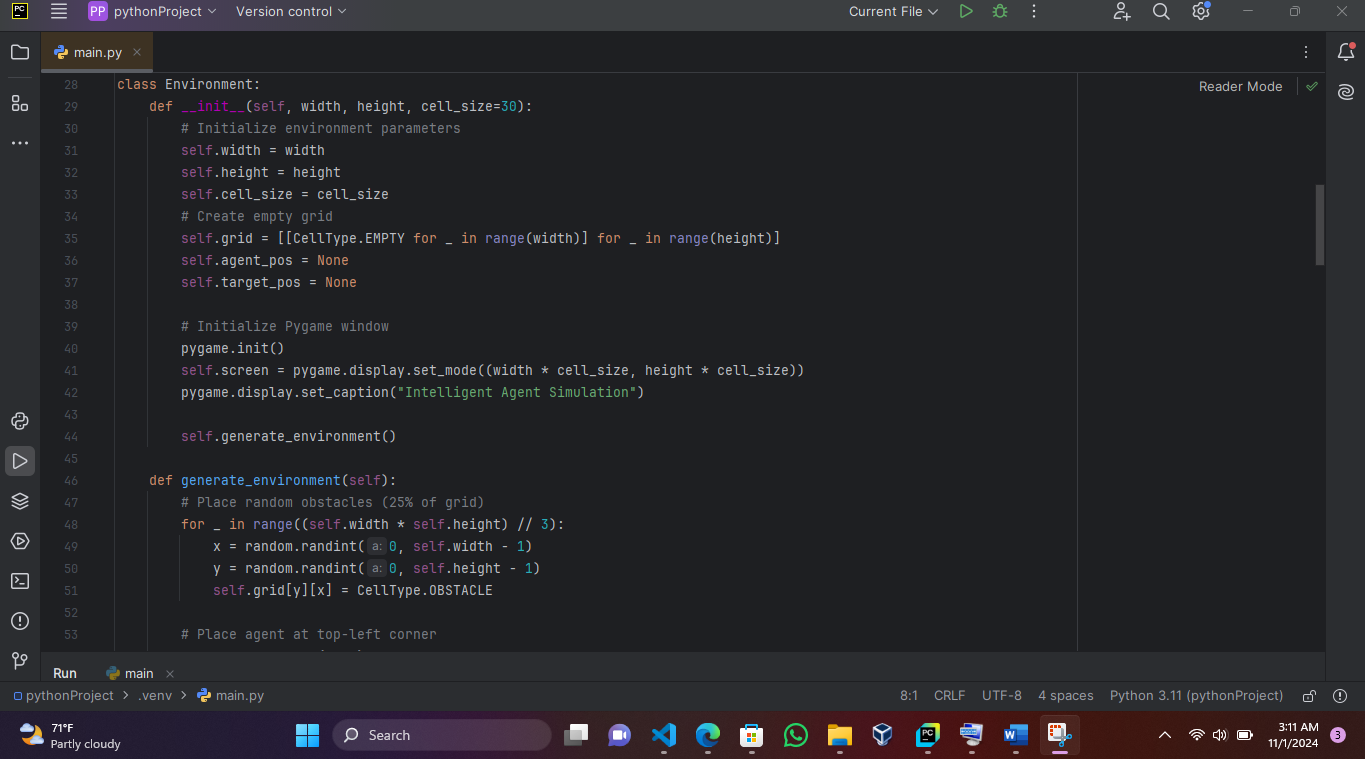


Fig 4: showing environment.

The main data structure is the grid, which is created as a 2D list where each position can be accessed using [y][x] coordinates. This is similar to how you might think of coordinates on a map, but using computer grid coordinates where (0,0) starts at the top-left corner.

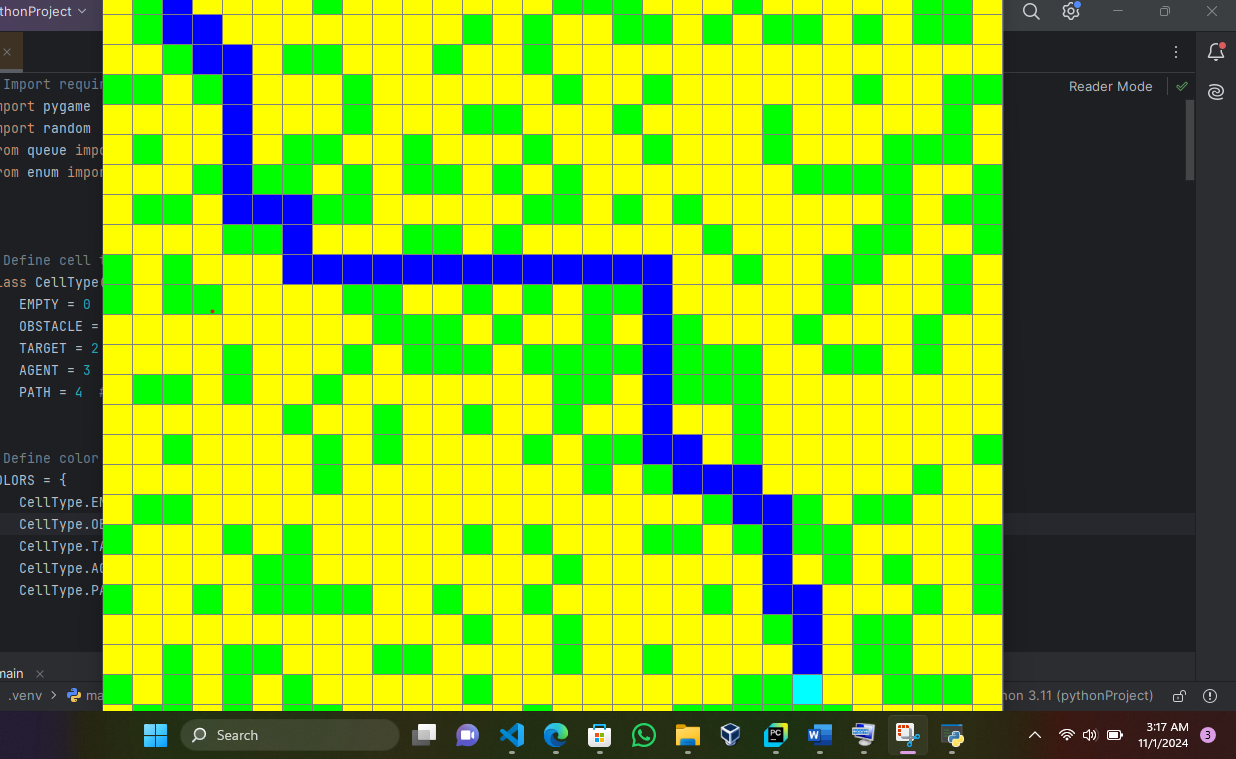


Fig 5: Demonstration of agent locating its targe

This code serves as the foundation for what appears to be a pathfinding simulation, where an agent will need to navigate through obstacles to reach a target. While this section doesn't show the actual movement or pathfinding logic, it creates all the necessary structure and rules for such a simulation to work.