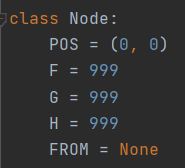
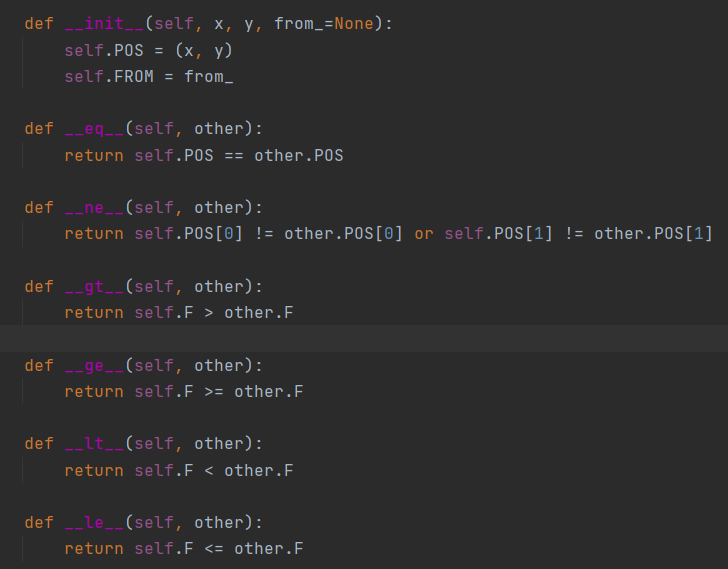
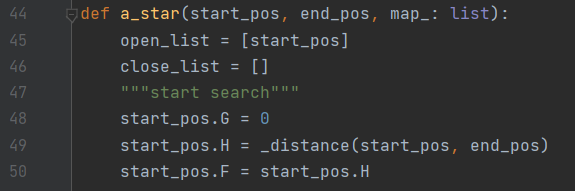
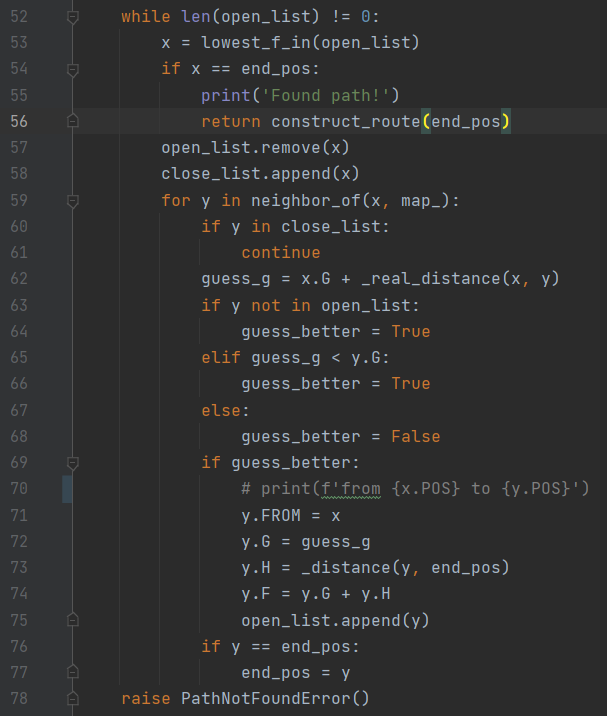
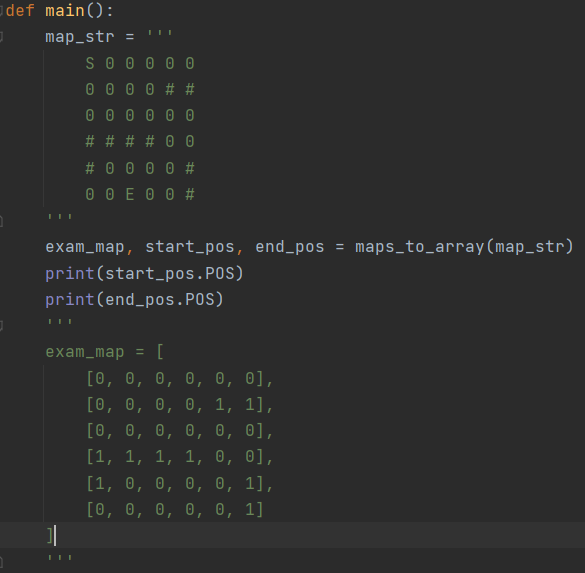
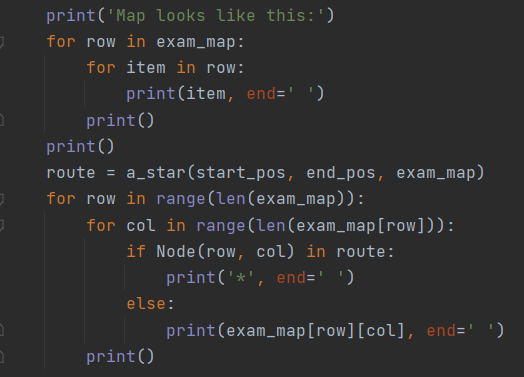
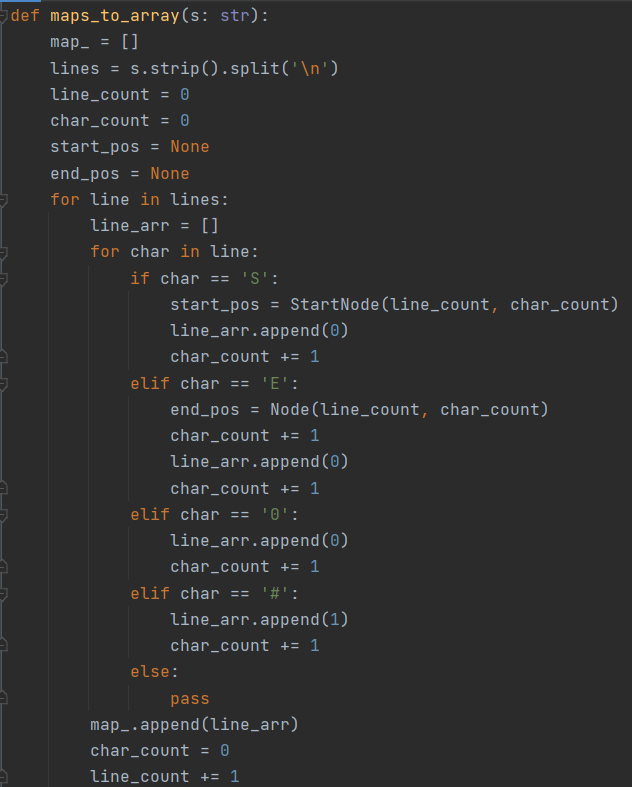
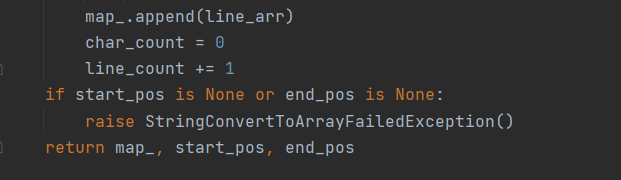
4107056006 資工三 游庭瑋

