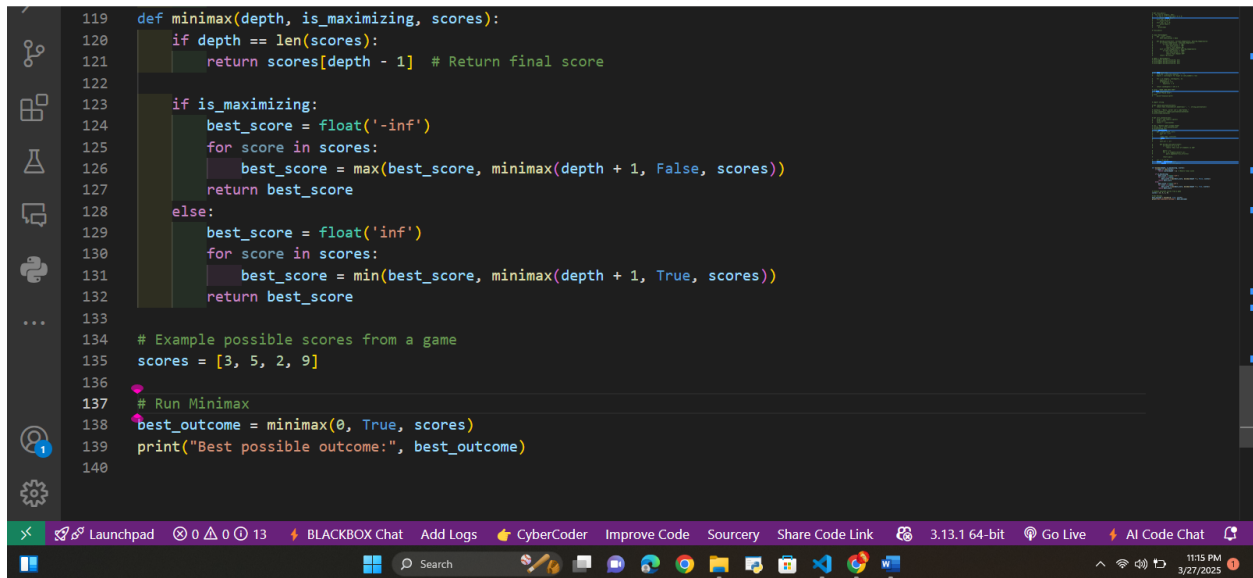


TASK 8

Name: Sami Imran

Roll No: 033

A screenshot of a code editor with a dark theme. The editor displays a Python function named 'minimax' that implements a minimax algorithm. The function takes three arguments: 'depth', 'is_maximizing', and 'scores'. It uses recursion to calculate the best score for a given depth and maximizing/minimizing player. The code includes comments and a test case at the bottom. The editor's interface includes a sidebar with icons for file explorer, search, and other tools. The bottom status bar shows various system and application icons, including a search bar, taskbar, and system clock indicating 11:15 PM on 3/27/2025.

```
119 def minimax(depth, is_maximizing, scores):
120     if depth == len(scores):
121         return scores[depth - 1] # Return final score
122
123     if is_maximizing:
124         best_score = float('-inf')
125         for score in scores:
126             best_score = max(best_score, minimax(depth + 1, False, scores))
127         return best_score
128     else:
129         best_score = float('inf')
130         for score in scores:
131             best_score = min(best_score, minimax(depth + 1, True, scores))
132         return best_score
133
134 # Example possible scores from a game
135 scores = [3, 5, 2, 9]
136
137 # Run Minimax
138 best_outcome = minimax(0, True, scores)
139 print("Best possible outcome:", best_outcome)
140
```

Simple Minimax Algorithm in Python

The **Minimax Algorithm** is a strategy used in two-player games where one player wants to **maximize the score** (MAX) while the other tries to **minimize it** (MIN). It checks all possible moves and picks the best one for each player.