A\* 演算法報告

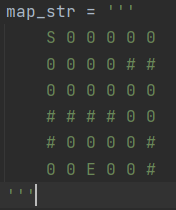
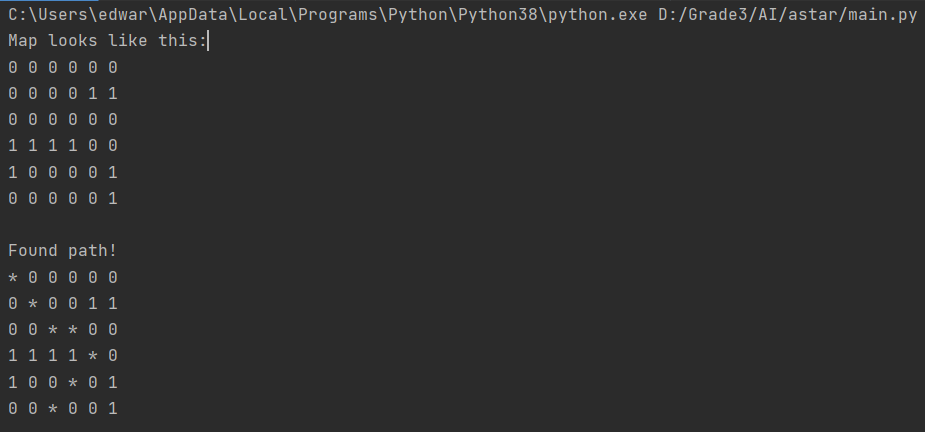
1. 目的：  
    本作業目的在於利啟發式演算法來達成尋找從起點到終點的最短路徑。有別於Dijkstra 演算法，A\* 並沒有先將所有路徑資訊進行考量，而是由現有資訊與推測資訊去找出最好的選擇，並查詢有沒有因為這個選擇造成曾經選過得路徑也有更好的選擇
2. 實做部份：  
    這次我選用python作為實做的語言。首先先定義每個座標點的類別，property應有位置、F、G、H數值及父節點，如下圖：  
     
    接下來定義constructor 及重載各類操作符，方便之後對節點的操作  
     
    再來是演算法本身：  
     
    我們首先將起始點放入open\_list，代表其尚未被搜尋，並建造一個空的close\_list，並將起始點的G設為0，表示與起始點距離為0；H設為起點到終點的歐拉距離；F即為兩者之和。  
    a. 接下來要先找到open\_list裡F最小的節點x，檢查看是不是終點。將其移出open\_list，放入close\_list代表尋找過。  
    b. 接著將它周圍的九宮格的每一個點開始測量，如果已經在close\_list，及已經搜尋過則跳過。否則將其G假定為從x到其的距離（斜角1.4，直線1）加上x原本的G值。  
    c. 接下來檢查它有沒有在open\_list裡，如果沒有則代表它是完全未接觸過的點，可以優先考慮；否則若新的路徑比較短也可以優先考慮，不然就不考慮。接著更新它的數值。如果它已經是終點了也要紀錄下來。重複以上a. b. c.直到open\_list是空的，即沒有可以考慮的點了。

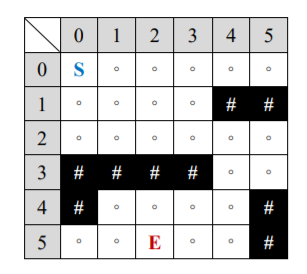
main函數：  
  
  
即簡單驅動astar演算法及顯示結果

輔助將文字轉換為使用格式的副程式

1. 執行結果：  
   輸入：

若以原地圖來看則是：  


ㄒ

ㄒ